163 VM update

Spring 2000

In this issue

VM Update

Published by

Software Diversified Services (SDS)
5155 East River Road
Minneapolis, MN 55421-1025
USA
www.sdsusa.com
sales@sdsusa.com
support@sdsusa.com
voice 763-571-9000
fax 763-572-1721

SDS became the publisher of *VM Update* with the January 2000 issue. Prior to that, it was published by Xephon plc.

Editor

Phil Norcross
vu-ed@sdsusa.com
763-571-9000

Editorial Panel

Chuck Meyer, president, Chuck Meyer Systems, Inc.

File formats

VM Update is published in pdf format, to be read with an Adobe® Acrobat® Reader. The Reader is available free of charge at www.adobe.com. Once the Reader is installed, Netscape and Microsoft browsers can display pdf files in browser windows.

Most of the code described in articles is also available in text or other formats that readers can readily copy to their VM machines.

Free subscription, back issues

VM Update is free of charge at www.sdsusa.com. At that site, SDS provides back issues through January 1997. Parts of older issues are available at www.xephon. com/archives/vmi.htm.

Contributions

SDS and *VM Update* welcome contributions. See "Contributing Articles" at www.sdsusa.com/vmupdate/vutoauthors.htm

Disclaimer

Readers are cautioned that, although the information in this journal is presented in good faith, neither SDS nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither SDS nor the contributing organizations or individuals accept any liability of any kind whatsoever arising out of the use of such material.

Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, EXECs, and other contents of this journal before making any use of it.

CMS document storage and retrieval system supporting full-text search

The code described below is available at www.sdsusa.com/vmupdate.htm —editor.

Origin and purpose of the system

Our company needed to store a lot of text documents that in one way or another had a firm relationship with the mainframe and with all kinds of procedures around it. The type and contents of these documents could vary. To mention some of them:

- JCL sources and operational documentation for JCL,
- program sources and documentation,
- administration of computer hardware and software manuals,
- administration of third-party software tapes,
- all kinds of documentation for systems programmers (site procedures), and
- names, addresses, and related information for third-party service people.

During the design phase of the system described here, named HBVM, we decided that the system must:

- be easy to use,
- be very good at full-text searching,
- require minimal maintenance and manual intervention, and
- have as little downtime as possible.

Contents of the HBVM database (how documents are stored physically)

Given the goals and requirements, I created the HBVM CMS userid, which runs as an unattended server. It has run for years already and appears to be very stable and needs nearly no maintenance.

The very base of the system, which I shall refer to as "the database" or "the HBVM database," is a CMS file having a specific layout.

Records have a variable format, with a maximum LRECL of IRL, where IRL is the internal record length. The actual value for IRL can be found in the source of the FINDTEXT program (FINDTEXT ASSEMBLE). The current value is 139.

Each record starts with an internal key with a length of IKL, where IKL is the internal key length. The actual value for IKL can be found in the source of the FINDTEXT program (FINDTEXT ASSEMBLE). The current value is 6. This key field is a zoned numeric field. Of course I realized that the key could have been defined as a binary field; however, in the rare case of a corrupt database, things will be much easier to repair when the contents of the database are readable by humans.

Each document in the database is stored as a series of records having the same numeric internal key. That numeric key corresponds to the relative record number in the database of the first record of the document.

So when a document spans records 1512 through 1520, each of those records starts with a key value of 1512 (possibly preceeded by zeroes, depending on IKL).

The first record stored for each document has a special layout:

- internal key (zoned numeric) with length IKL
- fixed five-character string "#DAT="
- date of creation or update of the document (yyyymmdd)
- fixed five-character string "#STA="

- document status (O=normal, D=logically deleted)
- fixed five-character string “#LBU=“
- lock status, containing either the name of the CMS machine having locked this document, or a string of underscores if the document isn't locked currently by some CMS userid
- fixed five-character string “#DOC=“
- five-character document number, consisting of
  - an uppercase letter
  - three numeric digits
  - an uppercase letter

  This unique number will be assigned by the HBVM system during initial creation of a document. The number to be assigned to the next new document is maintained in the file LASTING GLOBALV.
- fixed five-character string “#UPU=“
- an eight-character name of the CMS userid having either created or last updated the document

Tables to be maintained manually on the server's A-disk (191)

Of the tables below, only the table HBVM AUT needs to be changed in order to match the demands at your site. The other tables should not be changed except for the descriptive records.

Every line in each table must start with a single letter A, B, C, or D, followed by a blank.

A, B, and C identify descriptive records that will be displayed in alphabetic order as the three headers of the table when you ask for help. After the A, B, or C, the rest of the data in the line is comment. If you supply duplicate data for A, B, and C, only the last line will be displayed.

Lines starting with the letter D are considered to be the genuine table data.

The HBVM AUT table contains userids having update access to HBVM. In lines starting with D, the second word is a CMS userid and the rest of line is comment.

The HBVM CMD table contains all valid commands. In lines starting with D, the second word is a valid command and the rest of the line is comment.

The HBVM OPR table contains all the valid comparison operators. In lines starting with D, the second word is a valid comparison operator, and the rest of the line is comment.

The HBVM PFX table contains all valid prefixes (field names). In lines starting with D, the second word is a valid prefix or field name, and the rest of line is comment.

Interaction between HBVM clients and the HBVM server

The HBVM system uses the VM reader for sending documents to and from the HBVM server. At first glance this might seem a negative aspect as far as responsiveness is concerned; however, there is one big advantage: users can add new documents and search documents in the database even when the HBVM server is down.

The server will process all new documents from its reader when it wakes up after a period of "sleep." Having said that, I must admit that this situation has rarely occurred, and most of the time due to a human mistake.

The HBVM client software does some handshaking with the server during startup (HBVMRCOV EXEC), as follows:

Checks the server status by sending SYNC files to it and receiving a SYNC confirmation.

If the server is UP, and

if the user is authorized for UPDATE (AUT table),

it checks whether the CMS userid has any open locks on documents; these locks will be removed automatically.

If the server is DOWN,
it informs that the server is DOWN and tells (UPDATE) users that they can still enter new documents and still search and retrieve documents in the database.

Note: a SYNC file consists of 1 record, having 2 words in it:
1. the word "SYNC," and
2. a unique timestamp.

Prerequisite for the HBVM system

The HBVM system depends on the MAPCOMP software package, published earlier in *VM Update* [January 2000, p. 3] in order to be able to compile the full screen map sources (files with filetype MAP). Each MAP file must be compiled once before the panel can be used.

Set up and initialize the HBVM server

1. Create a new entry in the VM directory for CMS userid HBVM, according to the following specifications:

   Virtual storage: 8 megabytes (this is sufficient for our company).
   CP privilege class G.
   DATEFORMAT FULLDATE.
   Minidisk with address 191:
   We need 15 cylinders of 3390. Try this value and adjust it later on if desired. This minidisk will contain the database and the tables.
   Minidisk with address 192:
   We need 1 cylinder of 3390. This minidisk will contain the software for HBVM.

2. Format the 191 and 192 disks (blocksize 4K is advisable).

3. Log onto the HBVM server.

4. Copy the following files to the 192 disk of HBVM:

FINDTEXT	ASSEMBLE	(search module)
FINDTEXT	RC	(list of possible FINDTEXT return codes)
HBQCPSET	EXEC	(extract information of CP Q SET command)
HBVM	MAP	(initial panel)

HBVMACC	EXEC	(access the HBVM minidisks)
HBVMDISP	EXEC	(display contents of a document)
HBVMGO	MAP	(panel to confirm update / delete)
HBVMHELP	EXEC	(process HELP information)
HBVMHELP	MAP	(HELP panel)
HBVMINIT	EXEC	(first-time initialization of a new database)
HBVMINV	EXEC	(create a new HBVM document)
HBVMLOG	EXEC	(server software)
HBVMPRNT	EXEC	(create prin tfile of HBVM document(s))
HBVMPRNT	MAP	(print panel)
HBVMRCOV	EXEC	(recover clients' locked documents)
HBVMREOR	EXEC	(reorganize HBVM database)
HBVMSELK	EXEC	(show documents found during search)
HBVMSELK	MAP	(search result panel)
HBVMTAB	EXEC	(query HBVM tables)
HBVMVM	EXEC	(allow CMS / CP commands using PF10)
HBVMWIJZ	EXEC	(update HBVM documents)
HBVMXHLP	XEDIT	(show HELP when adding or changing documents)
HBVMZOEK	EXEC	(search documents in the database)
HBVMZOEK	MAP	(document search panel)
HBVM2	EXEC	(initiate client processing)

5. Temporarily LINK to and ACCESS the shared disk which contains the MAPCOMP software.

6. For each file on the 192 disk having a filetype of MAP, issue the command `MAPCOMP <CMS-fileid>`.

7. Copy all files on disk 191 (A) having filetype XEDIT to the 192 disk and erase them from the 191 disk afterwards.

8. Change the minidisk password in HBVMACC EXEC for the CP LINK command.

9. Assemble FINDTEXT ASSEMBLE and put the generated MODULE on the 192 disk.

10. Run exec HBVMINIT in order to create an initial HBVM database and to initialize the file LASTING GLOBALV with a document number.

11. In order to avoid the database being overwritten later, it is wise to rename HBVMINIT EXEC now to some other filetype and at

least give it a filemode number zero, so that users sharing the disk won't see it.

12. Copy the following files to the 191 disk of HBVM:

HBVMSELK	MAP	(search result panel)
HBVM	AUT	(update authorization)
HBVM	CMD	(valid commands in search panel)
HBVM	OPR	(valid comparison operators in search panel)
HBVM	PFX	(valid field names in search panel)
PROFILE	EXEC	(startup profile)

13. Change the target to which the console will be spooled in the PROFILE EXEC to the target userid you wish.

14. Change the list of userids to be authorized for UPDATE in the file HBVM AUT. That means add records in which the first "word" is the letter D and the second word is the CMS userid to be authorized.

15. Initiate either a manual or an automated procedure that regularly, once a week for example, issues the following command:

    ```
    CP SMSG HBVM REORGANIZE
    ```

    When HBVM receives this special message, it will (unattend-edly) reorganize the database. Of course you'd better let HBVM reorganize when nobody is trying to access HBVM from a client.

    Note: Check the definition for TDISK space in HBVMREOR EXEC and alter it if required.

16. Copy the file HBVM EXEC to a disk shared by all HBVM clients. The client starts this exec to invoke HBVM.

17. Change the minidisk password in HBVM EXEC on the LINK statement.

Using the HBVM server from a client userid

Execute the HBVM exec, which should be present on a shared disk.

If you are defined as an UPDATE user (in HBVM AUT on HBVM 191 disk), HBVM will check if there are any documents left still

locked by you (caused by a system crash). If so, they will be unlocked for you now.

On the panels, you can use PF10 in order to invoke a CMS subset and issue CMS commands.

PF3 will always take you one level back.

PF7 and PF8 can be used to page backward and forward (if appropriate).

On the first panel that will come up (HBVM), you can choose between 1) find documents, and 2) add a new document.

Finding documents in the HBVM database

When you choose 1 in panel HBVM, the panel HBVMZOEK is displayed.

In this panel you can issue the following commands:

`RUN` Search the database according to the criteria specified in the map, or according to the criteria stored in a CMS file, in which case the syntax is RUN <name>.

`RUNR` Same as RUN, except that it will ignore any on-screen criteria.

`SAVE` Store the on-screen criteria in a CMS file, which can be referred to later in a RUN or GET command. The criteria file will have the file name which you specify as the only operand to SAVE(R). The filetype always is HBVMSELK. The file will be stored on the A-disk of the client.

`SAVER` Same as SAVE, except that it will overwrite a criteria file in CMS if it did exist already.

`GET` Retrieve the criteria from a CMS file that was stored by an earlier SAVE command.

`GETR` Same as GET, except that screen contents will be overwritten.

How to search the HBVM database? Specify search criteria and issue the command RUN to start a search. Of course you can define

several standard sets of serach criteria using command SAVE(R) and afterwards RUN such a set.

How to specify search criteria? Every criterion consists of a so-called "prefix" (a kind of field name), a comparison operator, and a value to compare the prefix with.

There is one special prefix called TEXT. Use this prefix in order to execute a free full-text search.

If you select prefix "#DAT" you should specify a date in the format yyyymmdd. To select all documents created in May 1999, for example, specify #DAT = 199905. To find documents newer than March 21, 1997, specify #DAT > 19970321 or #DAT >= 19970322.

What can be specified for each of those criteria fields? To obtain HELP or exact information about which input is allowed in the search criteria (and in the rest of the panel), position the cursor in the first position of the screen field you need help about and press PF1. For some "fields" there is a table available to choose a value from; to pick a value, mark the desired entry with X and press PF1 again.

What relationship can be specified among two or more criteria? For OR, specify the character | (vertical bar). For AND, specify the character & (ampersand). The AND relationship has precedence over OR.

Update existing documents in the database

First search the database for the documents you want to change. After a succesfull search the panel HBVMSELK will show up. Fill out the document number found in column "Nr" at the prompt "Your choice" and press PF4.

Modify the document using standard XEDIT commands and FILE it.

Panel HBVMGO will show up in order to facilitate a final confirmation. After answering that with PF4, the updated document will be sent to the server and applied to the database.

Note: To update the same document twice, you have to rerun the query.

Delete existing documents in the database

First search the database for the documents you want to delete. After a succesfull search the panel HBVMSELK will show up. Fill out the document number found in column “Nr” at the prompt “Your choice” and press PF5.

Panel HBVMGO will show up in order to facilitate a final confirmation. After answering that with PF4, the document will be logically deleted in the database by the server. Logically deleted records will be removed physically during a reorganization (HBVMREOR).

Create printfiles of documents in the database

First search the database for the documents you want to print. After a succesfull search the map HBVMSELK will show up. Fill out the document number found in column “Nr” at the prompt “Your choice.”

Note that in this case you may specify * for Nr in order to create a report of all the documents found during the search (not only the documents shown on the current panel, but all of them). After that, press PF6.

Panel HBVMPRNT will appear. In this panel you can specify to which CMS file the report will be written. Also you can specify whether you want full detail or titles only (the titles are the first line of each document).

Furthermore you can choose between a report meant for printing on the system printer (in our case AFP) or on an office printer. In our shop we use CANON printers coupled to coaxial protocol convertors. These convertors are programmed with our own printer escape sequences. Of course this will not be relevant for your shop. However, it won't be very difficult for you to adjust HBVMPRNT to your local needs.

Adding new documents

When you enter choice 2, add a new document, on panel HBVM, an XEDIT session will be started.

In the first line, always type a meaningful short title for your document because this line will appear in the search results panel (HBVMSELK).

The rest of of the file is free text. However, in order to facilitate keyword searches you'd better reserve a special character (e.g. the backslash \ as we do in our shop).

Here, for example is a keyword list using \ as the keyword marker:

```
\ibm\mainframe\dasd\3390\
```

When this information is in the database, you can search for free text 3390 or 390, and also for keyword \3390\ (the trailing slash is optional).

A.P. van Wingerden
Systems Programmer
Binnendams 74
3373 AE Hardinxveld - Giessendam
The Netherlands
a.vanwingerden@pcmuitgevers.nl

PF2PS — a text-to-PostScript translator

The code described below is available at www.sdsusa.com/vmupdate.htm —editor.

PF2PS is a text-to-PostScript translator that takes an ordinary text file such as might be created by VM XEDIT (or, for that matter, by Windows Notepad) and converts it line-by-line into a PostScript document. It is written as a REXX EXEC and requires CMS pipelines.

This program is particularly useful if one has an older laser printer whose PostScript capability is enabled only by powering the printer down, inserting a circuit module, and then powering up, all of which takes considerable time. Then the steps have to be reversed to put the printer back into non-PostScript mode.

A solution is to leave the printer in PostScript mode all the time and instead convert non-PostScript files to PostScript for printing, which is what this program does.

The program is also useful you desire to imbed a piece of ordinary text into an existing PostScript file. If the text is short, this is easily done manually, but otherwise it can be quite tedious.

Directions for use are below and are displayed by the program itself by typing `PF2PS ?`

PF2PS

```
/* V 1.2.  © Copyright 2000, B. E. Chi, bec@nysernet.org

PF2PS fn1 ft1 fm1 fn2 ft2 fm2 [([CC|PAG|NOH][) [n] [L]]] reads a file
<fn1 ft1 fm1> and translates it to PostScript format, writing the result
in <fn2 ft2 fm2>.  If the input file has printer control characters in
column 1 (either ASA or channel commands), use the CC option.  Otherwise
the file will be formatted 55 lines/page with a header on each page,
giving the same appearance as does the standard CMS PRINT (NOCC)
command, unless the PAG option is given, in which case there will be
page numbers but no header, or the NOH option is given, in which case
the file will be formatted 58 lines/page with neither header nor page
numbers.  Options CC, PAG and NOH are mutually exclusive:  if more than
one are specified, all but the first is ignored.
```

```
The parameter n following the options causes n copies of each page to
be printed as it is encountered.  If n be unspecified, n = 1 is assumed.

The flag L following the options causes printing to occur in landscape
mode.  (The font size is reduced to permit the same number of lines/page
as for portrait, with a resulting line width of about 150 characters.)

Defaults:  ft1 = LISTING; fm1 = *; fn2 = fn1; ft2 = PS, fm2 = A1.

Any existing <fn2 ft2 fm2> is overwritten without warning.

*/
/* Configurable parameters (fix up help text above if changed):       */
version = "1.0"                                      /* Base release. */
version = "1.1"      /* Added %%Page:, %%Trailer, %%Pages comments to */
                         /* make Pageview and Ultrix previewer happy. */
version = "1.2"          /* Added P option:  page numbers w/o header. */
sysname = "IBM 9672 R20"                      /* For NOCC header line. */
pw = 8.5                                       /* Page width (inches). */
ph = 11.0                                             /* Page height. */
lm = 0.5                                              /* Left margin. */
tm = 0.8                                               /* Top margin. */
font = "Courier"                     /* Type font.  Must be monopitch. */
fs = 11                                         /* Font size (points). */
ls = 12                                       /* Line spacing (points). */
lpp = 55                                  /* Lines/page for NOCC input. */
ppnc = 71    /* Portrait column where "Page n" starts in NOCC header. */
lpnc = 135  /* Landscape column where "Page n" starts in NOCC header. */

ARG fn1 ft1 fm1 fn2 ft2 fm2 "("options")" flags
IF fn1 = "" | fn1 = "?" THEN DO
  PARSE SOURCE . . fn ft fm .
  "PIPE <" fn ft fm "| FRLAB  | TOLAB */| CONSOLE"
  EXIT 0
END

IF ft1 = "" THEN ft1 = "LISTING"
IF fm1 = "" THEN fm1 = "*"

IF nc = "" THEN nc = 1              /* Number of (decollated) copies. */

"PIPE CMS LISTFILE" fn1 ft1 fm1 "(NOH ALLOC | VAR X"
IF RC ^= 0 THEN DO                       /* See if input file exists, */
  SAY x
  EXIT RC                   /* and if it does, get its mode and size. */
END
ELSE PARSE VAR x . . fm1 . lrecl recs .

IF fn2 = "" THEN fn2 = fn1
```

```
IF ft2 = "" THEN ft2 = "PS"
IF fm2 = "" THEN fm2 = "A1"

IF lrecl*recs > 65535 THEN long = "LONG"
ELSE long = ""

pw = TRUNC(72*pw+0.5); ph = TRUNC(72*ph+0.5)          /* Convert inch */
lm = TRUNC(72*lm+0.5); tm = TRUNC(72*tm+0.5)      /* specs to points. */
bbw = pw; bbh = ph                                 /* For Bounding Box. */

PARSE SOURCE . . source .                  /* For "%%CREATOR" comment. */

nc = 1                       /* Number of (decollated) copies to print. */
rot = ""                                      /* Assume portrait mode. */
pnc = ppnc
DO WHILE flags ^= ""
  PARSE VAR flags x flags
  IF x = "L" THEN DO                           /* Set up for landscape */
    scale = TRUNC(pw/ph,3)
    rot = "90 rotate 0 -"pw" translate" scale scale "scale "
    pnc = lpnc
    bbw = ph; bbh = pw                              /* For Bounding Box. */
  END
  ELSE IF DATATYPE(x,"W") THEN nc = x               /* Number of copies. */
END

PARSE VALUE STRIP(options) WITH option 2                 /* C, N or P. */
IF option = "C" THEN preprocess = "MCTOASA"
                      /* Insure file contains ASA control characters. */

ELSE IF option = "N" THEN                    /* File must be formatted. */
  preprocess =,                  /* (Neither headers nor page numbers.) */
    "JOIN" lpp+2 "STRING /"'3F'X"/",       /* Concatenate lpp+3 lines, */
    "| SPEC /1/ 1 1-* 2",     /* precede first line with newpage code, */
    "| SPLIT BEFORE 3F",                    /* split lines back apart, */
    "| CHANGE /"'3F'X"/ /"   /* preceding each except 1st with blank. */

ELSE DO                         /* File must be formatted with headers. */
  IF option = "P" THEN hdr = ""                  /* Just page numbers. */
  ELSE DO                                  /* Headers and page numbers. */
    PARSE VALUE DIAG(8,"QUERY CPLEVEL") WITH "/"a . . b","
    hdr = "FILE:",                                /* Page header text. */
      LEFT(fn1,9)LEFT(ft1,9)LEFT(fm1,3)sysname a b
  END
  preprocess =,
    "JOIN" lpp-1 "STRING /"'3F'X"/",         /* Concatenate lpp lines, */
    "| SPEC /1"hdr"/ 1",        /* precede with header or page number, */
      "/Page/" pnc+1 "RECNO" pnc+6 "LEFT X3E NEXT 1-* NEXT",
    "| SPLIT BEFORE 3E-3F",                 /* split lines back apart, */
```

```
    "| CHANGE /"'3F'X"/ /",  /* preceding each except hdr with blank, */
    "| CHANGE /"'3E'X"/-/"  /* but triple-space before 1st text line. */
END

s.1 = "%!PS-Adobe-2.0"      /* Begin standard PostScript boilerplate. */
s.2 = "%%Title:" fn2 ft2 fm2
s.3 = "%%Source:" fn1 ft1 fm1
s.4 = "%%Creator:" source version
s.5 = "%%CreationDate:" DATE("U") TIME()
s.6 = "%%BoundingBox: 0 0" bbw bbh
s.7 = "%%EndComments"
s.8 = "gsave 0 setgray"
s.9 = "/#copies" nc "def"
s.10 = "/"font "findfont" fs "scalefont setfont"
s.11 = "/cr {"lm "vp moveto} def"  /* "Carriage return" w/o linefeed. */
s.12 = "/nl {/vp vp" ls "sub def cr} def"               /* New line. */
s.13 = "/np {"rot"/vp" ph-tm "def cr} def"              /* New page. */
s.14 = "%%Page: 1 1"
s.15 = "np"
s.0 = 15

bigpipe =,
  "PIPE ("long "END !)",
  "  STEM s. ",                          /* First write boilerplate. */
  "| a: FANIN ",                 /* Modified input records come here. */
  "| JOIN * /"'3C'X"/",     /* Combine all records with 3C separator. */
  "| SPLIT AT 3D",                      /* Split after each showpage. */
  "| SPEC RECNO 1 LEFT 1-* 11",               /* Insert %%Page: n n. */
  "| SPEC /%%Page:/ 1 W1 NEXTW W1 NEXTW /"'3C'X"/ NEXTW 11-* NEXT",
  "| SPLIT AT 3C",                        /* Split records back apart. */
  "| DROP 1",                       /* (Spurious %%Page: 1 1 at top.) */
  "| d: FANOUT",                 /* Make 2nd copy for counting pages. */
  "| e: FANIN",               /* All records come here to be written. */
  "| >" fn2 ft2 fm2; bigpipe = bigpipe||,
  "! <" fn1 ft1 fm1,                                /* Read input file, */
  "| XLATE *-* 00-3F BC",  /* change all non-printing codes to inv ?, */
  "|" preprocess,              /* and insure that it's in ASA format. */
  "| b: TAKE 1 ",
  "| CHANGE 1-1 /1/+/",           /* Remove any formfeed from line 1. */
  "| c: FANIN",                                    /* For each line, */
  "| CHANGE /\/\\/",  /* precede any \, (, ) with escape character \. */
  "| CHANGE /(/\(/",
  "| CHANGE /)/\)/",
  "| CHANGE 1-1 /+/cr (/",         /* Replace ASA + with cr operator, */
  "| CHANGE 1-1 / /nl (/",                 /* blank with nl operator, */
  "| CHANGE 1-1 /0/nl nl (/",             /* 0 with two nl operators, */
  "| CHANGE 1-1 /-/nl nl nl (/",                                /* etc. */
  "| CHANGE 1-1 /1/showpage"'3D'X"np (/"; bigpipe = bigpipe||,
  "| STRIP TRAILING",                     /* Discard trailing blanks. */
```

```
  "| SPEC 1-* 1 /) show/ NEXT",  /* Encl text in (), append show opr. */
  "| CHANGE 1-10 /nl () show/nl/",/* Eliminate shows for empty lines. */
  "| a:",                       /* Write modified line to output file. */
  "! b:",                    /* Second and subsequent lines come here. */
  "| c:",                             /* Edit and write as for line 1. */
  "! LITERAL showpage"'3C'X"/#copies 1 def"'3C'X"grestore",
  "| a:",     /* Append these commands to the end of the output file. */
  "! d:",                     /* 2nd copy of modified file comes here. */
  "| FIND %%Page:"||,                                  /* Count pages. */
  "| COUNT LINES",
  "| SPEC /%%Pages: / 1 1-* NEXT",    /* %%Pages n comment to output. */
  "| LITERAL %%Trailer",
  "| e:"

bigpipe

EXIT RC
```

B. E. Chi
NYSERNet, Inc., Troy, New York
bec@nysernet.org

A full screen console interface—part 20

Here is the final installment of an article VM Update *has been publishing in pieces since August 1998. Text files that provide code from all 20 installments are available at www.sdsusa.com/vmupdate.htm —editor.*

REMOTE HELPCSCC

```
.cm VM Software Services
.cm
.cs Ø on
 (c) Copyright CSC Inc, 1997

                                []CSC Tool[%
[%
[%
 Use the REMOTE statement to identify the CSC remote nodes.

   EXAMPLE: REMOTE VM2 RES2

            Define APPC/VM resource RES2 for remote node VM2.

.cs Ø off
.cs 1 on
 (c) Copyright CSC Inc, 1997

                                []CSC Tool[%
[]Purpose[%

 Use the REMOTE statement to identify the CSC remote nodes.
.cs 1 off
.cs 2 on

[]Format[%

 >>-REmote-nodeid-resourceid----------------------><

.cs 2 off
.cs 3 on
[]Operands[%

 nodeid
     name to identify to remote node. Must be from 1 to 8 characters.
```

```
 resourceid
     name for the resource associated with this node. Must be a valid
     APPC/VM resource name.

.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
.cs 5 off
.cs 6 on
[]Messages[%

 Ø050E  Missing REMOTE operand(s). Statement discarded
 Ø052E  Unexpected REMOTE operand: operand. Statement discarded
 Ø053E  REMOTE operand “operand...” is too long. Statement discarded
 Ø071E  Node name name is not unique. Statement discarded
 Ø072E  Resource name name is not unique. Statement discarded
.cs 6 off
```

ROUTE HELPCSCC

```
.cm VM Software Services
.cm
.cs Ø on
 (c) Copyright CSC Inc, 1997

                                []CSC Tool[%
[%
[%
 Use the ROUTE statement to define a list of users to receive a message.

   EXAMPLE: ROUTE ABC VM1 USER1 VM1 USER2

            Create route ABC with users USER1 at VM1 and USER2 at VM1.

.cs Ø off
.cs 1 on
 (c) Copyright CSC Inc, 1997

                                []CSC Tool[%
[]Purpose[%

 Use the ROUTE statement to define a list of users to receive a message.
.cs 1 off
.cs 2 on

[]Format[%

                            .<-------<.
 >>-.-Route-.-routename-'-nodeid-userid-'----------------><
```

```
     '-RTE-'

.cs 2 off
.cs 3 on
[]Operands[%

 routename
     route name. Must be from 1 to 8 characters.

 nodeid
     VM nodeid for user to be defined next.

 userid
     user to receive the message.

.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%

   1.   Lists can be extended by coding multiple ROUTE statements with
        the same route name.

   2.   RSCS is used to send a message to a remote VM node.

   3.   RTE is a synonym for ROUTE.

.cs 5 off
.cs 6 on
[]Messages[%

 Ø050E  Missing ROUTE operand(s). Statement discarded
 Ø053E  ROUTE operand “operand...” is too long. Statement discarded
 Ø100E  Node without Userid found on ROUTE statement. Discarded
.cs 6 off
```

TITLE HELPCSCC

```
.cm VM Software Services
.cm
.cs Ø on
 (c) Copyright CSC Inc, 1997

                                []CSC Tool[%
[%
[%
 Use the TITLE statement to change the default Title line on user
sessions.
```

```
   EXAMPLE: TITLE New Title

            Change Title line to “New Title”.

.cs Ø off
.cs 1 on
 (c) Copyright CSC Inc, 1997

                                  []CSC Tool[%
[]Purpose[%

 Use the TITLE statement to change the default Title line on user
sessions.
.cs 1 off
.cs 2 on

[]Format[%

 >>-.-Title-.-.---.-------------------------><
     '-TTL-'  '-title-'

.cs 2 off
.cs 3 on
[]Operands[%

 title
     new title. If omitted no title is displayed. Maximum is 3Ø
characters.

.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%

   1.   TTL is a synonym for TITLE.

.cs 5 off
.cs 6 on
[]Messages[%

 Ø11ØW  Title title... too long. Truncated
.cs 6 off
```

USER HELPCSCC

```
.cm VM Software Services
.cm
.cs Ø on
 (c) Copyright CSC Inc, 1997
```

```
                                []CSC Tool[%
[%
[%
 Use the USER statement to authorize users to access the CSC Service
Machine.

   EXAMPLE: USER ABC CLASSES 1 2 3

            Define user ABC with access classes 1, 2 and 3.

.cs Ø off
.cs 1 on
 (c) Copyright CSC Inc, 1997

                                []CSC Tool[%
[]Purpose[%

 Use the USER statement to authorize users to access the CSC Service
Machine.
.cs 1 off
.cs 2 on

[]Format[%

 >>-.-User-.-.-userid-.-.----------.-------------><
     '-USR-'  '-*---'  ]          .<--<. ]
                           '-Classes-+---+-'
                                     '-class-'

.cs 2 off
.cs 3 on
[]Operands[%

 userid
     user to authorize. Enter an asterisk (*) to define a universal
access.

 class
     classes the user is authorized to access. Must be in the range 1-
32.

.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%

   1.   Specifying a class twice has no effect.
```

```
   2.   USR is a synonym for USER.

.cs 5 off
.cs 6 on
[]Messages[%

 Ø05ØE  Missing USER operand(s). Statement discarded
 Ø051E  Invalid USER operand: operand. Statement discarded
 Ø053E  USER operand "operand..." is too long. Statement discarded
 Ø12ØE  Non numeric USER class: class. Ignored
 Ø121E  USER class class not in the range Ø1-32. Ignored
.cs 6 off
```

CSCSVP HELPMENU

```
.cm VM Software Services
.cm
(c) Copyright CSC Inc, 1997

                                []CSC Tool[%

A file may be selected for viewing by placing the cursor under any
character of the file wanted and pressing the ENTER key or the PF1 key.
A MENU file is indicated when a name is preceded by an asterisk (*).
A TASK file is indicated when a name is preceded by a colon  (:).
For a description of the HELP operands and options, type HELP HELP.

*Query_S
CMS
END
Query
```

CSCSVP HELPABBR

```
QUERY     Query        1
```

CMS HELPCSCS

```
.cm VM Software Services
.cm
.cs Ø on
 (c) Copyright CSC Inc, 1997

                                 []CSC Tool[%
[%
[%
 Use the CMS command to execute any CMS command.

   EXAMPLE: CMS QUERY DISK
```

```
            Execute CMS command QUERY DISK.

.cs Ø off
.cs 1 on
 (c) Copyright CSC Inc, 1997

                                  []CSC Tool[%
[]Purpose[%

 Use the CMS command to execute any CMS command.
.cs 1 off
.cs 2 on

[]Format[%

 >>-CMS-command------------------------------><

.cs 2 off
.cs 3 on
[]Operands[%

 command
     any CMS command.

.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%

   1.   CSC does not collect messages or process users while the CMS
        command is running.

.cs 5 off
.cs 6 on
[]Messages[%

 Ø61ØE  CMS command is missing
 Ø611I  CMS command “command” ended with return code code
.cs 6 off
```

END HELPCSCS

```
.cm VM Software Services
.cm
.cs Ø on
 (c) Copyright CSC Inc, 1997

                                  []CSC Tool[%
[%
```

```
[%
 Use the END command to teminate CSCSVP on the CSC Service Machine.

   EXAMPLE: END

            Terminate CSCSVP.

.cs Ø off
.cs 1 on
 (c) Copyright CSC Inc, 1997

                                  []CSC Tool[%
[]Purpose[%

 Use the END command to teminate CSCSVP on the CSC Service Machine.
.cs 1 off
.cs 2 on

[]Format[%

 >>-.-END---.------------------------------><
     ]-BYE---]
     ]-EXIT---]
     ]-GOBACK--]
     ]-QUIT---]
     '-TERMINATE-'

.cs 2 off
.cs 3 on
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%

   1.   All sessions and links are normally terminated if possible.

   2.   BYE, EXIT, GOBACK, QUIT and TERMINATE are valid synonyms for
        END.
        Others should be added soon.

.cs 5 off
.cs 6 on
[]Messages and Return Codes[%

 Ø6Ø5E  Unexpected END operand: operand
.cs 6 off
```

QUERY HELPCSCS

```
.cm VM Software Services
.cm
.mt CSCQ
(c) Copyright CSC Inc, 1997

                                []CSC Tool[%

A file may be selected for viewing by placing the cursor under any
character of the file wanted and pressing the ENTER key or the PF1 key.
A MENU file is indicated when a name is preceded by an asterisk (*).
A TASK file is indicated when a name is preceded by a colon  (:).
For a description of the HELP operands and options, type HELP HELP.

Links
Storage
```

QUERY_S HELPMENU

```
.cm VM Software Services
.cm
.mt CSCQ
(c) Copyright CSC Inc, 1997

                                []CSC Tool[%

A file may be selected for viewing by placing the cursor under any
character of the file wanted and pressing the ENTER key or the PF1 key.
A MENU file is indicated when a name is preceded by an asterisk (*).
A TASK file is indicated when a name is preceded by a colon  (:).
For a description of the HELP operands and options, type HELP HELP.


Links
Storage
```

LINKS HELPCSCQ

```
.cm VM Software Services
.cm
.cs Ø on
 (c) Copyright CSC Inc, 1997

                                 []CSC Tool[%
[%
[%
 Use the QUERY LINKS command to list all defined APPC/VM links.

   EXAMPLE: QUERY LINKS
```

```
            Display all defined APPC/VM links.

.cs Ø off
.cs 1 on
 (c) Copyright CSC Inc, 1997

                                  []CSC Tool[%
[]Purpose[%

 Use the QUERY LINKS command to list all defined APPC/VM links.
.cs 1 off
.cs 2 on

[]Format[%

 >>-Query-Links------------------------------><

.cs 2 off
.cs 3 on
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
.cs 5 off
.cs 6 on
[]Responses[%

 Ø643I  APPC/VM links defined
 Ø644I     Local    name       resource   status     type
 Ø645I     Remote   name       resource   status

 Where:

 Local
 Remote
     identifies the type of node. It can be Local or Remote.

 name
     is the name of the node.

 name
     is the APPC/VM resource associated with this node.

 status
     is the current status of the link. Valid values are Active,
     Inactive, or Pending.

 type
```

```
     is the of APPC/VM resource. Valid values are Local or Global.

[]Messages[%

 Ø6Ø5E  Unexpected QUERY operand: operand
 Ø642E  APPC/VM support is not enabled
.cs 6 off
```

STORAGE HELPCSCQ

```
.cm VM Software Services
.cm
.cs Ø on
 (c) Copyright CSC Inc, 1997

                                []CSC Tool[%
[%
[%
 Use the QUERY STORAGE command to display a storage allocation summary.

   EXAMPLE: QUERY STORAGE

            Display allocation storage summary.

.cs Ø off
.cs 1 on
 (c) Copyright CSC Inc, 1997

                                []CSC Tool[%
[]Purpose[%

 Use the QUERY STORAGE command to display a storage allocation summary.
.cs 1 off
.cs 2 on

[]Format[%

 >>-Query-Storage----------------------------><

.cs 2 off
.cs 3 on
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%

   1.   The storage balance show how many bytes are presently allocated
        after CSC initialization completed.
```

```
.cs 5 off
.cs 6 on
[]Responses[%

 Ø64ØI  n1 bytes allocated in n2 allocations
 Ø641I  Balance is n3 bytes in n4 allocations

 Where:

 n1
     bytes allocated.

 n2
     number of allocations.

 n3
     bytes allocated after CSC initialization ended.

 n4
     allocations performed after initialization ended.

[]Messages[%

 Ø6Ø5E  Unexpected QUERY operand: operand
.cs 6 off
```

NEWS HELPMENU

```
.cm VM Software Services
.cm
.mt CSC
(c) Copyright CSC Inc, 1997

                                []CSC Tool[%

List of new additions to[]CHECK[%. (CSC Help Effort Construction Kit)

A file may be selected for viewing by placing the cursor under any
character of the file wanted and pressing the ENTER key or the PF1 key.
A MENU file is indicated when a name is preceded by an asterisk (*).
A TASK file is indicated when a name is preceded by a colon  (:).
For a description of the HELP operands and options, type HELP HELP.


CSCSVP
CSCUSR
```

CSC_SVP HELPCSC

```
.cm VM Software Services
.cm
 (c) Copyright CSC Inc, 1997

                                []CSC Tool[%

Service program.

  Configuration statements: (minimum abbreviation in capitals)

    DFRecs  n
    DFSize  same as DFRECS
    Message User userid   Alarm   Exit exit Hold Name name Nodisplay
            Release relname Route rname Unique Locate mask
            HIgh BLInk REvvideo UNderline Blue RED Pink Green Turquoise
            Yellow White
    MSG     same as MESSAGE
    Options MSG ] MSGNOH
    Prefix  l Class c
    PFX     same as PREFIX
    Route   rname node1 userid1 node2 userid2 node3 userid3...
    RTE     same as ROUTE
    User    userid Classes c1 c2 c3
    USR     same as USER
```

CSC_USR HELPCSC

```
.cm VM Software Services
.cm
 (c) Copyright CSC Inc, 1997

                                []CSC Tool[%

User program.

  Defined PF keys:

    PFØ1 / PF13   -   Hlp. Help (local command)
    PFØ3 / PF15   -   End. Terminate session (local command)
    PFØ4 / PF16   -   Top. Display top of file. Oldest records
    PFØ5 / PF17   -   Bot. Display Bottom of file. Newest records
    PFØ6 / PF18   -   Rep. Repeat last command. Not reliable.
    PFØ7 / PF19   -   Bwd. Scroll backward one screen. Top line becomes
                          new bottom line
    PFØ8 / PF2Ø   -   Fwd. Scroll forward one screen. Bottom line
                          becomes new top line
    PFØ9 / PF21   -   Cur. Display current screen. NoDisplay records are
```

```
                          not shown. Screen is automatically refreshed
  PF10 / PF22   -   Shf. Shift right/left 64 columns
  PF11 / PF23   -   Rtf. Retrieve forward last commands. Not reliable
  PF12 / PF24   -   Rtb. Retrieve backward last commands. Not reliable

Defined commands: (minimum abbreviation in capitals)

  Backward n   - Scroll backward n screens
  BWD n        - Same as Backward
  BOTtom       - Display bottom of file. Newest records
  Clear        - Clear screen
  COnnect node - Connect to another CSC node
  CMS command  - Execute CMS command (local command)
  CURrent      - Display current screen and enter "refresh" mode
  DIsconnect   - Disconnect from a connected CSC node
  Down n       - Move n lines toward the bottom of the log file
  END          - Terminate session. This is not a local command
  Exclude abc  - Exclude some prefixes from display
  Forward n    - Scroll forward n screens
  FWD n        - Same as forward
  Locate /xxx/ - Scan log file search for string /xxx/. Scan is done
                 backwards. From bottom to top of log file
  /xxx/        - Same as locate. String delimiter must be "/"
  DOWNLocate   - Similar to Locate. Search is done from top to end of
                 log file
  DLocate      - Same as DOWNLocate
  Match *mask* - Scan log file search for pattern *mask*. Same rules
                 as for CMS LISTFILE generic names.
  DOWNMatch    - Similar to Match. Search is done from top to end of
                 log file
  DMatch       - Same as DOWNMtach
  Go date time - Locate record by date and time. Faster than Locate
  Include abc  - Include some prefixes to display screen
  Next n       - Same as Down
  OP p commd   - Execute "commd" as if entered on console defined by
                 prefix "p"
  Print n      - Print n records from Data file
  Release m n  - Release messages on hold from m to n. Default is
                 Release 1 1
  Repeat       - Repeat last entered command. Not reliable
  =            - Same as repeat
  SEt          - Set CSC options. Not available yet
  SHift Left n - Shifts data left or right n columns
       Right n
  Swap n1 n2   - Switch on and off the following fields and flags
                   Date   Show date
                   Time   Show time
                   User   Show user
                   Filter Filter NoDisplay records in browse mode
```

```
                    Cms    Scroll data the "CMS" way
                    Wrap   Not coded
    Switch       - Same as Swap
    Top          - Display top screen. Oldest reords
    Up n         _ Move screen up n lines
    Write n      _ Write n lines from Data file to PrintLog file
```

Maintenance

Use the CSC exec to re-assemble the required modules after a source modification. This exec calls another exec GM that reads a control file to decide what needs to be done.

CSC EXEC

```
/* */
'EXEC GM CSC'
Exit rc
```

GM EXEC

```
/*---------------------------------*/
/*                                                                 */
/*   Name    - GM                                                  */
/*                                                                 */
/*   Purpose - To control generation of multi-phase objects.       */
/*                                                                 */
/*   Entry conditions    -                                         */
/*                                                                 */
/*   Exit conditions     -                                         */
/*                                                                 */
/*   Attributes          - REXX                                    */
/*                                                                 */
/*   External references -                                         */
/*                                                                 */
/*---------------------------------*/
/* yy/mm/dd <...> <.............................................>  */
/* 93/11/29       Created.                                       */

parse arg parms
address command
trace o

call init                              /* Initialization          */

'EXECIO * DISKR' file_id '(STEM RECORD.'

do i = 1 to record.Ø                   /* Process control file    */
```

```
   call scan                           /* Get next statement         */
   select                              /* Check statement code       */
      when code = ' ' then nop
      when code = '*' then nop
      when code = '>' then call object
      when code = ':' then call element
      when code = '-' then call command
      otherwise call error 10
   end
end
drop record.                           /* Clean up variables         */

if error then                          /* Good control file?         */
   call error 11
else
   call process                        /* Yes process...             */

exit
                                       /* ------------- */
                                       /* Initialize variables, ...  */
init:
   parse var parms fn ft fm in_comm in_rest '(' in_opts

   no_object  = 0                      /* Allow Object statements    */
   no_element = 1                      /* ... and nothing else       */
   no_command = 1
   cnt. = 0                            /* Zero statement counters    */
   elm.@txt.0 = 0                      /* TEXT Global Elements       */

   opts    = 0                         /* Options control            */
   opt_all = 0
   opt_bsc = 0
   opt_exe = 0
   opt_sce = 0
   opt_to  =                           /* TO and FROM options        */
   opt_frm =

   error = 0                           /* Logical variables          */
   maxrc = 0
   generate = 0

   if in_rest ¬= '' then               /* Check input parameters     */
      do
         call error 20
         exit 8
      end

   select
      when in_comm = ''                          then gm_comm = 'BD'
```

```
   when abbrev('BUILD', translate(in_comm), 1) then gm_comm = 'BD'
   when abbrev('CHECK', translate(in_comm), 2) then gm_comm = 'CK'
   when abbrev('COPY',  translate(in_comm), 1) then gm_comm = 'CP'
   otherwise
      do
         call error 21              /* Invalid command           */
         exit 8
      end
end

do i = 1 to words(in_opts) until error
   opts = opts + 1
   key = word(in_opts, i)
   select
      when abbrev('ALL',    translate(key), 1) then opt_all = 1
      when abbrev('BASIC',  translate(key), 1) then opt_bsc = 1
      when abbrev('EXEC',   translate(key), 1) then opt_exe = 1
      when abbrev('SOURCE', translate(key), 1) then opt_sce = 1
      when abbrev('TO',     translate(key), 1) ],
           abbrev('FROM',   translate(key), 1) then
         do
            opts = opts - 1         /* Do not count TO and FROM   */
            i = i + 1               /* Extract value              */
            val = translate(word(in_opts, i))
            'VALIDATE * *' val      /* Is it a valid mode?        */
            if rc > Ø then error = 1
            if abbrev('TO', translate(key), 1) then
               opt_to = val
            else
               opt_frm = val
         end
      otherwise error = 1
   end
end

if error then
   do
      call error 22
      exit 8
   end

select                              /* Validate command/option    */
   when gm_comm = 'BD' then
      if in_opts ¬= '' & ¬opt_all then call error 22
   when gm_comm = 'CK' then if in_opts ¬= '' then call error 22
   when gm_comm = 'CP' then if opt_to = '' then call error 23
   otherwise nop
end
```

```
   if opts > 1 then call error 24

   if error then exit 8

   if ft = '' then ft = '$$CNTL$$'     /* Default file type          */
   if fn = '' then
      do
         call error 25                 /* We must have at least ...  */
         exit 8                        /*    ... the file name       */
      end
   file_id = translate(fn ft fm)       /* Uppercase file Id          */
   'VALIDATE' file_id
   if rc ¬= 0 then
      do
         call error 26                 /* Invalid file Id            */
         exit 8
      end
   'ESTATE' file_id
   if rc ¬= 0 then
      do
         call error 27                 /* Control file not found     */
         exit 8
      end

   'FINIS' file_id                     /* Close control file         */
   return
                                       /* ------------- */
                                       /* Scan control file          */
scan:
   next = pos(';', record.i)           /* Check for separator        */
   if next = 0 ] word(record.i, 1) = '*' then
      line = record.i
   else                                /* Split statements           */
      do
         line = substr(record.i, 1, next - 1)
         record.i = substr(record.i, next + 1)
         i = i - 1
      end
   code = word(line, 1)                /* Check statement code       */
   select
      when code = ' ' then nop         /* Ignore blanks and comments */
      when code = '*' then nop
      when code = '>' then
         do                            /* Parse statement            */
            tmp = pos('/*', line)
            if tmp > 0 then if pos('(', line) = 0 then
               line = insert('(', line, tmp - 1)
            tmp = pos('(', line)
            if tmp > 0 then if pos(':', line) = 0 then
```

```
               line = insert(':', line, tmp - 1)
            parse var line code type obj_name obj_rest ':' elm_name,
                  elm_type elm_rest '(' options '/*' comments
            if options ¬= '' then options = '('options
         end
      when code = ':' then
         do
            tmp = pos('/*', line)
            if tmp > Ø then if pos('(', line) = Ø then
               line = insert('(', line, tmp - 1)
            tmp = pos('(', line)
            parse var line code elm_name elm_type elm_rest,
                  '(' options '/*' comments
            if options ¬= '' then options = '('options
         end
      when code = '-' then
         do
            parse var line code command '/*' comments
            command = strip(command)
         end
      otherwise nop
   end
   return

                                    /* ------------- */
object:                             /* Check Object statements    */
   if no_object then call error 3Ø  /* Is statement allowed?      */
   no_object  = 1                   /* Disallow object statements */
   no_command = Ø                   /* Allow all others           */
   no_element = Ø
   select                           /* Check TYPE                 */
      when type = 'MLB' then call check_obj '@MLB MACLIB'
      when type = 'LLB' then call check_obj '@LLB LOADLIB'
      when type = 'TLB' then call check_obj '@TLB TXTLIB'
      when type = 'MOD' then call check_obj '@MOD MODULE'
      when type = 'TXT' then
         if obj_name = '' then      /* No name, element appended  */
            do
               obj_name = elm_name
               call check_obj '@TXT TEXT'
               call check_elm_txt
               no_object = Ø        /* Allow new Objects and ...  */
               no_command = 1       /*    disallow Commands       */
            end
         else
            if obj_name ¬= '*' then /* Global requirements        */
               call check_obj '@TXT TEXT'
            else
               no_command = 1
```

```
      when type = 'GLB' then call check_obj_glb
      when type = 'AUX' then call check_obj_aux '@AUX'
      when type = 'DOC' then call check_obj_aux '@DOC'
      otherwise call error 31
   end
   return

                                     /* ------------- */
check_obj:                           /* Check Object statement     */
   parse arg vid objtype

   if obj_name = '' then call error 4Ø /* Name is missing           */
   if obj_rest ¬= '' then call error 41
   if type ¬= 'TXT' & (elm_name ¬= '' ] options ¬= '') then
      call error 41

   tmp = cnt.vid + 1                 /* Update number of items     */
   cnt.vid = tmp
   gen.vid.tmp = Ø                   /* Generate control           */
   cmd.vid.tmp = Ø                   /* Total of commands          */
   elm.vid.tmp = Ø                   /* Total of elements          */
   nme.vid.tmp = obj_name            /* Name of object             */
   opt.vid.tmp = options             /* Options                    */
   'ESTATE' obj_name objtype '*'
   if rc = Ø then                    /* We have one, check date    */
      do
         'MAKEBUF'
         'LISTFILE' obj_name objtype '* (DATE FIFO'
         parse pull . . . . . . . date time .
         'DROPBUF'
         date = right(date, 8, 'Ø')
         time = right(time, 8, 'Ø')
         date = translate('781245', date, '12345678')
         dte.vid.tmp = date ]] time
      end
   else
      gen.vid.tmp = 1                /* Not found, create it       */
   return

                                     /* ------------- */
check_obj_glb:                       /* Check GLB statements       */
   no_command = 1                    /* Disallow Commands          */

   if obj_rest ¬= '' ] elm_name ¬= '' ] options ¬= '' then
      call error 41

   select
      when obj_name = 'MACLIB' then vid = '@GLM'
      when obj_name = 'TXTLIB' then vid = '@GLT'
```

```
      otherwise
         do                            /* If not MAC or TXT is ...   */
            vid = '@INV'               /* ... invalid                */
            call error 42
         end
   end

   tmp = cnt.vid + 1                   /* Count GLBs within type     */
   cnt.vid = tmp
   elm.vid.tmp =                       /* Clear elements             */
   if cnt.vid > 1 then call error 43   /* Error if duplicate         */
   return

                                       /* ------------- */
check_obj_aux:                         /* Check AUX statements       */
   parse arg vid
   no_command = 1                      /* Disallow Commands          */

   if obj_name ¬= '' ] obj_rest ¬= '' ] elm_name ¬= '' ],
                        options ¬= '' then call error 41

   tmp = cnt.vid + 1
   cnt.vid = tmp
   elm.vid = Ø
   if tmp > 1 then call error 44
   return

                                       /* ------------- */
element:                               /* Process Element statements */
   if no_element then call error 5Ø    /* Is statement allowed?      */
   no_object = Ø                       /* Allow Object statements    */
   select
      when type = 'MLB' then call check_elm '@MLB MACRO COPY'
      when type = 'LLB' then call check_elm '@LLB TEXT'
      when type = 'TLB' then call check_elm '@TLB TEXT'
      when type = 'MOD' then call check_elm '@MOD TEXT'
      when type = 'GLB' then call check_elm_glb
      when type = 'TXT' then call check_elm_txt
      when type = 'AUX' then call check_elm_aux '@AUX'
      when type = 'DOC' then call check_elm_aux '@DOC'
      otherwise call error 51
   end
   return

                                       /* ------------- */
check_elm:                             /* Check Element statements   */
   parse arg vid elmtype
   if elm_type = '' then call error 6Ø /* We must have name and type */
   if elm_rest ¬= '' then call error 61
```

```
   if find(elmtype, elm_type) = Ø then call error 62

   m = cnt.vid                          /* Register Element           */
   n = elm.vid.m + 1
   elm.vid.m = n
   elm_gen.vid.m.n = Ø
   elm_nme.vid.m.n = elm_name
   elm_typ.vid.m.n = elm_type
   'ESTATE' elm_name elm_type '*'       /* Verify Element             */
   if rc = Ø then
      do
         'MAKEBUF'
         'LISTFILE' elm_name elm_type '* (DATE FIFO'
         parse pull . . . . . . . date time .
         'DROPBUF'
         date = right(date, 8, 'Ø')
         time = right(time, 8, 'Ø')
         date = translate('781245', date, '12345678') ]] time
         if date > dte.vid.m then
            do
               gen.vid.m = 1          /* Create Object and Element  */
               elm_gen.vid.m.n = 1
            end
      end
   else
      call check_elm_missing            /* Element not found          */
   return

                                        /* ------------- */
check_elm_glb:                          /* Check GLB details          */
   if elm_name = '' then call error 6Ø

   select
      when obj_name = 'MACLIB' then vid = '@GLM'
      when obj_name = 'TXTLIB' then vid = '@GLT'
      otherwise vid = '@INV'
   end

   if options ¬= '' then call error 63
   tmp = cnt.vid
   elm.vid.tmp = elm.vid.tmp elm_name elm_type elm_rest
   return

                                        /* ------------- */
check_elm_txt:                          /* Check TEXT Elements        */
   if elm_type = '' then call error 6Ø /* We must have name and type */
   if elm_rest ¬= '' then call error 61
   if find('MACRO COPY ASSEMBLE', elm_type) = Ø then call error 62
```

```
   vid = '@TXT'
   m = cnt.vid
   if obj_name = '*' then m = Ø        /* This is a global element   */

   n = elm.vid.m + 1
   elm.vid.m = n
   elm_nme.vid.m.n = elm_name
   elm_typ.vid.m.n = elm_type
   if elm_type = 'ASSEMBLE' then
      do
         if n > 1 then call error 64
         if n = 1 & obj_name ¬= elm_name then call error 65
         if n < 1 then call error 66
      end
   'ESTATE' elm_name elm_type '*'
   if rc = Ø then
      do
         'MAKEBUF'
         'LISTFILE' elm_name elm_type '* (DATE FIFO'
         parse pull . . . . . date time .
         'DROPBUF'
         date = right(date, 8, 'Ø')
         time = right(time, 8, 'Ø')
         date = translate('781245', date, '12345678') ]] time
         if m > Ø then
            if date > dte.vid.m then gen.vid.m = 1
            else nop
         else
            dte.vid.Ø.n = date
      end
   else
      call error 67
   return

                                       /* ------------- */
check_elm_aux:                         /* Check for AUX and DOC      */
   parse arg vid

   n = elm.vid + 1
   elm.vid = n
   elm_nme.vid.n = elm_name
   elm_typ.vid.n = elm_type
   'ESTATE' elm_name elm_type '*'
   if rc ¬= Ø then call check_elm_missing
   return

                                       /* ------------- */
check_elm_missing:                     /* Check missing Elements     */
   if ¬(gm_comm = 'BD' & (elm_type = 'TEXT' ] elm_type = 'MACLIB' ],
```

```
                  elm_type = 'TXTLIB' ] elm_type = 'LOADLIB' ],
                  elm_type = 'MODULE' ] elm_type = 'LISTING')) then
      call error 68
   return

                                       /* ------------- */
command:                               /* Process Command statement  */
   no_object = Ø                       /* Allow Object statements    */
   select
      when type = 'MLB' then vid = '@MLB'
      when type = 'TXT' then vid = '@TXT'
      when type = 'LLB' then vid = '@LLB'
      when type = 'MOD' then vid = '@MOD'
      otherwise
         do
            call error 7Ø
            return
         end
   end
   m = cnt.vid                         /* Register command           */
   n = cmd.vid.m + 1
   cmd.vid.m = n
   cmd.vid.m.n = command
   return


                                       /* ------------- */
process:                               /* Let's do something         */
   select
      when gm_comm = 'BD' then call process_build
      when gm_comm = 'CK' then call process_check
      when gm_comm = 'CP' then call process_copy
   end
   return

                                       /* ------------- */
process_build:                         /* Generate Objects           */
   do k = 1 to cnt.@mlb                /* MACLIB first               */
      if gen.@mlb.k ] opt_all then call gen_object '@MLB MACLIB'
   end

   if maxrc = Ø then
      call def_library '@GLM MACLIB'   /* Define search libraries    */

   if maxrc = Ø then                   /* Create TEXT if no problems */
      do k = 1 to cnt.@txt
         do g = 1 to elm.@txt.Ø until gen.@txt.k
            if dte.@txt.Ø.g > dte.@txt.k then gen.@txt.k = 1
         end
```

```
         if gen.@txt.k ] opt_all then call gen_object '@TXT TEXT'
      end

   if maxrc = Ø then                    /* Create TXTLIB              */
     do k = 1 to cnt.@tlb
      if gen.@tlb.k ] opt_all then call gen_object '@TLB TXTLIB'
   end

   if maxrc = Ø then
      call def_library '@GLT TXTLIB'    /* Define TXT libraries       */

   if maxrc <= 4 then                   /* Create LOADLIB and Modules */
      do
         do k = 1 to cnt.@llb
            if gen.@llb.k ] opt_all ] generate & elm.@llb.k = Ø then
               call gen_object '@LLB LOADLIB'
         end
         do k = 1 to cnt.@mod
            if gen.@mod.k ] opt_all ] generate & elm.@mod.k = Ø then
               call gen_object '@MOD MODULE'
         end
        end


   if maxrc > Ø then call error 8Ø
   if maxrc = Ø & ¬generate then call error 81
   return

                                        /* ------------- */
process_check:                          /* Process CHECK command      */
   call error 82                        /* It's all done...           */
   return

                                        /* ------------- */
process_copy:                           /* Process COPY command       */
   if opt_frm = '' then opt_frm = 'A'
   cp_options = '(OLDDATE'
   say 'Copy requested from' opt_frm 'disk to' opt_to 'disk.'

   select
      when opt_all then groups = 'MOD LLB AUX MLB SCE TXT DOC'
      when opt_exe then groups = 'MOD LLB AUX'
      when opt_sce then groups =         'AUX MLB SCE'
      when opt_bsc then groups = 'MOD LLB AUX MLB SCE TXT'
      otherwise         groups = 'MOD LLB AUX MLB SCE DOC'
   end

   do i = 1 to words(groups) until rc > Ø
      key = word(groups, i)
```

```
select
   when key = 'LLB' then
      do
         say 'Copy LOAD libraries.'
         do k = 1 to cnt.@llb until rc > 0
            say '   Copy' nme.@llb.k 'LOADLIB' opt_frm'.'
            'COPYFILE' nme.@llb.k 'LOADLIB',
                       opt_frm '= =' opt_to cp_options
         end
      end
   when key = 'MOD' then
      do
         say 'Copy CMS modules.'
         do k = 1 to cnt.@mod until rc > 0
            say '   Copy' nme.@mod.k 'MODULE' opt_frm'.'
            'COPYFILE' nme.@mod.k 'MODULE',
                       opt_frm '= =' opt_to cp_options
         end
      end
   when key = 'MLB' then
      do
         say 'Copy Macro Libraries.'
         do k = 1 to cnt.@mlb until rc > 0
            say '   Copy' nme.@mlb.k 'MACLIB' opt_frm' members.'
            do m = 1 to elm.@mlb.k until rc > 0
               'COPYFILE' elm_nme.@mlb.k.m elm_typ.@mlb.k.m,
                          opt_frm '= =' opt_to cp_options
            end
         end
      end
   when key = 'AUX' then
      do
         say 'Copy AUX files.'
         do k = 1 to cnt.@aux until rc > 0
            do m = 1 to elm.@aux until rc > 0
               'COPYFILE' elm_nme.@aux.m elm_typ.@aux.m,
                          opt_frm '= =' opt_to cp_options
            end
         end
      end
   when key = 'SCE' then
      do
         say 'Copy Source files.'
         do k = 1 to cnt.@txt until rc > 0
            'COPYFILE' elm_nme.@txt.k.1 elm_typ.@txt.k.1,
                       opt_frm '= =' opt_to cp_options
         end
      end
   when key = 'TXT' then
```

```
            do
               say 'Copy Object (TEXT) files.'
               do k = 1 to cnt.@txt until rc > Ø
                  'COPYFILE' nme.@txt.k 'TEXT',
                             opt_frm '= =' opt_to cp_options
               end
            end
         when key = 'DOC' then
            do
               say 'Copy Documents (DOC files).'
               do k = 1 to cnt.@doc until rc > Ø
                  do m = 1 to elm.@doc
                     'COPYFILE' elm_nme.@doc.m elm_typ.@doc.m,
                                opt_frm '= =' opt_to cp_options 'PACK'
                  end
               end
            end
      end
   end

   if rc > Ø then call error 83
   return

                                       /* ------------- */
def_library:                           /* Define library search      */
   parse arg vid libtype

   if cnt.vid > Ø then                 /* Do we have a library?      */
      do
         'GLOBAL' libtype elm.vid.1    /* Execute GLOBAL             */
         if rc > maxrc then maxrc = rc
         if rc > Ø then call error 9Ø  /* Check for problems         */
      end
   return

                                       /* ------------- */
gen_object:                            /* Create objects             */
   parse arg vid objtype
   generate = 1                        /* We changed something       */

   if vid ¬= '@TXT' then
      do
         'ESTATEW' nme.vid.k objtype 'A'
         if rc = Ø then
            do
               say 'Erasing “'nme.vid.k objtype 'A”.'
               'ERASE' nme.vid.k objtype 'A'
               if rc > maxrc then maxrc = rc
            end
```

```
         say 'Creating new' objtype '"'nme.vid.k'".'
      end

   if cmd.vid.k = 0 then               /* Any specific commands      */
      select                           /* No, use defaults           */
         when vid = '@TXT' then call gen_text    vid objtype
         when vid = '@MLB' then call gen_library vid objtype
         when vid = '@TLB' then call gen_library vid objtype
         when vid = '@LLB' then call gen_loadlib vid objtype
         when vid = '@MOD' then call gen_module  vid objtype
      end
   else                                /* Use specified commands     */
      do c = 1 to cmd.vid.k until rc > 0
         say '   Executing command "'cmd.vid.k.c'".'
         cmd.vid.k.c
         if rc = -3 then say '   Unknown CMS command.'
         if rc > 0 then say '   Return code from last command is' rc'.'
         if rc < 0 then rc = 16
         if rc > maxrc then maxrc = rc
      end

   if rc > 4 & objtype ¬= 'MODULE' then
      do
         'MAKEBUF'
         'LISTFILE' nme.vid.k objtype '* (DATE FIFO'
         parse pull . . objmode .
         'DROPBUF'
         say '   Erasing "'nme.vid.k objtype objmode'".'
         'ERASE' nme.vid.k objtype objmode
      end
   return

                                       /* ------------- */
gen_library:                           /* Create MACLIB or TXTLIB    */
   parse arg vid libtype

   say '   Adding member' elm_nme.vid.k.1 elm_typ.vid.k.1'.'
   libtype 'GEN' nme.vid.k elm_nme.vid.k.1
   if rc > maxrc then maxrc = rc
   do m = 2 to elm.vid.k while rc = 0
      say '   Adding member' elm_nme.vid.k.m elm_typ.vid.k.m'.'
      libtype 'ADD' nme.vid.k elm_nme.vid.k.m
      if rc > maxrc then maxrc = rc
   end
   return

                                       /* ------------- */
gen_loadlib:                           /* Create LOADLIB             */
   parse arg vid libtype
```

```
   say '   Adding member' elm_nme.vid.k.1 elm_typ.vid.k.1'.'
   libtype 'GEN' nme.vid.k elm_nme.vid.k.1
   if rc > maxrc then maxrc = rc
   do m = 2 to elm.vid.k while rc = Ø
      say '   Adding member' elm_nme.vid.k.m elm_typ.vid.k.m'.'
      libtype 'ADD' nme.vid.k elm_nme.vid.k.m
      if rc > maxrc then maxrc = rc
   end
   return

                                    /* ------------- */
gen_text:                           /* Assemble files             */
   say 'Assembling' nme.@txt.k'.'
   say '   Executing "HLASM' strip(nme.@txt.k opt.@txt.k)'".'
   'SET CMSTYPE HT'
   'HLASM' nme.@txt.k opt.@txt.k
   tmp = rc
   'SET CMSTYPE RT'
   rc = tmp
   if rc > Ø then say '   Return code from HASM is' rc'.'
   if rc > maxrc then maxrc = rc
   do m = 1 to cnt.@mod
      do n = 1 to elm.@mod.m
         if elm_nme.@mod.m.n = nme.@txt.k & elm_typ.@mod.m.n = 'TEXT'
         then gen.@mod.m = 1
      end
   end
   return


gen_module:                         /* Create CMS Modules         */
   say 'Creating new MODULE "'nme.@mod.k'".'
   'LOAD' nme.@mod.k
   if rc > maxrc then maxrc = rc
   if rc = Ø then 'GENMOD' nme.@mod.k
   if rc > maxrc then maxrc = rc
   if rc > Ø then say '   Return code from last command is' rc'.'
   return

                                    /* ------------- */
error:                              /* Display Error messages     */
   parse arg num
   select
      when num = '1Ø' then msg = 'Invalid statement code "'code'".'
      when num = '11' then msg = 'Errors found in control file.'
      when num = '2Ø' then msg = 'Invalid parameter "'in_rest'".'
      when num = '21' then msg = 'Invalid command "'in_comm'".'
      when num = '22' then msg = 'Invalid option "'in_opts'".'
```

```
   when num = '23' then msg = 'Missing or invalid TO option.'
   when num = '24' then msg = 'Too many options.'
   when num = '25' then msg = 'Missing name for control file.'
   when num = '26' then msg = 'Invalid name "'fn ft fm'"',
                              'for control file.'
   when num = '27' then msg = 'Control file "'fn ft fm'" not found.'
   when num = '30' then msg = 'Object statement' type 'found in an',
                              'invalid position.'
   when num = '31' then msg = 'Unknown object type "'type'".'
   when num = '40' then msg = 'Object name is missing on' type,
                              'statement.'
   when num = '41' then msg = 'Invalid Object statement.'
   when num = '42' then msg = 'Missing or invalid type for GLB',
                              'statement.'
   when num = '43' then msg = 'Duplicate or invalid GLB statement.'
   when num = '44' then msg = 'AUX and DOC statements must be',
                              'unique.'
   when num = '50' then msg = 'Element statement' type 'found in an',
                              'invalid position.'
   when num = '51' then msg = 'Element "'elm_name elm_type'" found',
                              'for unknown object' type'.'
   when num = '60' then msg = 'Missing name for element.'
   when num = '61' then msg = 'Invalid Element statement.'
   when num = '62' then msg = 'Invalid type "'elm_type'" for',
                             obj_name 'object.'
   when num = '63' then msg = 'Options are not allowed for GLB',
                              'elements.'
   when num = '64' then msg = 'Source file must be defined first.'
   when num = '65' then msg = 'Invalid source name. It must be the',
                              'same as the object.'
   when num = '66' then msg = 'ASSEMBLE files cannot be used as',
                              'gobal sources.'
   when num = '67' then msg = 'Source file "'elm_name elm_type '*'"',
                              'not found.'
   when num = '68' then msg = 'Element "'elm_name elm_type'" not',
                              'found.'
   when num = '70' then msg = 'Command statement found for unknown',
                              'or unsupported object.'
   when num = '80' then msg = 'Maximum return code is' maxrc'.'
   when num = '81' then msg = 'No objects created. Nothing to do.'
   when num = '82' then msg = 'No errors found in control file.'
   when num = '83' then msg = 'Unable to copy files.'
   when num = '90' then msg = 'Error defining' libtype 'libraries.'
   otherwise msg = 'Error message "'num'" not defined.'
end
say msg
error = 1
return
```

CSC $$CNTL$$

```
*    > Object
*      GLB - OS/CMS Libraries to activate (GLOBAL)
*      LLB - OS/CMS LoadLib
*      TLB - OS/CMS TextLib
*      MLB - OS/CMS MacLib
*      MOD - CMS Module
*      TXT - CMS TEXT file
*      AUX - Auxiliary material
*      DOC - Extra documentation
*            LISTING
*    : Component
*    ( Options
*    - Generation command
*    ; Command separator
*   /* Comments
*
> GLB MACLIB                          /* Define libraries
      : CSC                           /* CSC  Macro library
      : DMSGPI                        /* CMS  libraries
      : DMSOM
      : HCPGPI                        /* CP   libraries
*     : HCPPSI                        /*      CSCMGX only
*     : HCPOM1                        /*      CSCMGX only
*     : HCPOM2                        /*      CSCMGX only
      : OSMACRO                       /* MVS  libraries

> MLB CSC
      : CSCHDR   MACRO                /* CSC Header
      : CSCLINK  MACRO                /* GO and BACK macros
      : CSCDATA  MACRO                /* CSC common Data area
      : CSCDS    MACRO                /* Contains all DSECTS
      : CSCCMMD  MACRO                /* Expand command tables

> MOD CSCSVP
      : CSCSVP   TEXT
      : CSCCPW   TEXT   ;   : CSCRDF   TEXT   ;   : CSCBLD   TEXT
      : CSCCFG   TEXT   ;   : CSCRLS   TEXT   ;   : CSCCLS   TEXT
      : CSCSCN   TEXT   ;   : CSCSEV   TEXT
      : CSCOPC   TEXT   ;   : CSCOPQ   TEXT   ;   : CSCOPA   TEXT
      : CSCUSC   TEXT   ;   : CSCUIN   TEXT   ;   : CSCULC   TEXT
      : CSCURL   TEXT   ;   : CSCUOP   TEXT   ;   : CSCUEX   TEXT
      : CSCUSA   TEXT   ;   : CSCUST   TEXT   ;   : CSCUPR   TEXT
      : CSCUSB   TEXT
      : CSCWRP   TEXT
      : CSCMSG   TEXT   ;   : CSCMSL   TEXT
      : CSCRNL   TEXT   ;   : CSCRNC   TEXT
      : CSCTMR   TEXT
```

```
      - LOAD     CSCSVP  (CLEAR
      - GENMOD   CSCSVP
      - CSCSVP                          /* Testing

> MOD CSCUSR
      : CSCUSR   TEXT
      - LOAD     CSCUSR  (CLEAR RLDSAVE
      - GENMOD   CSCUSR  (ALL
      - CSCUSR                          /* Testing

> TXT : CSCSVP   ASSEMBLE  (XREF(SHORT)

> TXT *                                 /* Global macros
      : CSCHDR   MACRO
      : CSCLINK  MACRO
      : CSCDATA  MACRO
      : CSCDS    MACRO

> TXT : CSCCFG   ASSEMBLE  (XREF(SHORT)   ; : CSCCMMD MACRO
> TXT : CSCRLS   ASSEMBLE  (XREF(SHORT)
> TXT : CSCCLS   ASSEMBLE  (XREF(SHORT)
> TXT : CSCCPW   ASSEMBLE  (XREF(SHORT)
> TXT : CSCRDF   ASSEMBLE  (XREF(SHORT)
> TXT : CSCBLD   ASSEMBLE  (XREF(SHORT)
> TXT : CSCMSG   ASSEMBLE  (XREF(SHORT)
> TXT : CSCMSL   ASSEMBLE  (XREF(SHORT)
> TXT : CSCSCN   ASSEMBLE  (XREF(SHORT)
> TXT : CSCSEV   ASSEMBLE  (XREF(SHORT)
> TXT : CSCOPC   ASSEMBLE  (XREF(SHORT)   ; : CSCCMMD MACRO
> TXT : CSCOPQ   ASSEMBLE  (XREF(SHORT)   ; : CSCCMMD MACRO
> TXT : CSCOPA   ASSEMBLE  (XREF(SHORT)   ; : CSCCMMD MACRO
> TXT : CSCUSC   ASSEMBLE  (XREF(SHORT)   ; : CSCCMMD MACRO
> TXT : CSCUIN   ASSEMBLE  (XREF(SHORT)
> TXT : CSCULC   ASSEMBLE  (XREF(SHORT)
> TXT : CSCURL   ASSEMBLE  (XREF(SHORT)
> TXT : CSCUOP   ASSEMBLE  (XREF(SHORT)   ; : CSCCMMD MACRO
> TXT : CSCUEX   ASSEMBLE  (XREF(SHORT)
> TXT : CSCUSA   ASSEMBLE  (XREF(SHORT)
> TXT : CSCUSB   ASSEMBLE  (XREF(SHORT)
> TXT : CSCUST   ASSEMBLE  (XREF(SHORT)   ; : CSCCMMD MACRO
> TXT : CSCUPR   ASSEMBLE  (XREF(SHORT)
> TXT : CSCWRP   ASSEMBLE  (XREF(SHORT)
> TXT : CSCRNL   ASSEMBLE  (XREF(SHORT)
> TXT : CSCRNC   ASSEMBLE  (XREF(SHORT)
> TXT : CSCTMR   ASSEMBLE  (XREF(SHORT)

> TXT : CSCUSR   ASSEMBLE  (XREF(SHORT)

> TXT : CSCMGX   ASSEMBLE  (XREF(SHORT) SYSPARM(SUP)
```

```
> AUX
      : CSC      $$CNTL$$
      : CSC      CONFIG
      : GM       EXEC
      : CSC      EXEC
      : CSCUPDT  EXEC
      : CSCSMP   ASSEMBLE
      : *        HELPABBR
      : *        HELPMENU
      : *        HELPTASK
      : *        HELPCSC
      : *        HELPCSCC
      : *        HELPCSCQ
      : *        HELPCSCS

> DOC
*     : LOAD     MAP       (UNPACK
*     : LISTING
```

Fernando Duarte
Analyst (Canada)
fernando_duarte@vnet.ibm.com

VM news

Beware of these characters
They tend to change during translation between EBCDIC and ASCII.

\	backslash	{ }	braces	[ ]	brackets
¦	broken bar	¢	cent	^	circumflex
$	dollar	`	grave accent	¬	not
£	pound sterling	~	tilde	|	vertical bar

SOFTWARE DIVERSIFIED SERVICES