161 VM update

January 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 612-571-9000
fax 612-572-1721

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

Editor

Rick Kothe
rickkothe@sdsusa.com
612-571-9000 ext. 121

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.

Beginning with January 2000, individual articles will also be available in html so that readers can more readily copy the codes.

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. The rate of payment varies. For information, contact the editor.

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.

Powerful full-screen map compiler for REXX programmers

This article describes all steps needed to create and use full-screen maps. For each map a souce should be defined, which can then be compiled to an XEDIT macro.

I built it already a long time ago, but it has proven to be very easy to use and very stable. A lot of our production systems make use of it and it never caused us a problem. The code runs under VM/HPO and VM/ESA.

This article has six parts:

1. define the source of a map
2. compile the source map to an XEDIT macro
3. using maps in your REXX program
4. programs and files needed for the package
5. restrictions
6. examples

1. DEFINE THE SOURCE OF A MAP

To define a map source you should take the following steps:

1.1 create a file (using XEDIT) having a filename of your own choice and a filetype of MAP. Filemode does not matter.

1.2. the first 24 lines of this source file should litterally describe the contents of the map, with the following exceptions:

a. positions at which a screen attribute must be defined, e.g. BLINK, REVERSE, HIGH etc., should contain attribute characters. Definitions of such characters will be explained in 1.3).

b. all positions in the map which belong to variable fields, should be filled with a special character (see 1.3)

c. the position right before every variable field in a map should contain an attribute character (as described in 1.2.a.)

d. in general it is adviseable to put an attribute character right before fixed texts as well. If you don't, you will

depend on the XEDIT defaults in use

1.3. all lines after the map definition (so, from line 25 and onward) should contain the definitions of all special characters and field names used in the map source.

AD 1.3. a. DEFINITION OF CONTROL CHARACTERS / ATTRIBUTES

To define control characters / attributes for the map, you should code one or more lines as described below in the source of the map, after lines 1 thru 24 :

SET CTLCHAR <special-character> <def1> <def2> <def3> <def4>

Description of the options used :

<special-character>
This can be any character except the double quote.
Rules :
- do not use a character which does also appear in normal text in the map
- no duplicates allowed, so, you may only specify it in one SET statement

<def1> can be ESCAPE, FIELD, NOPROTECT or PROTECT
- if ESCAPE
  then this character will be used for internal purposes only; in this case <def2> thru <def4> shouldn't be specified.
  Note :
  it is mandatory to define exactly 1 SET CTLCHAR ESCAPE statement.
- if FIELD
  then this character will be used for all positions of variable fields in the map; in this case <def2> thru <def4> shouldn't be specified
- if NOPROTECT
  then this character defines an unprotected variable field in the map; in this case you should define <def2> thru <def4> as well
- if PROTECT
  then this character defines a protected variable field in the map; in this case you should define <def2> thru <def4> as well

<def2> can be BLUE, RED, PINK, GREEN, TURQUOISE, YELLOW,

WHITE
or DEFAULT; if DEFAULT, the default color in use by XEDIT for this type of field will be used.

<def3> can be BLINK, REVVIDEO, UNDERLINE or NONE
- if BLINK
  in this case the variable field will appear blinking on the screen
- if REVVIDEO
  in this case the variable field will appear with reversed foreground and background colours on the screen
- if UNDERLINE
  in this case the variable field will appear underlined on the screen
- if NONE
  in this case none of the above features apply to the variable field

<def4> can be HIGH, NOHIGH, INVISIBLE
- if HIGH
  in this case the variable field will appear with high intensity on the screen
- if NOHIGH
  in this case the variable field will appear with normal intensity on the screen
- if INVISIBLE
  in this case the contents of the variable field will not appear on the screen; this is a useful feature for a password field to be entered

AD 1.3. b. DEFINITION OF DYNAMIC FIELD ATTRIBUTES

If you wish to be able to alter field attributes on the fly from within your REXX program, independent of what you specified in the map source, you can use the following statement in line 25 and onward in the map :
SET DYNAMIC <ctlchar1> .... <ctlcharn>

<ctlchar1> thru <ctlchrn> is a list of control characters / attributes, as explained above, but only those having a <def1> equal to NOPROTECT or PROTECT.

In your REXX exec you can use the special stem variable <field>.A, where <field> is the name of the variable field.

To assign an attribute dynamically to the variable field <field> in the map, you should assign a numeric value to the REXX variable <field>.A in your REXX program, which does correspond with the offset of the desired control character in the list of control characters specified in the SET CTLCHAR statement.

However, if you assign a value of zero, the attribute originally defined for that field will be used.

Example :
  You specified the following statement :
    SET DYNAMIC $ % @
  Then you should code:
    <field>.A = 2
  to assign the attribute % (the second in the list) to the variable field <field> in the map.

AD 1.3. c. ASSIGN FIELD NAMES TO VARIABLE FIELDS

It is mandatory to supply a field name for all the variable fields defined in the map.

Fields are defined by placing one or more consecutive characters as defined with the SET CTLCHAR FIELD statement.

To define inividual variable fields, specify for each field a line :
  SET FIELD <num> NAME <name> JUSTIFY <just> <case>

Description of the options used :

  <num>
    this is the number of the field in the map from left to right and from top to bottom, starting from 1.
    Note :
      since it can be a tedious task to assign the numbers manually put a dummy number (e.g. 1) in place for <num> for all fields; then execute the supplied XEDIT macro named MAPRENUM, which will automatically assign the correct numbers

  <name>
    this is the name which can be used in your REXX program to refer to a variable field in the map.

Note :
- it should be a valid REXX identifier (can even be a STEM variable)
- the following reserved names should not be used for field names :
  @ALARM : used to be able to sound a beep
  @CURSORRF : cursor read field (returned after map read)
  @CURSORRO : cursor read offset (returned after map read)
  @CURSORWF : cursor write field (you can specify it)
  @CURSORWO : cursor write offset (you can specify it)
  @KEY : function key used (returned after map read)
  @Q@ : used internally
  @MAPNAME : used internally
  @MAPWAIT : used internally
  @I@. : used internally
  _@ : used internally
- if you assign duplicate names, always the last one will be used. This can be a useful action for a read only (PROTECT) field, which must be displayed more than once in the same map.
- if there are several fields in the map which have the same characteristics and have to be accessed with indexing techniques, you may assign them REXX STEM names with appropriate STEM indexes.
  For example, if you wish to display an array of 3 cars in a map you could define in the source map :
  SET FIELD 5 NAME CAR.1 JUSTIFY LEFT UPPERCASE
  SET FIELD 6 NAME CAR.2 JUSTIFY LEFT UPPERCASE
  SET FIELD 7 NAME CAR.3 JUSTIFY LEFT UPPERCASE
  you could fill the fields in your REXX program (from e.g. a database source) with the following lines of code :

```
do @i = 1 to 3
  @car.@i = @dbcar.@i
end
```

<just>
this is the field justification, which can be LEFT, CENTER, CENTRE or RIGHT. It causes the REXX variable corresponding to the name of the variable field in the map to be positioned in the field according to the justification specified.
Note :
- in the same way as you may use <field>.A to assign attributes to fields dynamically, you can dynamically alter the field justification by assigning one of the appropriate values (e.g. "LEFT") to the REXX variable <field>.J in your REXX program

<case>
this is the case of the field : UPPERCASE or LOWERCASE
- if LOWERCASE the contents of the field in the map at execution time will be handed unchanged to the corresponding REXX variable
- if UPPERCASE the contents of the field in the map at execution time will be handed to the corresponding REXX variable after having been translated to uppercase
- in the same way as you may use <field>.A to assign attributes to fields dynamically, you can dynamically alter the field case by assigning one of the appropriate values (e.g. "LOWERCASE") to the REXX variable <field>.U in your REXX program

2. COMPILE THE SOURCE MAP TO AN XEDIT MACRO

Before compilation of the map, you should assign the correct offset numbers to all fields. For this purpose you should xedit the map source and enter the following command :

MAPRENUM

The macro MAPRENUM executes the following tasks :
- renumber all field numbers in all SET FIELD control statements
- translates lines 25 thru end-of-file to uppercase

After this step you should store the map source and proceed with the next step, the compilation :

MAPCOMP <filename> <filetype> <filemode>

Note :
Because CMS is able to maintain lowercase fileids, it is mandatory to supply the filetype (MAP) in uppercase (if you do specify it, which is optional).

If you omit filetype, then "MAP" will be assumed.
If you omit filemode, then the first file in the CMS search order will be compiled.

If MAPCOMP detects an error, an apprpriate message will be displayed and the compilation will be aborted. In that case you should first correct the map source and the recompile it.

The output of the compilation will have the same filename as the input map source; the filtype will be "XEDIT" and the filemode "A".

After correct compilation the compiled map will be displayed using the proigram MAPDISPL.

If you would like to display the compiled map without having to invoke a program written by yourself, you can use the following supplied EXEC :
  MAPDISPL <filename> <filetype> <filemode>
It will display the map with the names (or if the field size is not big enough a part of it) of the variable fields in the corresponding variable fields.

3. USING MAPS IN YOUR REXX PROGRAM

FILE PLACEMENT

The compiled maps (XEDIT macro's) have to be on an accessible filemode.
Also the file MAPCOPY COPY shouls be accessible.
Please read also "4. PROGRAMS AND FILES NEEDED FOR THE PACKAGE"

COPY BOOK USAGE

To be able to use a map within your REXX program, you need to copy the supplied copy book "MAPCOPY COPY *" at the end of your REXX program. In this book some essential subroutines (INIT_MAP and WRITE_MAP) are supplied. Note that you should end your main program with either RETURN or EXIT to avoid a so called "fall through" condition in the program.

INITIALIZE MAP

Before you start using a map in your program (you may use as many maps as you wish) you must execute the map initialization routine INIT_MAP as supplied in the copy book mentioned before for each map :
  call init_map 'mapname'

INIT_MAP executes the following initializations :
  - for all fields in the map :
      <field>.A = <default attribute>
      <field>.J = <default justification>

<field>.U = <default case translation>
<field>.C = <field number of the field in the map>

- cursor positioning fields :
  @CURSORWF = 0
  @CURSORWO = 0
  @CURSORRF = 0
  @CURSORRO = 0

CURSOR MANIPULATION

For cursor positioning 2 fields are available :
@CURSORWF (cursor write fieldnumber)
@CURSORWO (cursor write offset)
@CURSORRF (cursor read fieldnumber)
@CURSORRO (cursor read offset)

If @CURSORWF contains a value of zero, then @CURSORWO will be the offset
relative to the begin of the PHYSICAL map. So, a value of 0 will put the cursor at the first row in the first column and a value of 83 will put the cursor in the second row in column 4.

If @CURSORWF contains a value bigger than zero, then this value will be interpreted as the field sequence number in the map definition. In this case @CURSORWO contains the relative offset to the start of that field. This last method is preferrable above the first one. In most cases you should use a value of zero for @CURSORWO, to position the cursor right at the beginning of a field.

There is a special REXX stem variable for each field in the map, which is called <field>.C. This field contains the field number of the field <field> in the map. When you have, for example a field TEST in your map, you can position the cursor at the start of that field with the following REXX code :

```
@CURSORWF = TEST.C
@CURSORWO = 0
```

After having written a map, when the user has entered something, the fields @CURSORRF and @CURSORRO contain the actual cursor position in the map. @CURSORRF always refers to the nearest field, trying to obtain a positive value for @CURSORRO. However, in some cases (e.g. when the cursor is positioned before the first field in the map, @CURSORRO will contain a negative offset value to that field.

HOW TO BEEP WHEN YOU DISPLAY AN ERROR MESSAGE

The field @ALARM can be filled with a value of 'ALARM' to cause the map to BEEP when displayed. Any other value will not cause a BEEP.

HOW TO OVERRULE THE DEFAULTS AS DEFINED IN THE SOURCE MAP

Modify :
<field>.A to change the field attribute (see SET DYNAMIC)
<field>.J to change the justification for the field
<field>.U to change the translation for the field

Note :
If you supply a bad value, always the default as defined in the corresponding SET FIELD statement in the source map will be used.

HOW TO DISPLAY / READ A MAP

In order to display the map in your REXX program you should code :
call write_map 'mapname mapwait'

The option "MAPWAIT" can have the following values :
- WAIT ===> display screen and read
- NOWAIT ===> display screen, but do not read
- otherwise ===> display screen and read

The NOWAIT value should only be used for long running processess in order to be able to display intermediate results.

HOW TO HANDLE FUNCTION KEYS AND THE ENTER KEY

After displaying a map with option WAIT, you can test the special variable @KEY. At that time it will contain one of the following values :
ENTER
PF1 .. PF24
PA1 .. PA3

4. PROGRAMS AND FILES NEEDED FOR THE PACKAGE

The whole package consists of the following files :
MAPCOMP EXEC : compiles from ft=MAP to ft=XEDIT
MAPCOPY COPY : copybook to be included in your REXX program
MAPCOPY FILE : dummy file needed by the copybook MAPCOPY COPY
MAPDISPL EXEC : displays compiled MAPs
MAPRENUM XEDIT : can adjust MAP source before compilation

IMPORTANT NOTE

In the copybook MAPCOPY COPY is a FIXED reference to file MAPCOPY FILE X.
The reason for using X as the filetype is that we have put all our shared stuff on a disk which every CMS userid always will access with the filemode X.

You should adjust the filemode according to a filemode which is appropriate for your site.

5. RESTRICTIONS

In fact there is only one restriction. You can only define screens of 24 by 80 columns.

If you have access to a physical terminal (or an emulated one) having a higher capacity, this isn't a problem. The map will be handled in a normal way. It will only be displayed in the upper left corner of the bigger physical screen area.

6. EXAMPLES

An example has been provided which contains a lot (but not all) possibilities of the package, so, you can see for yourself.

It consists of the files :
MAPEXAMP EXEC : exec which will invoke the map MAPEXAMP
MAPEXAMP MAP : source MAP; you need to compile it with MAPCOMP first

MAPCOMP.RXX

```
/*=================================================================*
/
/* THIS EXEC COMPILES A FULL-SCREEN MAP
*/
/*    A.P. VAN WINGERDEN                    VERSION V-1-9
*/
/*=================================================================*
/
   PARSE ARG @MAPFN @MAPFT @MAPFM @I@
   IF @I@ = 'DISPLAY'
      THEN DO
              @I@ = @MAPFN
              DROP  @MAPFN
              DROP  @MAPFT
              DROP  @MAPFM
              /* DISPLAY THE GENERATED MAP */
              CALL INIT_MAP @I@
              @ALARM  = 'ALARM'
              CALL WRITE_MAP @I@
              RETURN
           END
   @VERSION = 'V-1-9'
   IF @MAPFN  = ''    THEN CALL MAP_ERROR 1
   IF @MAPFT  = ''    THEN @MAPFT = 'MAP'
   IF @MAPFT <> 'MAP' THEN CALL MAP_ERROR 2
   IF @MAPFM  = ''    THEN @MAPFM = '*'
   @MAPSOURCE = @MAPFN @MAPFT @MAPFM
  'STATE' @MAPSOURCE
   IF RC <> 0 THEN CALL MAP_ERROR 3
   @MAPOBJECT = @MAPFN 'XEDIT A'
  'ERASE' @MAPOBJECT
   @t1 = '@I@ @Q@ @CURSORRF @CURSORRO @CURSORWF @CURSORWO @ALARM @KEY'
   @t2 = 'LEFT RIGHT CENTRE CENTER'
   @t3 = 'UPPERCASE LOWERCASE'
   @t4 = 'FIELD ESCAPE PROTECT NOPROTECT'
   @t5 = 'BLUE RED PINK GREEN TURQUOISE YELLOW WHITE DEFAULT'
   @t6 = 'BLINK REVVIDEO UNDERLINE NONE'
   @t7 = 'HIGH NOHIGH INVISIBLE'
   @F       = 0
   @LINENUM = 0
   DO @L = 1 TO 24
      CALL READ_MAP_SOURCE
      IF @RC <> 0 THEN CALL MAP_ERROR 4
      @LINE.@L = @RECORD
   END
   @AI = 0
   @FI = 0
```

```
   @FH = 0
   @DC = ''
   @DCL = 0
   @EC_FOUND = 'N'
   @FC = ''
   DO FOREVER
      CALL READ_MAP_SOURCE
      IF @RC  = 2 THEN LEAVE
      IF @RC <> 0 THEN CALL MAP_ERROR 5
      CALL SCAN_STATEMENTS translate(@record)
   END
   IF @AI > 0,
    & @EC_FOUND = 'N' THEN CALL MAP_ERROR 6
   DO @L = 1 TO 24
      CALL EXTRACT_FIELD_SPECS
   END
   CALL CHECK_DYNAMIC_CTLCHARS
   CALL CHECK_FIELD_SPECS
   CALL CHECK_FIELD_ATTRIBUTES
   CALL OVERLAY_MAP
   CALL INSERT_CONTROL_INFO
   CALL WRITE_MAP_OBJECT
  'FINIS' @mapsource
   CALL MAPCOMP @MAPFN @MAPFT @MAPFM 'DISPLAY'
   RETURN

SCAN_STATEMENTS:
   PARSE ARG @ATT
   IF WORD(@ATT,1) <> 'SET'      THEN CALL MAP_ERROR 7 @att
   SELECT
      WHEN WORD(@ATT,2) = 'FIELD'    THEN CALL SCAN_FIELD
      WHEN WORD(@ATT,2) = 'CTLCHAR'  THEN CALL SCAN_CTLCHAR
      WHEN WORD(@ATT,2) = 'DYNAMIC'  THEN CALL SCAN_DYNAMIC_CTLCHARS
      OTHERWISE CALL MAP_ERROR 8 @att
   END
   RETURN

SCAN_FIELD:
   PARSE VAR @ATT @P1 @P2 @P3 @P4 @P5 @P6 @P7 @P8 @REST
   IF DATATYPE(@P3,'W') <> 1           THEN CALL MAP_ERROR 17 @att
   IF @P4               <> 'NAME'      THEN CALL MAP_ERROR 29 @att
   IF LENGTH(@P5)        >  28,
    | SYMBOL(@P5)        = 'BAD',
    | find(@t1,@p5)      <> 0          THEN CALL MAP_ERROR 18 @att
   IF @P6               <> 'JUSTIFY'   THEN CALL MAP_ERROR 19 @att
   if find(@t2,@p7) = 0                THEN CALL MAP_ERROR 20 @att
   if find(@t3,@p8)       = 0          THEN CALL MAP_ERROR 33 @att
```

```
   IF @REST    <> ''                   THEN CALL MAP_ERROR 35 @att
   /* V-1-9 : Fast check for existing field number */
   if symbol('@xfnu.@p3') = 'VAR'
      then call map_error 21 @att
   @FI = @FI + 1
   IF @P3 > @FH THEN @FH = @P3
   /* V-1-9 : Fast check for existing field number */
   @xfnu.@p3 = ''
   @FNU.@FI = @P3
   @FNA.@FI = @P5
   @FJU.@FI = @P7
   @FUP.@FI = @P8
   RETURN

SCAN_CTLCHAR:
   PARSE VAR @ATT @P1 @P2 @P3 @P4 @P5 @P6 @P7 @REST
   IF LENGTH(STRIP(@P3,B)) > 1 THEN CALL MAP_ERROR 9 @att
   IF @P3       = '"'          THEN CALL MAP_ERROR 32 @att
   if find(@t4,@p4) = 0          THEN CALL MAP_ERROR 10 @att
   IF @P4       = 'FIELD'
      THEN DO
              IF @FC <> '' THEN CALL MAP_ERROR 14 @att
              @FC = STRIP(@P3,B)
              IF @FC  = '' THEN CALL MAP_ERROR 15 @att
              RETURN
           END
   IF @P4       = 'ESCAPE'
      THEN DO
              IF @EC_FOUND = 'Y' THEN CALL MAP_ERROR 16 @att
              @EC_FOUND = 'Y'
              @EC = STRIP(@P3,B)
              RETURN
           END
   if find(@t5,@p5) = 0          THEN CALL MAP_ERROR 11 @att
   if find(@t6,@p6) = 0          THEN CALL MAP_ERROR 12 @att
   if find(@t7,@p7) = 0          THEN CALL MAP_ERROR 13 @att
   IF @REST    <> ''             THEN CALL MAP_ERROR 34 @att
   @AI = @AI + 1
   @ATT_CHAR.@AI = STRIP(@P3,B)
   @ATTRIBUTE_DEF.@AI = @ATT
   RETURN

SCAN_DYNAMIC_CTLCHARS:
   @PX = SUBWORD(@ATT,3)
   IF @DCL <> 0 THEN CALL MAP_ERROR 23 @att
   DO @I = 1 TO WORDS(@PX)
      IF LENGTH(WORD(@PX,@I)) > 1 THEN CALL MAP_ERROR 24 @att
```

```
      DO @J = 1 TO @DCL
         IF WORD(@PX,@I) = SUBSTR(@DC,@J,1) THEN CALL MAP_ERROR 25 @att
      END
      @DCL = @DCL + 1
      IF @DC = '' THEN @DC = WORD(@PX,@I)
                  ELSE @DC = @DC || WORD(@PX,@I)
   END
   RETURN

EXTRACT_FIELD_SPECS:
   /* TEMPORARY WORK-FIELD CONTAINING ONE LINE OF THE MAP
*/
   @LINEX = @LINE.@L
   /* REPLACE ALL NON-FIELD CHARACTERS BY BLANKS
*/
   DO @C = 1 TO 80
      IF SUBSTR(@LINEX,@C,1) <> @FC THEN
         @LINEX = OVERLAY(' ',@LINEX,@C,1)
   END
   /* EXTRACT PER FIELD THE SPECIFICATIONS
*/
   DO @W = 1 TO WORDS(@LINEX)
      /* SET POINTER TO NEXT FIELD-TABLE-ENTRY
*/
      @F = @F + 1
      @FLL.@F    = @L
      @FCC.@F  = WORDINDEX(@LINEX,@W)
      @FNC.@F = WORDINDEX(@LINEX,@W)
      @FL.@F  = WORDLENGTH(@LINEX,@W)
      @FV.@F   = COPIES(' ',@FL.@F)
   END
   RETURN

CHECK_FIELD_SPECS:
   IF @F  < @FI THEN CALL MAP_ERROR 26 'found' @f 'fields /' @fi 'names'
   IF @F  > @FI THEN CALL MAP_ERROR 27 'found' @f 'fields /' @fi 'names'
   IF @F <> @FH THEN CALL MAP_ERROR 30
   DO @I = 1 TO @F
      DO @J = 1 TO @FI
         IF @FNU.@J = @I
            THEN DO
                    @FN.@I = @FNA.@J
                    @FJ.@I = @FJU.@J
                    @FU.@I = @FUP.@J
                    LEAVE
                 END
      END
```

```
   END
   RETURN

CHECK_FIELD_ATTRIBUTES:
   DO @I = 1 TO @F
      IF @FCC.@I = 1
         THEN CALL MAP_ERROR 31 'Line='@fll.@i'/ column='@fcc.@i
      @XLL = @FLL.@I
      @XCC = @FCC.@I - 1
      @FOUND = 'N'
      DO @J = 1 TO @AI
         IF SUBSTR(@LINE.@XLL,@XCC,1) = @ATT_CHAR.@J
            THEN DO
                    @FOUND = 'Y'
                    LEAVE
                 END
      END
      IF @FOUND = 'N'
         THEN CALL MAP_ERROR 31 'Line='@fll.@i'/ column='@fcc.@i
   END
   RETURN

CHECK_DYNAMIC_CTLCHARS:
   DO @I = 1 TO @DCL
      @FOUND = 'N'
      DO @J = 1 TO @AI
         IF SUBSTR(@DC,@I,1) = @ATT_CHAR.@J
            THEN DO
                    @FOUND = 'Y'
                    LEAVE
                 END
      END
      IF @FOUND = 'N'
         THEN CALL MAP_ERROR 28 'attribute='||substr(@dc,@i,1)
   END
   RETURN

OVERLAY_MAP:
   DO @I = 1 TO @F
      @LI = @FLL.@I
      @CO = @FCC.@I
      @LE = @FL.@I
      @LINE.@LI = OVERLAY(@FV.@I,@LINE.@LI,@CO,@LE)
   END
   RETURN

INSERT_CONTROL_INFO:
```

```
   /* INSERT ESCAPE CHARACTERS AND XEDIT STATEMENTS
*/
   DO @L = 1 TO 24
      @NLINE.@L = ''
      DO @COLUMN  = 1 TO 80
         @FOUND = 'N'
         DO @IA = 1 TO @AI
            IF SUBSTR(@LINE.@L,@COLUMN,1) = @ATT_CHAR.@IA
               THEN DO
                       @FOUND = 'Y'
                       LEAVE
                    END
         END
         IF @FOUND = 'Y'
            THEN CALL INSERT_POS @ATT_CHAR.@IA
            ELSE @NLINE.@L = @NLINE.@L || SUBSTR(@LINE.@L,@COLUMN,1)
      END
      @LINE.@L = @NLINE.@L
   END
   RETURN

INSERT_POS:
   PARSE ARG @CHARACTER
   @NLINE.@L = @NLINE.@L || @EC || @CHARACTER
   DO @IINS = 1 TO @F
      IF @FLL.@IINS   = @L,
       & @FCC.@IINS > @COLUMN
         THEN @FNC.@IINS = @FNC.@IINS + 1
   END
   RETURN

WRITE_MAP_OBJECT:
   QUEUE '/*============================================================*/'
   QUEUE '/* THIS XEDIT MACRO WAS GENERATED BY MAPCOMP ON' DATE(E) '*/'
   QUEUE '/*============================================================*/'
   QUEUE 'PARSE ARG @TYPE @CWF @CWO @ALARM'
   DO @I = 1 TO @F
      QUEUE '@FLL.'@I '=' @FLL.@I||'; @FCC.'@I '=' @FCC.@I||,
            '; @FNC.'@I '=' @FNC.@I||,
            '; @FL.'@I '=' @FL.@I'; @FN.'@I '= "'@FN.@I'"'||,
            '; @FJ.'@I '= "'@FJ.@I'"; @FU.'@I '= "'@FU.@I'"'
   END
   QUEUE 'IF @TYPE = "F" THEN CALL PUSHF'
   QUEUE 'IF @TYPE = "I" THEN CALL PUSHI'
   QUEUE "DO @I = 1 TO 3;  'SET PA'@I 'BEFORE CURSOR HOME'; END"
   QUEUE "DO @I = 1 TO 24; 'SET PF'@I 'BEFORE CURSOR HOME'; END"
   DO @I = 1 TO 24
```

```
   DO @PTRI = 1 TO LENGTH(@LINE.@I)
      @LINOCHAR = SUBSTR(@LINE.@I,@PTRI,1)
      IF @LINOCHAR = '"'
         THEN @LINOCHAR = '""'
      IF @PTRI = 1
         THEN @LINO = @LINOCHAR
         ELSE @LINO = @LINO || @LINOCHAR
   END
   QUEUE '@LINE.'@I  '=' '"'@LINO'"'
END
QUEUE 'IF DATATYPE(@CWF) <> "NUM" | DATATYPE(@CWO) <> "NUM"',
       '| DATATYPE(@CWF,'W') <> 1 | DATATYPE(@CWO,'W') <> 1,'
QUEUE  ' | @CWF < 0 | @CWF >' @F 'THEN DO; @CWF = 0; @CWO = 0; END'
QUEUE 'DO @I =' @F 'TO 1 BY -1'
QUEUE '   PARSE PULL @FA.@I @FJ.@I @FU.@I @FV.@I'
QUEUE '   IF @FJ.@I <> "LEFT" & @FJ.@I <> "RIGHT" &',
      '@FJ.@I <> "CENTRE" & @FJ.@I <> "CENTER" THEN @FJ.@I = "LEFT"'
IF @DCL <> 0
   THEN DO
           QUEUE '   IF DATATYPE(@FA.@I,"W") <> 1 |',
                 '@FA.@I < 0 | @FA.@I >' @DCL 'THEN @FA.@I = 0'
        END
QUEUE '   IF @CWF = @I THEN DO; @CWF = 0;',
         '@CWO = 80 * (@FLL.@I - 1) + @FCC.@I + @CWO - 1;',
         'IF @FLL.@I = 1 THEN @CWO = @CWO + 1; END'
QUEUE 'END'
QUEUE 'DO @I = 1 TO' @F
QUEUE "   INTERPRET '@LINE.'@FLL.@I '= OVERLAY("||,
         "'@FJ.@I'(@FV.@I,@FL.@I),@LINE.'@FLL.@I',"||,
         "@FNC.@I,@FL.@I)'"
IF @DCL <> 0
   THEN DO
           QUEUE "   IF @FA.@I <> 0 THEN DO; @J = @FNC.@I - 1"
           QUEUE "      INTERPRET '@LINE.'@FLL.@I '= OVERLAY("||,
                 "SUBSTR("""@DC""",@FA.@I,1),@LINE.'@FLL.@I',"||,
                 "@J,1)'; END"
        END
QUEUE 'END'
QUEUE 'CMS ''CLRSCRN'';',
      'IF @ALARM = ''ALARM'' THEN ''EMSG'''
QUEUE "'SET LINEND OFF'; 'SET MSGMODE OFF'; 'SET MSGLINE OFF';",
      "'SET SCALE  OFF'; 'SET TABLINE OFF'; 'SET CMDLINE OFF';"
QUEUE "'SET CTLCHAR" @EC "ESCAPE'"
DO @IA = 1 TO @AI
   QUEUE "'" || @ATTRIBUTE_DEF.@IA || "'"
END
QUEUE "'SET SCREEN SIZE 24';",
```

```
      "DO @I = 1 TO 24; 'SET RESERVED' @I 'NOH' @LINE.@I; END"
QUEUE 'DO WHILE @CWO <    0; @CWO = @CWO + 1920; END'
QUEUE 'DO WHILE @CWO > 1919; @CWO = @CWO - 1920; END'
QUEUE '@CWF = @CWO % 80 + 1; @CWO = @CWO // 80 + 1'
QUEUE "'CURSOR SCREEN' @CWF @CWO"
QUEUE 'IF @TYPE = "S" THEN CALL SHOW'
QUEUE 'READ NOCHANGE TAG'
QUEUE 'DO WHILE QUEUED() <> 0'
QUEUE '   PARSE PULL @TKEY @LL @CC @VALUE'
QUEUE '   IF @TKEY = "ETK" THEN @KEY = "ENTER"'
QUEUE '   IF @TKEY = "PFK" THEN @KEY = "PF" || @LL'
QUEUE '   IF @TKEY = "PAK" THEN @KEY = "PA" || @LL'
QUEUE '   IF @TKEY = "RES"'
QUEUE '      THEN DO @I = 1 TO' @F
QUEUE '               IF @FLL.@I <> @LL THEN ITERATE'
QUEUE '               @CFC = @FCC.@I'
QUEUE '               IF @FLL.@I = 1 THEN @CFC = @CFC + 1'
QUEUE '               IF @CFC <> @CC THEN ITERATE'
QUEUE '               IF @FU.@I = "UPPERCASE" THEN UPPER @VALUE'
QUEUE '               @FV.@I = @VALUE'
QUEUE '               LEAVE'
QUEUE '           END'
QUEUE 'END'
QUEUE "'EXTRACT /CURSOR'"
IF @F = 0
   THEN DO
QUEUE '@CRF = 0; @CRO = 80 * (CURSOR.1 - 1) + CURSOR.2 - 1'
        END
   ELSE DO
QUEUE '@CRF = 0'
QUEUE 'DO @I =' @F 'TO 1 BY -1'
QUEUE '   IF @FLL.@I < CURSOR.1',
         '| (@FLL.@I = CURSOR.1 & @FCC.@I ª> CURSOR.2) THEN DO;'
QUEUE '      @CRO = CURSOR.2 - @FCC.@I + 80 * (CURSOR.1 - @FLL.@I);',
           '@CRF = @I; LEAVE; END; END'
QUEUE 'IF @CRF = 0 THEN DO; @CRF = 1;',
        '@CRO = CURSOR.2 - @FCC.1 - 80 * (CURSOR.1 - @FLL.1); END'
QUEUE 'IF @CRF = 1 THEN @CRO = @CRO - 1'
        END
QUEUE 'QUEUE "QQUIT"'
QUEUE 'QUEUE "@KEY = '''||@KEY||"''';',
      '@CURSORRF =" @CRF"; @CURSORRO =" @CRO'
QUEUE 'DO @I = 1 TO' @F ';',
        'QUEUE @FN.@I "= '''||QUOTE(@FV.@I)||"'''"; END'
QUEUE 'EXIT'
QUEUE 'PUSHF:'
QUEUE 'QUEUE "QQUIT"'
```

```
   QUEUE 'DO @I = 1 TO' @F
   QUEUE '   QUEUE @FN.@I||".A = 0;"',
            '@FN.@I||".J =" @FJ.@I||";"',
            '@FN.@I||".U =" @FU.@I||";"',
            '@FN.@I||".C =" @I'
   QUEUE 'END'
   QUEUE 'QUEUE "@CURSORWF = 0; @CURSORWO = 0;',
                '@CURSORRF = 0; @CURSORRO = 0"'
   QUEUE 'EXIT'
   QUEUE 'SHOW:'
   QUEUE "'REFRESH'"
   QUEUE 'QUEUE "QQUIT"'
   QUEUE 'EXIT'
   QUEUE 'PUSHI:'
   QUEUE 'QUEUE "QQUIT"'
   QUEUE 'DO @I = 1 TO' @F
   QUEUE '   QUEUE "QUEUE" @FN.@I||".A"',
            '@FN.@I||".J" @FN.@I||".U" @FN.@I'
   QUEUE 'END'
   QUEUE 'EXIT'
   QUEUE 'QUOTE:'
   QUEUE 'PARSE ARG @QI'
   QUEUE 'IF INDEX(@QI,"''") = 0 THEN RETURN @QI'
   QUEUE '@QO = ''''
   QUEUE 'DO @QX = 1 TO LENGTH(@QI)'
   QUEUE '   IF SUBSTR(@QI,@QX,1) = "''"'
   QUEUE '      THEN @QO = @QO || "''''"'
   QUEUE '      ELSE @QO = @QO || SUBSTR(@QI,@QX,1)'
   QUEUE 'END'
   QUEUE 'RETURN @QO'
   QUEUE ''
  'EXECIO * DISKW' @MAPOBJECT '0 V (FINIS'
   RETURN

READ_MAP_SOURCE:
   @RECORD = ''
   @LINENUM = @LINENUM + 1
  'EXECIO 1 DISKR' @MAPSOURCE @LINENUM '(STRIP'
   @RC = RC
   IF @RC = 0 THEN PARSE PULL @RECORD
   RETURN

MAP_ERROR:
   parse arg @err @extra
   @ERROR.1 = 'Error 1 : input FN (filename) of source not specified'
   @ERROR.2 = 'Error 2 : input FT (filetype) of source not "MAP"'
```

```
@ERROR.3 = 'Error 3 : input file "'@MAPSOURCE'" not found'
@ERROR.4 = 'Error 4 : read "'@MAPSOURCE'" line='@LINENUM 'rc='@rc
@ERROR.5 = 'Error 5 : read "'@MAPSOURCE'" line='@LINENUM 'rc='@rc
@ERROR.6 = 'Error 6 : statement "SET CTLCHAR ESCAPE" missing'
@ERROR.7 = 'Error 7 : first word on attribute line not "SET"'
@ERROR.8 = 'Error 8 : second word in "SET" statement not recognized'
@ERROR.9 = 'Error 9 : length of attribute char "'@P3'" exceeds 1'
@ERROR.10 = 'Error 10 : invalid keyword "'@P4'" specified'
@ERROR.11 = 'Error 11 : invalid COLOR "'@P5'" specified'
@ERROR.12 = 'Error 12 : invalid EXTHI "'@P6'" specified'
@ERROR.13 = 'Error 13 : invalid ATTRIBUTE "'@P7'" specified'
@ERROR.14 = 'Error 14 : duplicate "SET CTLCHAR FIELD" found'
@ERROR.15 = 'Error 15 : invalid "SET CTLCHAR FIELD" found'
@ERROR.16 = 'Error 15 : duplicate "SET CTLCHAR ESCAPE"',
            'statement found'
@ERROR.17 = 'Error 17 : incorrect field number',
            'in "SET FIELD" statement'
@ERROR.18 = 'Error 18 : invalid name "'@P5'"',
            'in "SET FIELD" statement'
@ERROR.19 = 'Error 19 : "JUSTIFY" option expected',
            'in "SET FIELD" statement'
@ERROR.20 = 'Error 20 : incorrect option JUSTIFY "'@P7'"',
            'in "SET FIELD" statement'
@ERROR.21 = 'Error 21 : "SET FIELD" statement references field' @P3,
            'again'
@ERROR.22 = 'Error 22 : "SET FIELD" statement followed by rubbish'
@ERROR.23 = 'Error 23 : more than 1 "SET DYNAMIC" statement'
@ERROR.24 = 'Error 24 : CTLCHAR longer than 1 in "SET DYNAMIC"',
            'statement'
@ERROR.25 = 'Error 25 : duplicate CTLCHAR found in "SET DYNAMIC"',
            'statement'
@ERROR.26 = 'Error 26 : more names defined than total number of',
            'fields in the MAP'
@ERROR.27 = 'Error 27 : less names defined than total number of',
            'fields in the MAP'
@ERROR.28 = 'Error 28 : ctlchar in "SET DYNAMIC" statement',
            'not defined in "SET CTLCHAR" statement'
@ERROR.29 = 'Error 29 : keyword "NAME" not found in',
            '"SET FIELD" statement'
@ERROR.30 = 'Error 30 : field numbers out of range,',
            'use "MAPRENUM" XEDIT macro in source'
@ERROR.31 = 'Error 31 : field in map not preceeded by ctlchar'
@ERROR.32 = 'Error 32 : CTLCHAR " not allowed in "SET CTLCHAR"'
@ERROR.33 = 'Error 33 : no UPPERCASE / LOWERCASE option in',
            '"SET FIELD" statement'
@ERROR.34 = 'Error 34 : "SET CTLCHAR" statement followed by rubbish'
@ERROR.35 = 'Error 35 : "SET FIELD" statement followed by rubbish'
```

```
   SAY @ERROR.@ERR
   if @extra <> ''
      then say @extra
   say 'Please check the error and recompile map after correction'
   say 'Compilation aborted now'
   EXIT

/*====================================================================*
/
/* THIS COPY-BOOK CAN BE USED IN A REXX EXEC TO CALL A FULL SCREEN
*/
/*====================================================================*
/

INIT_MAP:
   PARSE ARG @MAPNAME
  'DESBUF'
   INTERPRET 'QUEUE "MACRO' @MAPNAME 'F"'
  'XEDIT MAPCOPY FILE X (NOPROFIL'
   DO WHILE QUEUED() <> 0; PARSE PULL @Q@; INTERPRET @Q@; END
   RETURN

WRITE_MAP:
   PARSE ARG @MAPNAME
  'DESBUF'
   INTERPRET 'QUEUE "MACRO' @MAPNAME 'I"'
  'XEDIT MAPCOPY FILE X (NOPROFIL'
   DO @Q@ = 1 TO QUEUED(); PARSE PULL @I@.@Q@; END
   INTERPRET 'QUEUE "MACRO' @MAPNAME 'D' @CURSORWF @CURSORWO @ALARM '"'
   DO @Q@ = (@Q@-1) TO 1 BY -1; INTERPRET @I@.@Q@; END
  'XEDIT MAPCOPY FILE X (NOPROFIL'
   DO WHILE QUEUED() <> 0; PARSE PULL @Q@; INTERPRET @Q@; END
   RETURN
MAPDISPL.RXX

/*====================================================================*
/
/* THIS EXEC DISPLAYS A PROTOTYPE OF A FULL-SCREEN MAP XEDIT MACRO
*/
/*      WHICH WAS GENERATED BY MAPCOMP
*/
/*    A.P. VAN WINGERDEN                    VERSION V-1-7
*/
/*====================================================================*
/
   PARSE ARG @MAPFN @MAPFT @MAPFM
   CALL INIT_MAP(@MAPFN)
```

```
   @ALARM  = 'ALARM'
   CALL WRITE_MAP(@MAPFN)
   RETURN

/*=================================================================*
/
/* THIS COPY-BOOK CAN BE USED IN A REXX EXEC TO CALL A FULL SCREEN
*/
/*=================================================================*
/

INIT_MAP:
   PARSE ARG @MAPNAME
  'DESBUF'
   INTERPRET 'QUEUE "MACRO' @MAPNAME 'F"'
  'XEDIT MAPCOPY FILE X (NOPROFIL'
   DO WHILE QUEUED() <> 0; PARSE PULL @Q@; INTERPRET @Q@; END
   RETURN

WRITE_MAP:
   PARSE ARG @MAPNAME
  'DESBUF'
   INTERPRET 'QUEUE "MACRO' @MAPNAME 'I"'
  'XEDIT MAPCOPY FILE X (NOPROFIL'
   DO @Q@ = 1 TO QUEUED(); PARSE PULL @I@.@Q@; END
   INTERPRET 'QUEUE "MACRO' @MAPNAME 'D' @CURSORWF @CURSORWO @ALARM '"'
   DO @Q@ = (@Q@-1) TO 1 BY -1; INTERPRET @I@.@Q@; END
  'XEDIT MAPCOPY FILE X (NOPROFIL'
   DO WHILE QUEUED() <> 0; PARSE PULL @Q@; INTERPRET @Q@; END
   RETURN
MAPRENUM.XED

/*=================================================================*
/
/* MAPRENUM : ASSIST FULL-SCREEN DEVELOPEMENT FOR MAPCOMP SCREENS
*/
/*              A.P. VAN WINGERDEN
*/
/*=================================================================*
/
/* FUNCTIONS :
*/
/*    1. CHECK WHETHER THE FILETYPE OF THE FILE BEING EDITED IS 'MAP'
*/
/*    2. THE FOLLOWING LINES DESCRIBE THE FUNCTIONS TO BE EXECUTED FOR
*/
/*       THE LINES FROM 25 AND ONWARDS IN THE FILE
```

```
*/
/*    3. RENUMBER ALL THE FIELDS IN THE CORRECT ORDER (SET FIELD)
*/
/*    4. TRANSLATE ALL LINES TO UPPERCASE
*/
/*=====================================================================*
/

   COMMAND 'EXTRACT /FTYPE'
   @FT = STRIP(FTYPE.1,B)
   IF @FT <> 'MAP'
      THEN DO
              COMMAND 'EMSG INVALID FILETYPE "'@FT'"; SHOULD BE "MAP"'
              RETURN
           END

   COMMAND 'PRESERVE'
   COMMAND 'SET LINEND OFF'
   COMMAND 'EXTRACT /SIZE'
   COMMAND 'SET IMAGE ON'
   COMMAND 'EXTRACT /LINE'
   COMMAND 'TOP'

   @FIELDNUM = 0
   DO @I = 25 TO SIZE.1
      COMMAND ':'@I
      COMMAND 'EXTRACT /CURLINE'
      PARSE UPPER VAR CURLINE.3 @RECORD
      IF WORD(@RECORD,1) = 'SET',
       & WORD(@RECORD,2) = 'FIELD'
         THEN DO
                @FIELDNUM = @FIELDNUM + 1
                @RECORD = WORD(@RECORD,1) WORD(@RECORD,2),
                          RIGHT(@FIELDNUM,3,0),
                          SUBWORD(@RECORD,4)
              END
      COMMAND 'REPLACE' @RECORD
   END

   COMMAND 'RESTORE'
   COMMAND ':25'
   COMMAND 'EMSG FIELD NUMBERS ADJUSTED',
               'AND LINES 25 THRU' SIZE.1 'TRANSLATED TO UPPERCASE'

   RETURN
MAPCOPY.COP
```

```
/*=================================================================*
/
/* THIS COPY-BOOK CAN BE USED IN A REXX EXEC TO CALL A FULL SCREEN
*/
/*=================================================================*
/

INIT_MAP:
   ARG @MAPNAME .
  'DESBUF'
   INTERPRET 'QUEUE "MACRO' @MAPNAME 'F"'
  'XEDIT MAPCOPY FILE X (NOPROFIL'
   DO WHILE QUEUED() <> 0; PARSE PULL @Q@; INTERPRET @Q@; END
   RETURN

WRITE_MAP:
   ARG @MAPNAME @MAPWAIT .
   SELECT
      WHEN ABBREV('WAIT',@MAPWAIT,1) = 1   THEN _@ = 'D'
      WHEN ABBREV('NOWAIT',@MAPWAIT,1) = 1 THEN _@ = 'S'
      OTHERWISE _@ = 'D'
   END
  'DESBUF'
   INTERPRET 'QUEUE "MACRO' @MAPNAME 'I"'
  'XEDIT MAPCOPY FILE X (NOPROFIL'
   DO @Q@ = 1 TO QUEUED(); PARSE PULL @I@.@Q@; END
   INTERPRET 'QUEUE "MACRO' @MAPNAME _@ @CURSORWF @CURSORWO @ALARM '"'
   DO @Q@ = (@Q@-1) TO 1 BY -1; INTERPRET @I@.@Q@; END
  'XEDIT MAPCOPY FILE X (NOPROFIL'
   DO WHILE QUEUED() <> 0; PARSE PULL @Q@; INTERPRET @Q@; END
   RETURN
MAPCOPY.FIL

/*=================================================================*
/
/* THIS FILE IS NEEDED BY THE MAPCOMP UTILITY, WHEN USING MAPCOPY COPY
*/
/*=================================================================*
/
```

MAPEXAMP.RXX

```
/*=================================================================*
/
```

```
/* MAPEXAMP : example for MAPCOMP usage
*/
/*=============================================================================*
/

   call init_map 'mapexamp' /* Initialize the map
*/

   @alarm    = 'NOALARM'    /* No BEEP
*/
   @msg      = ''           /* Wipe error message
*/
   @pfk      = 'none'       /* First display no function key used
*/
   @rf       = '????'       /* Cursor first time unavailable
*/
   @ro       = '????'       /* Cursor first time unavailable
*/
   @qpfk     = 'PF3'        /* Assign QUIT to PF3 per default
*/
   @fld      = 'centred'    /* Assign default text to text field
*/
   @fld.A    = 9            /* Political centre is BLUE
*/
   @jfld     = 'C'          /* Default centred display
*/
   @fld.J    = 'CENTRE'     /* Put justification to work
*/
   @cmd      = ''           /* No command supplied yet
*/
   @att      = 0            /* Default attribute to be used
*/
   @maxres   = 5            /* Arraysize on screen (result set)
*/
   do @i = 1 to @maxres     /* Wipe result set of command
*/
      @res.@i = ''
   end
   @cursorwf = @qpfk.c      /* Put cursor on field @qpfk
*/
   @cursorwo = 0            /*     right at the beginning
*/

   do forever
      call write_map 'mapexamp' /* Write the map (WAIT is default)
*/
      if @key = @qpfk           /* Was key assigned to QUIT ?
```

```
*/
      then leave             /*     if yes, then leave main loop
*/
   @msg      = 'message line'/* Reset error message
*/
   @alarm    = 'NOALARM'     /* Reset alarm
*/
   @pfk      = @key          /* Display function key used by user
*/
   @rf       = @cursorrf     /* Display cursorrf
*/
   @ro       = @cursorro     /* Display cursorro
*/
   @cursorwf = @cursorrf     /* Keep cursor in same position
*/
   @cursorwo = @cursorro     /* Keep cursor in same position
*/
   @badpf = 1
   do @i = 1 to 24
      if @qpfk <> 'PF'||@i
         then iterate
      @badpf = 0
      leave
   end
   if @badpf
      then do
              @msg      = 'Bad key - must be PF1 .. PF24 !'
              @alarm    = 'ALARM'
              @cursorwf = @qpfk.C
              @cursorwo = 0
              iterate
           end
   select
      when @jfld = 'L'
           then do
                  @fld.A = 8
                  @fld.J = 'LEFT'
                  @fld   = 'justified to the left'
                end
      when @jfld = 'C'
           then do
                  @fld.A = 9
                  @fld.J = 'CENTRE'
                  @fld   = 'justified to the centre'
                end
      when @jfld = 'R'
           then do
```

```
                    @fld.A = 10
                    @fld.J = 'RIGHT'
                    @fld   = 'justified to the right'
                 end
         otherwise
              @msg      = 'Wrong justification specified !'
              @alarm    = 'ALARM'
              @cursorwf = @jfld.C
              @cursorwo = 0
              iterate
      end
      if bad_command()
         then iterate
   end

   exit

bad_command:
   do @i = 1 to @maxres
      @res.@i = ''
   end
   if @cmd = ''
      then return 0
   parse var @cmd @type .
   if @type <> 'CP' & @type <> 'CMS'
      then do
              @msg      = 'No CP or CMS specified as first word in',
                          'the command'
              @alarm    = 'ALARM'
              @cursorwf = @cmd.C
              @cursorwo = 0
              return 1
           end
  'PIPE' @cmd '| STEM @RES.'
   @msg = 'Rc='rc 'from command :' @cmd
   if rc <> 0
      then @alarm = 'ALARM'
   if @att <> '' & @att >= '0' & @att <= 10
      then do
              do @i = 1 to @maxres
                 @res.@i.A = @att
              end
           end
```

```
else do
              @msg      = 'Bad attribute number specified (not 0 .. 10)'
              @alarm    = 'ALARM'
              @cursorwf = @att.C
              @cursorwo = 0
              return 1
           end
   return 0

/*═══════════════════════════════════════════════════════════════════════*
/
/* THIS COPY-BOOK CAN BE USED IN A REXX EXEC TO CALL A FULL SCREEN
*/
/*═══════════════════════════════════════════════════════════════════════*
/

INIT_MAP:
   ARG @MAPNAME .
  'DESBUF'
   INTERPRET 'QUEUE "MACRO' @MAPNAME 'F"'
  'XEDIT MAPCOPY FILE X (NOPROFIL'
   DO WHILE QUEUED() <> 0; PARSE PULL @Q@; INTERPRET @Q@; END
   RETURN

WRITE_MAP:
   ARG @MAPNAME @MAPWAIT .
   SELECT
      WHEN ABBREV('WAIT',@MAPWAIT,1) = 1   THEN _@ = 'D'
      WHEN ABBREV('NOWAIT',@MAPWAIT,1) = 1 THEN _@ = 'S'
      OTHERWISE _@ = 'D'
   END
  'DESBUF'
   INTERPRET 'QUEUE "MACRO' @MAPNAME 'I"'
  'XEDIT MAPCOPY FILE X (NOPROFIL'
   DO @Q@ = 1 TO QUEUED(); PARSE PULL @I@.@Q@; END
   INTERPRET 'QUEUE "MACRO' @MAPNAME _@ @CURSORWF @CURSORWO @ALARM '"'
   DO @Q@ = (@Q@-1) TO 1 BY -1; INTERPRET @I@.@Q@; END
  'XEDIT MAPCOPY FILE X (NOPROFIL'
   DO WHILE QUEUED() <> 0; PARSE PULL @Q@; INTERPRET @Q@; END
   RETURN
```

```
MAPEXAMP.MAP

/?????????????????????????????????????????????????????????????????????????????
????????

%You used function key :_?????%
%CURSORRF read was     :_????%
%CURSORRO read was     :_????%


!Fill in the name of a function key to QUIT the map :$????%

%You may change the justification of the following text :
/>\??????????????????????????????????????????????????????????????????????/<%
%Use justification :$?% (Choose R, L or C for Right, Left or Centre)

%Fill in a CMS or CP (e.g. query) command; the first 5 lines of the
%result set will be displayed below.!Your command :$??????????????????????
%(Please prefix your command with either CMS or CP)
*????????????????????????????????????????????????????????????????????????????
*????????????????????????????????????????????????????????????????????????????
*????????????????????????????????????????????????????????????????????????????
*????????????????????????????????????????????????????????????????????????????
*????????????????????????????????????????????????????????????????????????????
%You may change the attribute for the output above.!Choose a number in
!the range 0 .. 10 (corresponding to SET CTLCHAR DYNAMIC) :$??%

SET CTLCHAR ` ESCAPE
SET CTLCHAR ? FIELD
SET CTLCHAR / PROTECT   PINK   REVVIDEO HIGH
SET CTLCHAR $ NOPROTECT RED    REVVIDEO HIGH
SET CTLCHAR % PROTECT   GREEN  NONE     NOHIGH
SET CTLCHAR ! PROTECT   BLUE   NONE     HIGH
SET CTLCHAR _ PROTECT   WHITE  BLINK    HIGH
SET CTLCHAR \ PROTECT   WHITE  REVVIDEO HIGH
SET CTLCHAR * PROTECT   YELLOW BLINK    HIGH
SET CTLCHAR @ PROTECT   RED    REVVIDEO HIGH
SET CTLCHAR - PROTECT   BLUE   REVVIDEO HIGH
SET CTLCHAR # PROTECT   GREEN  REVVIDEO HIGH
SET DYNAMIC / $ % ! _ \ * @ - #
SET FIELD 001 NAME @MSG     JUSTIFY LEFT LOWERCASE
```

```
SET FIELD 002 NAME @PFK     JUSTIFY LEFT UPPERCASE
SET FIELD 003 NAME @RF      JUSTIFY LEFT UPPERCASE
SET FIELD 004 NAME @RO      JUSTIFY LEFT UPPERCASE
SET FIELD 005 NAME @QPFK    JUSTIFY LEFT UPPERCASE
SET FIELD 006 NAME @FLD     JUSTIFY LEFT LOWERCASE
SET FIELD 007 NAME @JFLD    JUSTIFY LEFT UPPERCASE
SET FIELD 008 NAME @CMD     JUSTIFY LEFT UPPERCASE
SET FIELD 009 NAME @RES.1   JUSTIFY LEFT UPPERCASE
SET FIELD 010 NAME @RES.2   JUSTIFY LEFT UPPERCASE
SET FIELD 011 NAME @RES.3   JUSTIFY LEFT UPPERCASE
SET FIELD 012 NAME @RES.4   JUSTIFY LEFT UPPERCASE
SET FIELD 013 NAME @RES.5   JUSTIFY LEFT UPPERCASE
SET FIELD 014 NAME @ATT     JUSTIFY LEFT UPPERCASE
```

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

A full screen console interface – part 18

Editor's note: the following article is the conclusion of an extensive piece of work which has been published over several issues of VM Update. It was felt that readers could benefit from the entire article and from the individual sections. Any comments or recommendations would be welcomed and should be addressed either to SDS or directly to the author at feranado_durante@vnet.ibm.com.

CONNECT HELPCSC

```
.cm VM Software Services
.cm
.cs Ø on
[]CSC Tool[%
[%
[%
Use the CONNECT command to establish a remote session with another
node.
EXAMPLE: CONNECT VM2
Start a session with CSC Service Machine on VM2.
.cs Ø off
.cs 1 on
[]CSC Tool[%
[]Purpose[%
Use the CONNECT command to establish a remote session with another
node.
.cs 1 off
.cs 2 on
[]Format[%
>>─COnnect─node──────────────────────────────><
[]Class[%
7
.cs 2 off
.cs 3 on
[]Operands[%
```

```
node
destination node where to start a remote session.
.cs 3 off
.cs 4 on.2
.cs 4 off
.cs 5 on
[]Usage Notes[%
1. Experimental code...
.cs 5 off
.cs 6 on
[]Messages[%
Ø31ØE Missing operand(s)
ØxxxE ...
.cs 6 off
```

CSCSVP HELPCSC

```
.cm VM Software Services
.cm
.cs Ø on
[]CSC Tool[%
[%
[%
The CSCSVP command start a CSC service machine.
EXAMPLE: CSCSVP
Start CSC.
.cs Ø off
.cs 1 on
[]CSC Tool[%
[]Purpose[%
The CSCSVP command start a CSC service machine.
.cs 1 off
.cs 2 on
[]Format[%
>>─CSCSVP──────────────────────────────><
.cs 2 off
.cs 3 on
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%.3
1. Directory options...
.cs 5 off
.cs 6 on
[]Messages and Return Codes[%
ØxxxE Invalid operand: operand
[]Return Codes[%
```

```
Ø Normal
.cs 6 off
.cs 7 on
.cs 7 off
```

CSCUSR HELPCSC

```
.cm VM Software Services
.cm
.cs Ø on
[]CSC Tool[%
[%
[%
```

The CSCUSR command establishes a session with a CSC service machine.

EXAMPLE: CSCUSR ABC

Start a session with CSC Service Machine ABC.

```
.cs Ø off
.cs 1 on
[]CSC Tool[%
[]Purpose[%
The CSCUSR command establishes a session with a CSC service machine.
.cs 1 off
.cs 2 on
[]Format[%
>>—CSCUSR—.———.—.————————————.—————————><
‘-target-' ‘-(—.———.—.———.—.—.-'
‘-NOALT-' ‘-NOEDS-' ‘-)-'
.cs 2 off
.cs 3 on
[]Operands[%
target
Identifies the name of the CSC Service Machine..4
.cs 3 off
.cs 4 on
[]Options[%
NOALT
Do not use alternate screen size if available.
NOEDS
Do not use Extended Data Streams (colours) if supported.
.cs 4 off
```

```
.cs 5 on
[]Usage Notes[%
```

1. The CSCUSR can be renamed to the name of the CSC Service

Machine.

2. If target is omitted, it defaults to the name of the module.

3. Default session values can be changed by using a profile. This

file may contain any valid CSCUSR commands and will be executed

before the first screen is displayed. It must have a filetype

of $PROFILE and the same name as this module (default CSCUSR).

Blank lines and records beginning with an asterisk (*) are

ignored.

```
.cs 5 off
.cs 6 on
[]Messages and Return Codes[%
Ø22ØE Invalid operand: operand (RC=8)
Ø222E Invalid option: option (RC=8)
Ø23ØE Program program already active
Ø231E Error executing HNDIUCV SET. Return code is code
Ø24ØE CSC Service machine ‘id' is not logged on
Ø241E CSC Servive machine ‘id' is not initialized
Ø242E Maximum connections for ‘id' CSC Service machine exceeded
Ø243E You are not authorized to connect to CSC Service machine ‘id'
Ø244E Invalid IUCV System Service name ‘name'
Ø249E Unexpected error from CMSIUCV CONNECT. Return code is code
Ø25ØE Cannot initialize console. Return code from CONSOLE OPEN is code
Ø252E Cannot restore console. Return code from CONSOLE CLOSE is code
Ø254E Cannot clear IUCV handler. Ret code from CMSIUCV SEVER is code
Ø256E Unable to terminate IUCV session
Ø271E IUCV connection severed by the CSC Service ma
chine (RC=1Ø1) Ø26ØE Console read error. Return code from CONSOLE READ
is code
Ø262E Invalid destination
Ø27ØE Unable to connect to the CSC Service Machine (RC=1ØØ)
```

```
Ø279E Unexpected interrupt from IUCV
Ø280E IUCV Receive buffer too small
Ø289E IUCV Receive error. IPRCODE is code
Ø290E IUCV Send error. IPRCODE is code.5
Ø292E Console I/O error. Return code from CONSOLE WRITE is code
[]Return Codes[%
Ø Normal
8 Invalid operand or option
>Ø Return codes from other CMS commands
.cs 6 off
.cs 7 on
.cs 7 off
```

CURRENT HELPCSC

```
.cm VM Software Services
.cm
.cs Ø on
[]CSC Tool[%
[%
[%
```

Use the CURRENT command to show the current screen of data.

EXAMPLE: CURRENT

Show and refresh current screen of data.

```
.cs Ø off
.cs 1 on
[]CSC Tool[%
[]Purpose[%
Use the CURRENT command to show the current screen of data.
.cs 1 off
.cs 2 on
[]Format[%
>>—CURrent———————————————————————><
[]Class[%
2
.cs 2 off
.cs 3 on
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%.6
```

1. PFØ9 and PF21 are assigned to the CURRENT command.

**2. Unlike the BOTTOM command, messages with the NODISPLAY attribute
are never shown, messages with the HOLD attribute are not
scrolled, and the screen is refreshed whenever a new message is
received by the service machine.**

```
.cs 5 off
.cs 6 on
[]Messages[%
Ø312E Unexpected CURRENT operand: operand
.cs 6 off
```

DISCONN HELPCSC

```
.cm VM Software Services
.cm
.cs Ø on
[]CSC Tool[%
[%
[%
```

**Use the DISCONNECT command to terminate the last established remote
session.
EXAMPLE: DISCONNECT
Terminate remote session.**

```
.cs Ø off
.cs 1 on
[]CSC Tool[%
[]Purpose[%
Use the DISCONNECT command to terminate the last established remote
session.
.cs 1 off
.cs 2 on
[]Format[%
>>—DIsconnect———————————————————————><
[]Class[%
7
.cs 2 off
.cs 3 on.7
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%
1. Experimental code...
```

```
.cs 5 off
.cs 6 on
[]Messages[%
Ø311E Invalid DISCONNECT operand: operand
ØxxxE ...
.cs 6 off
```

DLOCATE HELPCSC

```
.cm VM Software Services
.cm
.cs Ø on
[]CSC Tool[%
[%
[%
Use the DLOCATE command to search forward the CSC log file for a
string.
```
EXAMPLE: DLOCATE /ABC
Search forward the log file and locate string ABC.
```
.cs Ø off
.cs 1 on
[]CSC Tool[%
[]Purpose[%
Use the DLOCATE command to search forward the CSC log file for a
string.
.cs 1 off
.cs 2 on
[]Format[%
>>—.-DLocate——.—string————————————————————><
‘-DOWNLocate-'
[]Class[%
4.8
.cs 2 off
.cs 3 on
[]Operands[%
string
is the data string to locate. The first byte is the string
delimiter.
The string can be from 1 to 36 characters long.
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%
```

**1. The search starts at the first data line on the screen and
proceeds forward. The first line found containing the search
string will become the new top line.
2. The closing delimiter is optional. DLOCATE /ABC/ and DLOCATE /
ABC are equivalent commands.
4. The string delimiter may appear in the string it-self. /A*B/ and
*A*B* are equivalent strings.
5. DOWNLOCATE is a synonym for DLOCATE.**

```
.cs 5 off
.cs 6 on
[]Messages[%
Ø310E Missing operand(s)
Ø350E String too long. Must be from 1 to 34 characters
Ø351I String not found
.cs 6 off
```

DMATCH HELPCSC

```
.cm VM Software Services
.cm
.cs Ø on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[%
```
**[%
Use the DMATCH command to search forward the CSC log file for a
character mask.
EXAMPLE: DMATCH ABC*.**9
Search forward the log file for record beginning with ABC.
```
.cs Ø off
.cs 1 on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[]Purpose[%
Use the DMATCH command to search forward the CSC log file for a
character mask.
.cs 1 off
.cs 2 on
```

```
[]Format[%
>>—.-DMatch——.—mask————————————————————><
‘-DOWNMatch-'
[]Class[%
4
.cs 2 off
.cs 3 on
[]Operands[%
mask
is the data mask to use. It can be from 1 to 36 characters long.
Two characters have a special meaning in a mask, * (asterisk) and
% (percent).
*
represents any number of characters.
%
represents one single character.
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
```

[]Usage Notes[%
1. The DMATCH command expects one single space between the command
name, or a valid abbreviation, and the mask. DMATCH *ABC* and
DMATCH *ABC* are not equivalent. The latter will only locate a
record with a blank as the first character.
2. DOWNMATCH is a synonym for DMATCH..10

```
.cs 5 off
.cs 6 on
[]Messages[%
Ø31ØE Missing operand(s)
Ø35ØE String too long. Must be from 1 to 36 characters
Ø351I String not found
.cs 6 off
```

DOWN HELPCSC

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

```
[%
[%
```
Use the DOWN command to scroll forward the CSC log file.

EXAMPLE: DOWN 3
Scroll forward 3 lines.

```
.cs Ø off
.cs 1 on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[]Purpose[%
Use the DOWN command to scroll forward the CSC log file.
.cs 1 off
.cs 2 on
[]Format[%
.-1-.
>>—.-Down-.—┼—┼———————————————————————><
‘-Next-' ‘-n-'
[]Class[%
3
.cs 2 off
.cs 3 on
[]Operands[%
n.11
number of lines to scroll. The default is one.
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%
1. NEXT is a synonym for DOWN.
.cs 5 off
.cs 6 on
[]Messages[%
Ø311E Invalid DOWN operand: operand
Ø312E Unexpected DOWN operand: operand
.cs 6 off
END HELPCSC
.cm VM Software Services
.cm
.cs Ø on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[%
[%
```

Use the END command to teminate a session with a CSC Service Machine.
EXAMPLE: END
Terminate the current session with CSC.

```
.cs Ø off
.cs 1 on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[]Purpose[%
Use the END command to teminate a session with a CSC Service Machine.
.cs 1 off
.cs 2 on
[]Format[%
>>─END──────────────────────────────────><
[]Class[%.12
Ø
.cs 2 off
.cs 3 on
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%
```

1. PFØ3 and PF15 perform a special form of the END command where all sessions are terminated and no messages are displayed.
2. If you are connected to a remote node, only the last activated session is terminated.
3. When the last active session is terminated, message Ø2Ø1E is displayed and the program ends with a return code 1Ø1.

```
.cs 5 off
.cs 6 on
```

[]Messages and Return Codes[%
Ø251E IUCV connection severed by the CSC Service machine (RC=1Ø1)
Ø312E Unexpected END operand: operand

```
.cs 6 off
```

EXCLUDE HELPCSC

```
.cm VM Software Services
.cm
```

```
.cs Ø on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[%
[%
Use the EXCLUDE command to control the messages to display.
EXAMPLE: EXCLUDE AB
Do not display messages with prefix A or B.
.cs Ø off
.cs 1 on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[]Purpose[%
Use the EXCLUDE command to control the messages to display..13
.cs 1 off
.cs 2 on
[]Format[%
.<——<.
>>—Exclude—+————+————————————————————><
‘-prefix-'
[]Class[%
Ø
.cs 2 off
.cs 3 on
[]Operands[%
prefix
one or more prefixes to select. Spaces separating prefixes are
optional.
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%
1. INCLUDE and EXCLUDE commands may be used together to change the
messages to display.
.cs 5 off
.cs 6 on
[]Messages[%
Ø33ØE EXCLUDE did not process prefix “prefix” successfully
Ø331W Prefix “prefix” not defined. Ignored
.cs 6 off
```

FORWARD HELPCSC

```
.cm VM Software Services
.cm
.cs Ø on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[%
[%
Use the FORWARD command to scroll forward the CSC log file.
EXAMPLE: FORWARD 3.14
Scroll forward 3 screens of data.
.cs Ø off
.cs 1 on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[]Purpose[%
Use the FORWARD command to scroll forward the CSC log file.
.cs 1 off
.cs 2 on
[]Format[%
.-1-.
>>—.-Forward-.—┼—┼————————————————————><
‘-FWD——' ‘-n-'
[]Class[%
3
.cs 2 off
.cs 3 on
[]Operands[%
n
number of screens to scroll. The default is one.
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%
1. PFØ8 and PF2Ø perform the command FORWARD 1.
2. FWD is a synonym for FORWARD.
.cs 5 off
.cs 6 on
[]Messages[%
Ø311E Invalid FORWARD operand: operand
Ø312E Unexpected FORWARD operand: operand
.cs 6 off
```

FWD HELPCSC.15

```
.cm VM Software Services
.cm
```

```
.cs Ø on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[%
[%
Use the FWD command to scroll forward the CSC log file.
EXAMPLE: FWD 3
Scroll forward 3 screens of data.
.cs Ø off
.cs 1 on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[]Purpose[%
Use the FWD command to scroll forward the CSC log file.
.cs 1 off
.cs 2 on
[]Format[%
.-1-.
>>─FWD─┼─┼──────────────────────────────><
'-n-'
[]Class[%
3
.cs 2 off
.cs 3 on
[]Operands[%
n
number of screens to scroll. The default is one.
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%
1. PFØ8 and PF2Ø perform the command FWD 1.
2. FWD is a synonym for FORWARD..16
.cs 5 off
.cs 6 on
[]Messages[%
Ø311E Invalid FWD operand: operand
Ø312E Unexpected FWD operand: operand
.cs 6 off
```

GO HELPCSC

.cs 3 off
.cs 4 on

.cs 4 off
.cs 5 on
[]Usage Notes[%
1. The time format hh: is only required when the date .cm VM Software Services
.cm
.cs Ø on

[]CSC Tool[%
[%
[%
Use the GO command to locate a CSC log file record based on its date or time.
EXAMPLE: GO 12:ØØ:ØØ
Locate first record created today with a timestamp of
12:ØØ:ØØ or greater.
.cs Ø off
.cs 1 on

[]CSC Tool[%
[]Purpose[%
Use the GO command to locate a CSC log file record based on its date or time.
.cs 1 off
.cs 2 on
[]Format[%
>>—.-Go—.————.—.————.——————————————><
] (1)]]-hh———]
]-dd———]]-hh:———] (2)
]-mm/dd——]]-hh:mm——]
‘-yy/mm/dd-' ‘-hh:mm:ss-'
Notes: (1) At least one operand is required.
(2) The format hh: is required only when the date is omitted.
[]Class[%.17
4
.cs 2 off
.cs 3 on
[]Operands[%
yy/mm/dd
reference date to locate. The default is today's date.
The abbreviated forms dd and mm/dd assume that the reference date is
on the same year and month as today's date.
hh:mm:ss
reference time to locate. The abbreviated forms hh and hh:mm uses
zero for the missing values. 12:Ø3 is equivalent to 12:Ø3:ØØ.
is omitted.

**To locate the first record created today after 12:ØØ:ØØ enter
GO 12:. The command GO 12 will locate the first record created
on the 12th day (or after) of this month.
2. One digit values are accepted for both date and time. The
command GO 97/6/23 8:12:5 is valid.
3. Most of the times, GO is much faster than LOCATE or MATCH to
find a record in the log file.
.cs 5 off
.cs 6 on**

```
[]Messages[%
Ø31ØE Missing operand(s)
Ø311E Invalid GO operand: operand
Ø312E Unexpected GO operand: operand
Ø352E Value value is too long for Date or Time values
Ø353E Invalid Date entered
Ø354E Invalid Time entered
Ø355E You cannot locate a future Date/Time
Ø356I No record found after specified Date/Time
.cs 6 off
```

HELP HELPCSC

```
.cm VM Software Services.18
.cm
.cs Ø on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[%
[%
Use the HELP command to display CSC Help files.
EXAMPLE: HELP SWITCH
Display SWITCH Help file.
.cs Ø off
.cs 1 on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[]Purpose[%
Use the HELP command to display CSC Help files.
.cs 1 off
.cs 2 on
[]Format[%
>>— Help— .— — — — — — .— — — — — — — — — — — — — — — — — — — — — —
— ><
```

```
‘-CSC-command-'
[]Class[%
Ø
.cs 2 off
.cs 3 on
[]Operands[%
CSC command
any CSC command. If omitted the CSC Help menu is displayed.
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%
1. PFØ1 and PF13 are assigned to the HELP command.
2. This is a local command, executed by the user program (CSCUSR).
The service machine is not informed.
3. Use the CMS command to display non CSC help files..19
Example: CMS HELP RENAME will display the help for CMS RENAME.
.cs 5 off
.cs 6 on
.cs 6 off
```

INCLUDE HELPCSC

```
.cm VM Software Services
.cm
.cs Ø on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[%
[%
Use the INCLUDE command to control the messages to display.
EXAMPLE: INCLUDE AB
Display only messages with prefix A or B.
.cs Ø off
.cs 1 on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[]Purpose[%
Use the INCLUDE command to control the messages to display.
.cs 1 off
.cs 2 on
[]Format[%
.<——<.
>>—Include—+————+————————————————————><
‘-prefix-'
[]Class[%
```

```
Ø
.cs 2 off
.cs 3 on
[]Operands[%
prefix
one or more prefixes to select. Spaces separating prefixes are
optional.
.cs 3 off.20
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%
1. INCLUDE and EXCLUDE commands may be used together to change the
messages to display.
.cs 5 off
.cs 6 on
[]Messages[%
Ø330E INCLUDE did not process prefix “prefix” successfully
Ø331W Prefix “prefix” not defined. Ignored
.cs 6 off
```

LOCATE HELPCSC

```
.cm VM Software Services
.cm
.cs Ø on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[%
[%
Use the LOCATE command to search the CSC log file for a character
string.
EXAMPLE: LOCATE /ABC
Locate string ABC in log file.
.cs Ø off
.cs 1 on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[]Purpose[%
Use the LOCATE command to search the CSC log file for a character
string.
.cs 1 off
.cs 2 on
[]Format[%
>>—.———.—string———————————————————><
‘-Locate-'
[]Class[%.21
```

```
4
.cs 2 off
.cs 3 on
[]Operands[%
string
is the data string to locate. The first byte is the string
delimiter.
The string can be from 1 to 36 characters long.
.cs 3 off
.cs 4 on
.cs 4 off
.cs 5 on
[]Usage Notes[%
1. The command name may be omitted if the string delimiter is the
character “/”. LOCATE <ABC and /ABC are equivalent commands.
2. The search starts at the first data line on the screen and
proceeds backward. The first line found containing the search
string will become the new top line.
3. The closing delimiter is optional. /ABC/ and /ABC are equivalent
strings.
4. The string delimiter may appear in the string itself. /A*B/ and
*A*B* are equivalent strings.
.cs 5 off
.cs 6 on
[]Messages[%
Ø31ØE Missing operand(s)
Ø35ØE String too long. Must be from 1 to 34 characters
Ø351I String not found
.cs 6 off
```

MATCH HELPCSC

```
.cm VM Software Services
.cm
.cs Ø on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[%
[%
Use the MATCH command to search backward the CSC log file for a.22
character mask.
EXAMPLE: MATCH ABC*
Search log file for record beginning with ABC.
.cs Ø off
.cs 1 on
(c) Copyright CSC Inc, 1997
[]CSC Tool[%
[
```

VM news

IBM announced there were no significant outages in VM systems during the Y2K period. IBM had hundreds of technical support staff available during the millenium shift but preparations that had been ongoing for the previous several years paid off. While worldwide preparations had been underway and it appeared most shops had taken advantage of the many tools available to upgrade their systems the final outcome was unknown. It became apparent as the countries in different time zones were unaffected by the passage to January 1, 2000, that systems people had done a superior job in preparing for the change.

***

IBM announced it was awarded the most U.S. patents in 1999. This was the seventh consecutive year IBM has been the leader in patents.

During the 1990s IBM has been issued over 15,000 patents, tripling its output of the previous decade. It also outdistanced the number-two for patents, Canon, by more that 2300 during the 1990s.

While it may not be possible to provide a wide array of patent examples that directly affect VM users, this continuing effort in expanding IBM's intellectual properties provides it with areas of long-term growth potential.

The most common thought is that IBM has patented specific hardware manufacturing properties and these are the cornerstone of its efforts. The truth is that during 1999 there were more than 900 software related patents issued to IBM. Microsoft Corporation was issued only 353 patents, and Oracle was not included in the top 50 of companies receiving patents.

Much of what was patented was in the area of e-commence positioning and of course the development of new chip manufacturing processes.

***

Again, while not having a specific impact on the VM community, it should be noted that IBM did release its 1999 year-end financial results. IBM had total revenue of $87.7 billion dollars and a net income of $7.7 billion.

The total revenue was approximately 7% ahead of the previous year's revenue. IBM bases all of its comparisons on constant currency figures.

Revenue increases occurred in all market areas and were fairly consistent. IBM did take certain accelerated expenses related to some of its acquisitions and write-downs related to the depreciable life of personal computers.

It also included $750 million in gains associated with several actions affecting its income. The sale of IBM Global Network to AT&T was noted as a specific item that affected its balance sheet.

It's good to know that Big Blue will be around for a long time.