CGTM Number 207
September 1982

A SCHEME FOR GENERATING, VERIFYING, AND UNLOADING SYSTEM-INDEPENDENT SOURCE TAPES UNDER VM/CMS

ROBERT C. BEACH
COMPUTATION RESEARCH GROUP
STANFORD LINEAR ACCELERATOR CENTER
STANFORD, CALIFORNIA 94305

TABLE OF CONTENTS

SECTION 1: INTRODUCTION

This document describes a group of programs that run under the VM/CMS operating system on the IBM 3081 computer at SLAC. One program, TAPEGENR, will generate tapes in a reasonably system-independent form. These tapes may be used to transport text files to other computing systems. The tape may be labeled or unlabeled but is always a 9 track tape. Files on the tape may contain fixed or variable length records, be blocked or unblocked, and use EBCDIC or ASCII coding. In addition, individual modules in a file may optionally be separated by IEBUPDTE control records. IEBUPDTE is an IBM utility program running under the OS/VS2 operating system. A second program, TAPEVERF, may be used to verify the tape by generating a listing of any of the files on the tape. Finally, a third program, TAPEUNLD, may be used to unload a tape written in this format, that is, the modules may be moved from tape to disk. The programs are invoked by EXEC programs which submit batch jobs.

In reading this document, it is suggested that the examples in Section 1.2 be read first. This may be enough information for many uses of the programs; if not, the detailed descriptions in Section 2 should be studied.

SECTION 1.1: RATIONALIZATION FOR THIS WORK

For many years, SLAC ran the OS/VS2 operating system and its predecessors on its IBM computers. Under OS/VS2, a large number of utility programs were available to generate, verify, and unload tapes in many formats. The basic IBM utilities included IEHMOVE, IEBCOPY, IEBGENER, IEBPTPCH, and IEBUPDTE. The WYLBUR Text Editor included utilities such as WLIBLIST, WPRESS, and WUNPRESS. In addition, there were general utilities like IOPROGM. It must be admitted that these formed a terribly inconsistent overlapping mess, but at least there was always a utility that made it relatively easy to manipulate a tape.

When an IBM 3081 computer was acquired, SLAC, for some arcane and stupid reason, decided to use the VM/CMS operating system. VM/CMS is an abysmally deficient operating system which is totally unsuited to SLAC's needs. For instance, it does not supply anything equivalent to the OS utilities and only supplies the most primitive tools to try to re-create these utilities. Since SLAC moved to VM, a number of people have attempted to re-create the parts of these utilities that they needed. This is another of those attempts.

SECTION 1.2: SOME SIMPLE EXAMPLES

Suppose it is desired to create a tape with three files. The first file is to contain a documentation file named DOC999 of type MEMO on the user's 191 disk. The second file is to contain all of the files on the user's 192 disk with a type of FORTRAN or ASSEMBLE, except those files whose name begins with "TEMP". The third file is to be created from a MACLIB named ASMMACS on the user's 192 disk. The first file is to contain only the data records, but the other files are to have individual modules separated by IEBUPDTE control records. Suppose that the user's account is WXYZ and the tape has a label of WXYZ01. Also suppose that the first file on tape is to have a name of DOCUMENT, the second a name of SOURCE, and the third a name of MACROS.

The first step for the user is to prepare a file of specification statements similar to the following:

```
..TAPE  WXYZ01 DEN=6250
..DISK  WXYZ 191 B
..DISK  WXYZ 192 C
..FILE  DOCUMENT
..MODS  DOC999 MEMO B
..EOF
..FILE  SOURCE UPDTE
..XCLD  TEMP* * C
..MODS  * FORTRAN C
..MODS  * ASSEMBLE C
..EOF
..FILE  MACROS UPDTE
..MODS  ASMMACS MACLIB C
..EOF
```

The first record describes the tape and specifies that it is a 6250 bit per inch tape instead of the default 1600. The next two records, the DISK statements, make the 191 and 192 disks of user WXYZ available to the tape generation program as the B and C disks. These disk mode letters, B and C, apply to a batch job that will be submitted and do not apply to the interactive session that submitted the batch job. The next three records define the first file, then comes five records for the second file, and three more records for the third file. The FILE statements initiate the start of a file on tape and an EOF statement terminates it. The MODS statement gives the names of disk files to be incorporated into the file on tape and the XCLD statement may be used to exclude certain files. By default, the files are fixed blocked with a record length of 80 and a block size of 4000. Since no translation is requested, the files are written in un-translated EBCDIC.

The second step for the user is to perform the VM/CMS operations:

```
GIME RCB 196
TAPEGENR
```

The first command makes the disk with the tape processing programs available. The second command invokes an EXEC program which will ask the user a few questions and submit a batch job. One of the questions will be the name of the file containing the

previously described specification statements. A listing of the file, along with any error messages, will either be sent to your terminal or written to one of your disks. The batch job will return a CON (console log) and PRT (print) file to your reader where you may examine them.

Now suppose the user wishes to verify this tape. Verification consists of reading the tape and generating a printed listing of the contents of the tape. In this case, the user may create a file like the following:

```
..TAPE  WXYZ01 DEN=6250
..FILE  DOCUMENT
..      TITLE='FILE CONTAINING THE DOCUMENT'
..FILE  SOURCE UPDTE
..      TITLE='FILE CONTAINING FORTRAN AND ASSEMBLER PROGRAMS'
..FILE  MACROS
..      TITLE='FILE CONTAINING ASSEMBLER LANGUAGE MACROS'
```

Notice that the FILE statements were continued on a second line. The TITLE data will become part of the heading on the printed listing. When this file has been prepared, the user issues the command:

```
TAPEVERF
```

and proceeds as before. The batch job will again return a CON and PRT file to your reader. The PRT file will contain the printed listings of all of the files and may be printed or sent to the microfiche recorder. In the case shown above, each module in the second file will begin on a new page, but the modules in the third file will be printed continuously because the UPDTE option is not present.

Finally, let us suppose that the user wishes to unload the first and third files from this tape onto disk. In this case, a file consisting of:

```
..TAPE  WXYZ01 DEN=6250
..FILE  DOCUMENT DNAME=DOC999 DTYPE=MEMO
..SKIP  1
..FILE  MACROS UPDTE DTYPE=COPY
```

is prepared and the command:

```
TAPEUNLD
```

is issued. The second record in the file listed above causes the first file on tape to be unloaded onto disk and given a name of DOC999 and a type of MEMO. The third record causes one file to be skipped on the tape. The final line causes the modules in the third file on tape to be unloaded with their names taken from the IEBUPDTE control records and their type being set to COPY. The batch job will return a CON file, a PRT file, and the unloaded files in your reader. The unloaded files may then be moved to a disk using the CARD LOAD command. The original MACLIB can be re-created from the COPY files.

SECTION 2: A DETAILED DESCRIPTION OF THE PROGRAMS

The following sections contain a detailed description of the TAPEGENR, TAPEVERF, and TAPEUNLD programs. The properties of the programs themselves are described along with the specification statements that they will accept.

The programs are all on the 196 disk in the RCB account. Access to these programs may be obtained by the command:

```
GIME RCB 196
```

The required operation may then be initiated by entering the name (TAPEGENR, TAPEVERF, or TAPEUNLD) of the program. This will invoke an EXEC program which will:

1. Ask you for the name of the file containing the specification statements.
2. Allow you to select your terminal or a disk file for a listing of the specification statements along with any error messages.
3. Perform a complete check of the syntax of your specification statements.
4. Submit a batch job to process the tape.

The batch job will return a CON (console log) and PRT (print) file to your reader. These files should always be examined for error messages. In the case of the TAPEUNLD function, the unloaded files will also be returned to your reader.

Specification statements begin with a double period in columns one and two and a three or four character identification immediately following that. Only the first 72 characters of each specification record are used. A specification record may be continued on the next record by beginning the continuations with a double period followed by a blank. The maximum length of a continued specification statement is 512 characters. This means that seven records may be used if they are the full 72 characters long; more records are permitted if the length of the records is smaller.

In general, the specification statements consist of a few required positional parameters and a number of optional parameters. The positional parameters must come first in the given order but the optional parameters may be given in any order. In the following description, angle brackets <...> enclose descriptive material. Thus <number> is to be replaced with a number on an actual specification statement. Square brackets [...] indicate optional parameters. Finally, braces and vertical lines {...|...|...} are used to indicate that a choice must be made from the alternatives. Thus {SL|NL} means that either SL or NL should be used on an actual specification statement.

The IEBUPDTE control records that these programs will generate and process are threefold. First, each individual module is preceded by an ADD record. The normal form of an ADD record is:

```
./      ADD   NAME=<module-name>
```

If the module has an alias (TXTLIB members may have aliases) then

the module is followed by one or more ALIAS records of the form:

```
./       ALIAS NAME=<alias-name>
```

Finally, the last record in the file has the form:

```
./       ENDUP
```

Note that IEBUPDTE is limited to fixed length, 80 character, records while these programs can handle a much wider range of formats.

SECTION 2.1: THE TAPEGENR PROGRAM

The TAPEGENR program can create tapes with an arbitrary number of files. When a tape is being written, it is possible to skip over the initial files of an existing tape and start writing after a given number of files. Normally, this program should be used to write tapes that contain source files only. However, TEXT and TXTLIB files can also be written. It is not possible to write out MODULE or WYLBUR edit format files because their logical record length is too large. Each file may consist of an arbitrary number of VM/CMS files concatenated together. When MACLIB or TXTLIB files are written to tape, they are broken down into their individual members and control information in the libraries is not written out. As an option, individual members of a file may be separated by IEBUPDTE control records. A large number of translation options may be applied to any of the files. The tapes may be labeled or unlabeled 9 track tapes of any valid density. Blocking factors on the tape may have any value allowed by VM/CMS.

When writing a file to tape, it is possible for a record in a file on disk to be longer than the logical record length (LRECL) on the tape. When this happens, the following occurs:

1. The first (LRECL-8) bytes concatenated with "**CONT**" are written out.
2. The next (LRECL-8) bytes are written out in the same format until less than LRECL bytes remain to be written.
3. The final bytes are written out.

When fixed length records are being written out, short records are padded on the right with blanks.

The TAPEGENR program will accept nine different specification statements: COM, CTRL, DISK, TAPE, SKIP, FILE, XCLD, MODS, and EOF. These statements will be described below.

The COM specification statement is used to insert a comment into the specification file. A COM statement can occur anywhere except between continuation records. The form of the COM statement is:

```
..COM   <anything>
```

The text on the COM statement is ignored.

The CTRL specification statement is used to control some diagnostic output. A CTRL statement may occur anywhere. The form of the CTRL statement is:

```
..CTRL [{DIAG|NODIAG}]
```

The parameters on the CTRL specification statement are:

{DIAG|NODIAG} These items turn the diagnostic print on or off.

The DISK specification statement is used to make a disk available to the program that writes the tape. The form of the DISK statement is:

```
..DISK <user> <disk> <mode>
       [MWPASS=<multi-write-password>]
```

The parameters on the DISK specification statement are:

<user> <disk> <mode> These positional parameters are the user identification, disk identification, and mode of the disk. The mode parameter is a mode letter that you are assigning with this statement; it applies to the batch job that will be submitted and not to the interactive session that submits the job. The modes of all of the specified disks must be distinct and must be between B and P.

MWPASS=<multi-write-password> The multi-write password of the disk. This is only necessary when you wish to write files whose mode number is zero.

The TAPE specification statement is used to describe the tape. Only one TAPE statement is permitted and it must come before any FILE statements. Only 9 track tapes can be processed. The form of the TAPE statement is:

```
..TAPE <vol-ser>
       [LABEL={SL|NL}]
       [DEN={800|1600|6250}]
```

The parameters on the TAPE specification statement are:

<vol-ser> This positional parameter is the volume serial number of the tape.

LABEL={SL|NL} The labeling mode of the tape. The default is standard labeling (SL).

DEN={800|1600|6250} The tape density. The default is 1600 bits per inch.

The SKIP specification statement is used to skip a group of files at the beginning of the tape before starting to write. A SKIP statement is optional, but if it occurs, it must be between the TAPE statement and the first FILE statement. The form of the SKIP statement is:

```
..SKIP <number>
```

The parameter on the SKIP specification statement is:

<number> The number of files to be skipped.

The FILE specification statement is used to describe a file that will be written to the output tape. The table of excluded file names, defined by the XCLD statements, is cleared when a FILE statement is encountered. A FILE statement must be preceded by a

single TAPE statement. The form of the FILE statement is:

```
..FILE  <name>
        [RECFM={F|FB|V|VB}]
        [LRECL=<number>]
        [BLKSIZE=<number>]
        [{UPDTE=<two-characters>|UPDTE|NOUPDTE}]
        [TRANSL={0|...|8}]
```

The parameters on the FILE specification statement are:

<name> This positional parameter is the file name of the output file. The file name is the FID under VM/CMS or the DSNAME under OS/VS2. This name is limited to 8 characters, must begin with a letter, and contain only letters and numerals. The file names must be distinct for each file on the tape.

RECFM={F|FB|V|VB} The record format of the output file. The default is fixed blocked (FB).

LRECL=<number> The logical record length of the output file. The default is 80 bytes. The minimum value is 16 and the maximum is 256.

BLKSIZE=<number> The block size of the output file. The default is 4000.

{UPDTE=<two-characters>|UPDTE|NOUPDTE} An item that specifies whether or not the modules in the file are to be separated by IEBUPDTE control records. UPDTE=xx means that ADD, ALIAS, and ENDUP records are to be generated with "xx" in columns 1 and 2. UPDTE is an abbreviation for UPDTE=./ and the default is NOUPDTE.

TRANSL={0|...|8} This item gives the translation option to be applied to each record. These options are described in Section 2.4. The default is TRANSL=0 which indicates that no translation is to take place.

The XCLD specification statement is used to supply the names of disk files to be excluded from tape. This statement works with the MODS statement to select the files to be written to tape. An XCLD statement must be bracketed between a FILE and EOF statement and must precede the MODS statements to which it applies. An arbitrary number of XCLD statements may occur between a FILE and EOF statement. The form of the XCLD statement is:

```
..XCLD  <anything-compatible-with-LISTFILE>
```

Thus the parameters on the statement will normally consist of three items which the VM/CMS LISTFILE command will match with all known file names. It is not possible to exclude members of TXTLIB or MACLIB files. If you only want to write out part of one of these files, you may make a copy of the file, delete the required members, and then write the copy to tape.

The MODS specification statement is used to supply the names of modules to be written to tape. Modules selected by a MODS statement but excluded by an XCLD statement are not written to tape. A MODS statement must be bracketed between a FILE and EOF statement pair and an arbitrary number of MODS statements may occur between these statements. The form of the MODS statement is:

```
..MODS  <anything-compatible-with-LISTFILE>
```

Thus, the parameters on the statement will normally consist of three items which the VM/CMS LISTFILE command will match with all known file names.

The EOF specification statement is used to terminate a file. Exactly one EOF statement must follow each FILE statement. The form of the EOF statement is:

```
..EOF
```

No parameters are permitted on an EOF statement.

SECTION 2.2: THE TAPEVERF PROGRAM

The TAPEVERF program can print the contents of a tape created by TAPEGENR, or any similar tape. Any individual files on the tape may be selected for printing and the listing may be limited to a given number of records at the beginning of the file.

Each record in a file is identified on the listing by its record number. If a record is longer than 120 characters, the remainder of the record is printed on succeeding lines. The beginning and end of each record is indicated by the vertical line character "|". If the specification statements indicate that IEBUPDTE control records are present, each member will start at the top of a new page. The output listing may include a user-specified title for each file. The title on the page includes the name of the file and the member name if IEBUPDTE control records are present. The title also includes up to three page numbers: the page number of the run, of the file, and of the module. The last page is a summary of the files printed during the run.

The TAPEVERF program will accept five different specification statements: COM, CTRL, TAPE, SKIP, and FILE. The COM, CTRL, and TAPE specification statements are exactly the same as those for TAPEGENR. The SKIP and FILE statements will be described below.

The SKIP specification statement is used to skip a group of files. Files anywhere on the tape may be skipped. The form of the SKIP statement is:

```
..SKIP  <number>
```

The parameter on the SKIP specification statement is:

```
<number>  The number of files to be skipped.
```

The FILE specification statement is used to describe a file that will be read from the input tape. A FILE statement must be preceded by a single TAPE statement. The form of the FILE statement is:

```
..FILE  <name>
        [RECFM={F|FB|V|VB}]
        [LRECL=<number>]
```

```
        [BLKSIZE=<number>]
        [{UPDTE=<two-characters>|UPDTE|NOUPDTE}]
        [TRANSL={0|...|8}]
        [LIMIT=<number>]
        [TITLE=<anything>]
```

The parameters on the FILE specification statement are:

- <name> This positional parameter is the file name of the input file. The file name is the FID under VM/CMS or the DSNAME under OS/VS2. This name is limited to 8 characters, must begin with a letter, and contain only letters and numerals. The file names must be distinct for each file on the tape.
- RECFM={F|FB|V|VB} The record format of the input file. The default is fixed blocked (FB).
- LRECL=<number> The logical record length of the input file. The default is 80 bytes. The minimum value is 16 and the maximum is 256.
- BLKSIZE=<number> The block size of the input file. The default is 4000.
- {UPDTE=<two-characters>|UPDTE|NOUPDTE} An item that specifies whether or not the modules in the file are separated by IEBUPDTE control records. UPDTE=xx means that ADD, ALIAS, and ENDUP records are present with "xx" in columns 1 and 2. UPDTE is an abbreviation for UPDTE=./ and the default is NOUPDTE.
- TRANSL={0|...|8} This item gives the translation option to be applied to each record. These options are described in Section 2.4. The default is TRANSL=0 which indicates that no translation is to take place.
- LIMIT=<number> The maximum number of records that will be read from the file. The default is to read the entire file.
- TITLE=<anything> A title that becomes part of the heading on each page. The title must be enclosed in apostrophes if it contains any blanks. An apostrophe within a title is indicated by a double apostrophe on the specification statement. The default is a line of 128 blanks.

SECTION 2.3: THE TAPEUNLD PROGRAM

The TAPEUNLD program can unload the contents of a tape created by TAPEGENR, or any similar tape, onto disk. Any individual files on the tape may be selected for unloading. Continued records created by TAPEGENR may be restored by this program. If the specification statements indicate that IEBUPDTE control records are present, each member will become an individual VM/CMS file. The VM/CMS file name of the unloaded file may be obtained from the specification statements or the IEBUPDTE ADD control records; the VM/CMS file type is always obtained from the specification statements. IEBUPDTE ALIAS records are ignored. The record format and logical record length of the unloaded file may be

specified independently of those on the tape. If the length of an unloaded record is longer than the logical record length of the VM/CMS file it is being written into, then the record is broken up using the same method that TAPEGENR used.

The files from the tape are always unloaded to the A disk of the batch worker machine. They will then automatically be sent to the user's reader queue with the CARD DUMP command. The user may retrieve them with the CARD LOAD command.

The TAPEUNLD program will accept five different specification statements: COM, CTRL, TAPE, SKIP, and FILE. The COM, CTRL, and TAPE specification statements are exactly the same as those for TAPEGENR. The SKIP specification statement is exactly the same as that for TAPEVERF. The FILE statement will be described below.

The FILE specification statement is used to describe a file that will be read from the input tape. A FILE statement must be preceded by a single TAPE statement. The form of the FILE statement is:

```
..FILE  <name>
        [RECFM={F|FB|V|VB}]
        [LRECL=<number>]
        [BLKSIZE=<number>]
        [{UPDTE=<two-characters>|UPDTE|NOUPDTE}]
        [TRANSL={0|...|8}]
        [{CONT|NOCONT}]
        [DNAME=<name>]
        [DTYPE=<type>]
        [DRECFM={F|V}]
        [DLRECL=<number>]
```

The parameters on the FILE specification statement are:

<name> This positional parameter is the file name of the input file. The file name is the FID under VM/CMS or the DSNAME under OS/VS2. This name is limited to 8 characters, must begin with a letter, and contain only letters and numerals. The file names must be distinct for each file on the tape.

RECFM={F|FB|V|VB} The record format of the input file. The default is fixed blocked (FB).

LRECL=<number> The logical record length of the input file. The default is 80 bytes. The minimum value is 16 and the maximum is 256.

BLKSIZE=<number> The block size of the input file. The default is 4000.

{UPDTE=<two-characters>|UPDTE|NOUPDTE} An item that specifies whether or not the modules in the file are separated by IEBUPDTE control records. UPDTE=xx means that ADD, ALIAS, and ENDUP records are present with "xx" in columns 1 and 2. UPDTE is an abbreviation for UPDTE=./ and the default is NOUPDTE.

TRANSL={0|...|8} This item gives the translation option to be

applied to each record. These options are described in Section 2.4. The default is TRANSL=0 which indicates that no translation is to take place.

{CONT|NOCONT} A flag to tell the program if it is to reconstruct continued records that were broken up by TAPEGENR. The default is NOCONT.

DNAME=<name> A name that will be used to construct the name of the output file on disk if IEBUPDTE control records are not available. The default is TEMPNAME.

DTYPE=<type> The type of the file that will be created. The default is TEMPTYPE.

DRECFM={F|V} The record format of the output file on disk. The default is fixed (F).

DLRECL=<number> The logical record length of the file. The default is 80 bytes. The minimum value is 16 and the maximum is 256.

SECTION 2.4: THE TRANSLATION OPTIONS

The basic translation function that these programs seek to provide is the ability to translate between EBCDIC and ASCII. If things were simple, only two translate functions would be necessary. Unfortunately, things are not simple, and eight functions are provided. This proliferation is the result of two problems.

The first problem comes from the "standard" EBCDIC-ASCII correspondence. This correspondence is the one invoked by the OPTCD=Q parameter in the DCB of OS/VS2 on the IBM equipment. This correspondence is also the one used by the LIB$TRA_ASC_EBC and LIB$TRA_EBC_ASC functions on the VAX-11 computers. Unfortunately, there are problems with this correspondence; for example, the EBCDIC exclamation mark does not correspond to the ASCII exclamation mark. Therefore, these programs provide a second correspondence, referred to here as the "extended" correspondence, which does a few thing better. The extended correspondence is compatible with the one used by the WYLBUR Text Editor for ASCII Terminals.

The second problem is caused by a stupid and unjustifiable decision made at SLAC in July of 1982. At that time, SLAC decided to change the internal hexadecimal representation of the braces "{" and "}" to "C0" and "D0" instead of the normal "8B" and "9B". There are many reasons why this change should not have been made, among them are:

1. The problem arose because of an idiosyncrasy in a specific terminal and could have been fixed by changing the translate table for that terminal.
2. When the IBM 360 computer was first announced, much discussion arose over the use of EBCDIC instead of ASCII. IBM argued that EBCDIC was better because it

did not have special characters between lower and upper case letters as ASCII has. This change violated this principal as well as mixing special characters among the upper case letters themselves.

3. Because of the second point, simple coding techniques such as "IF (('A'<=CHAR)&(CHAR<='Z')) ..." formerly were sufficient to identify an upper case letter and this test was valid on both EBCDIC and ASCII computers. This simple technique will no longer work.

Because of these problems a total of eight translation schemes are provided by these programs. These schemes are selected by the TRANSL parameter on the FILE specification statement and are:

1. EBCDIC to ASCII: Both braces are translated according to the standard correspondence.
2. EBCDIC to ASCII: Both braces are translated according to the extended correspondence.
3. ASCII to EBCDIC: The standard correspondence is used and the standard braces are generated.
4. ASCII to EBCDIC: The extended correspondence is used and the standard braces are generated.
5. EBCDIC to EBCDIC: Both braces are translated to the standard braces.
6. EBCDIC to EBCDIC: Both braces are translated to the SLAC braces.
7. ASCII to EBCDIC: The standard correspondence is used and the SLAC braces are generated.
8. ASCII to EBCDIC: The extended correspondence is used and the SLAC braces are generated.

Any of these translation options can be invoked by any of the programs. However, in normal circumstances, only certain ones (1, 2, 5, and 6) make sense for TAPEGENR and others (3, 4, 5, 6, 7, and 8) make sense for TAPEVERF or TAPEUNLD. Also notice that it makes very little sense to generate a labeled tape using an EBCDIC to ASCII translation because one then has ASCII data on a tape with EBCDIC labels.