164

VM update

Summer 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, Chuck Meyer Systems, Inc., USA.

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

Dynamic option on and off: set the VM block

Many of CP's directory options (such as DEVINFO, MAINTCCW, TODENABLE, etc.) require the user to LOGOF and LOGON before the option takes effect. Now you may turn many of these options ON and OFF dynamically, while the user is logged on, without even requiring any modification to the user's directory entry.

SETVMDBK

SETVMDBK allows a class-C user to alter a VMDBK (the VM block) of any currently logged-on user. If VSE1 needs MAINTCCW set ON (perhaps so that it can run DSF), just have any Class-C user issue "SETVMDBK VSE1 MAINTCCW", and the job is done. Would you like to set VSE2's LNKNOPAS to OFF? Enter "SETVMDBK VSE2 \LNKNOPAS". You may turn OFF any option, by prefacing it with any commonly used not sign ("?", "\", "~").

Some OPTION information (such as SPOOLMAX, V=R, etc.) is either not kept in the VMDBK, or only takes effect at LOGON time. Such options can not be altered by this program.

This program also operates as a REXX function-call, but then it operates only on its own VMDBK. When invoked this way, data is returned which can be used to restore the option back to its original value. So an EXEC that has to run DSF on any class-C userid may use the following to guarantee execution:

```
sav1 = SETVMDBK('MAINTCCW')        /* set my MAINTCCW to ON     */
'ICKDSF ..."
Call SETVMDBK sav1                 /* restore MAINTCCW as found */
```

Whether invoked as a COMMAND or as a FUNCTION, the following are always true:

- Each execution alters only one VBDBK.
- Any number of options may be turned ON and/or OFF on one execution.
- Abbreviations are allowed, to the minimum length needed for uniqueness.

- Some options allow for many characters or bytes to be altered—by specifying a keyword, an equal sign, and the desired value (*a la* Assembler keyword macros).

So the following command will alter four of PHRED's options:

```
SETVMDBK PHRED \LNKE LNKS PRIV=+KVS DEVI
```

1. Link-Exclusive is set OFF
2. Link-Stable is set ON
3. PrivClass is promoted to KVS
4. Device-Info is set ON

PrivClas alterations are done REGARDLESS of the "SET_PRIVCLASS" option defined in the SYSTEM.CONFIG.

SETVMDBK has no external requirements, just class-C at execution time.

Two somewhat related, but otherwise independent, programs are also provided in this article. They also requiring only class-C privileges to execute.

DH

DH provides full-screen-display of HostStorage ("D H" command), with a lot of shortcut options. Some possible options are as follows:

- "DH 1000 200" shows same storage as "CP D HT1000 200"
- "DH PHRED VDEV 191" shows PHRED's VDEV-block for 191
- "DH RDEV 220" shows the RDEV-block for device 0220

QVTOD

QVTOD displays the TOD clock of any logged-on user. This is mainly to inquire about guests who have altered their TOD clock by doing one of the following:

- Issuing a "CP SET VTOD" command.
- Using the guest OpSys to set a new TOD value.
- Specifying "TOD=xxx" or "EPOCH=yyy" on a SETVMDBK command.

SETVMDBK EXEC

```
********************************************************************************
**** 237-line source for SETVMDBK EXEC X6 (2000-04-03 12:38:49) follows ... ****
********************************************************************************

/*-------------------------------------------------------------------*/
/* SETVMDBK                                                          */
/*                                                                   */
/* FORMAT: SETVMDBK   userid    option1  <... optionn>               */
/*                                                                   */
/* Alters data in the VMDBK of a specified, logged-on, user.         */
/* If invoked as a FUNCTION, DO NOT specify userid, since this       */
/* form can ONLY alter the CALLER's VMDBK;  it also returns          */
/* values which can later be restored.                               */
/*                                                                   */
/*     savlnk = SETVMDBK('MAINTCCW')  /* allow DSF */                */
/*     'ICKDSF ... '                  /* RUN DSF   */                */
/*     Call     SETVMDBK savlnk       /* restore   */                */
/*                                                                   */
/*                                                                   */
/* 1999-08-10 CHM Version 0.0                                        */
/* 1999-12-01 CHM Add TODENable, and TOD (they are NOT co-reqs)      */
/* 2000-04-03 CHM Add PRIVCLAS and SECUSER (QUIETLY); and CPUID (full) */
/*               Clean up SET_BIT processing;  Add HELP.             */
/*-------------------------------------------------------------------*/
copyright = 'Copyright: Chuck Meyer Systems, Inc.;  1999 2000'
version   = '2000.04.03'

Parse Upper Source op_sys how_called myfn myft myfm myalias .
Parse Upper Arg  arg
Trace       OFF
?cmd    = (how_called='COMMAND')
?fun    = (how_called='FUNCTION')
calbak  = '44'x     /* Flag-char to indicate a CallBack ("RESTORE") */
sep     = '/'       /* Separates sub-parameters on return           */
?rst    = (POS(calbak,arg)>0)             /* Is this a RESTORE ??   */
arg     = SPACE(TRANSLATE(arg,,calbak))   /* remove possible flag   */
ret_str = ''                              /* ReturnString to caller */
expoze  = 'vmdbk ret_str sep ?fun ?cmd ?rst ?not optnam vmdbkn expoze'
vmdbkn  = ''                 /* VMDBK field-name, optional bit-name */
Address COMMAND
If (WORDS(arg)<1)  Then Exit HELP_ME()
Signal ON  NOVALUE
If (WORDS(arg)=1)  |  ?fun
  Then Parse Value '*'  With userid .
  Else Parse Value arg  With userid arg
If (userid='*')  Then userid = USERID()
Call DIAG_8 'Q U'    userid
Call DIAG_8 'LOCATE' userid , '  (PrivClas "C" needed)'
vmdbk   = WORD(cp_msg,3)                           /* A(VMDBK)     */

Do i=1  For WORDS(arg)            /* check all input words           */
  Parse Upper Value WORD(arg,i)  With  wd . 1  c1 +1 wd1
  If \DATATYPE(c1,'A')            /* If col-1 IS a special-character */
    Then x =  '0'     wd1         /*  indicate we turn it OFF        */
    Else x =  '1'     wd          /*  else we turn it ON             */
  Parse Var x  ?not w   .
  Parse Var w  w1 '=' w2
  w2x    = C2X(LEFT(w2,8))        /* possible chardata into hex      */
  w2b    = TRANSLATE(w2,,'.,/:-') /* with separators blanked out     */
  w21    = SPACE(w2b ,0)          /* with blanks smooshed out        */
  If DATATYPE(w21,'W')  Then w2d = D2X(w21,8);  Else w2d = COPIES('0',8)

  ?2     = (w21\='')              /* Is there a second sub-word ??   */
  optnam = ''
  Select
    When ?rst                       Then Call SET_RST     w
    When ?A('??CFLF    RFEAT CFLF' ) Then Call SET_BIT   721 4
    When ?A('ACCTUser  ACTID'      ) Then Call SET_BYTE  136'?'
    When ?A('ACct      RFEAT ACTRC') Then Call SET_BIT   721 6
    When ?A('APplmon   MONFA AMDIR') Then Call SET_BIT  1564 0
    When ?A('BIts'                 ) Then Call SET_BIT   w2b
    When ?A('BYtes'                ) Then Call SET_BYTE  w2b
    When ?A('COmsrv    RFEAT CMSRV') Then Call SET_BIT   721 5
    When ?A('CPuid     CPVER'      ) Then Call SET_BYTE  704 w21
    When ?A('DEVInfo   IAGFL DEVI' ) Then Call SET_BIT   720 4
    When ?A('DEVMaint  IAGFL DEVM' ) Then Call SET_BIT   720 5
    When ?A('DIAG98    IAGFL IAG98') Then Call SET_BIT   720 0
    When ?A('DIStrib   DIST'       ) Then Call SET_BYTE  160 w2x
    When ?A('D84nopas  IAGFL D84NP') Then Call SET_BIT   720 2
    When ?A('Epoch     EPOCH'      ) Then Call SET_EPOCH w21
    When ?A('LKfac'                ) Then Call SET_BIT  '??? ?'
    When ?A('LNKExclv  IAGF2 LNKS' ) Then Call SET_BIT   719 4
    When ?A('LNKNopas  IAGFL LNKNP') Then Call SET_BIT   720 1
    When ?A('LNKStabl  IAGF2 LNKS' ) Then Call SET_BIT   719 4
    When ?A('MAIntccw  IAGFL MCCW' ) Then Call SET_BIT   720 6
    When ?A('MAXConn'              ) Then Call SET_BYTE '???'w2d
    When ?A('MAXVmcfi  MAXVF'      ) Then Call SET_BYTE 2268 w2d
    When ?A('NOMDcfs   IAGF2 NOFSL') Then Call SET_BIT   719 0
    When ?A('NOPdata   CCWOP NOP'  ) Then Call SET_BIT  1432 2
    When ?A('NOVf      RFEAT NOVFA') Then Call SET_BIT   721 0
    When ?A('PRIVclas  PCL'        ) Then Call SET_PRIVC  w2
    When ?A('Quickdsp  SCDF1 QDSPU') Then Call SET_BIT  1912 4
    When ?A('Rmchinfo  IAGF2 CSRMI') Then Call SET_BIT   719 1
    When ?A('SECuser   SECU'       ) Then Call SET_BYTE 1264 w2x
    When ?A('SETorig   IAGFL SETOR') Then Call SET_BIT   720 3
    When ?A('STDevopt'             ) Then Call SET_BIT  '742 ?'
    When ?A('STGexempt'            ) Then Call SET_BIT  '??? ?'
    When ?A('SVC76vm   RFEAT VERP' ) Then Call SET_BIT   721 3
    When ?A('SVMstat   RFEAT SMVST') Then Call SET_BIT   721 1
    When ?2 & ?A('Tod  EPOCH'      ) Then Call SET_EPOCH w21
    When ?A('Todenable RFEAT VTOD' ) Then Call SET_BIT   721 7
    Otherwise  Call SAY_IT 'Cannot understand option "'wd'";  it it ignored.'
    End  /*  Select  */
  End  /*  Do i=1 ...  */
Call ALL_DONE 0 /*================================================*/

?A:  /* Is W1 a true ABBREViation of non-lower-case portion of A1 ??? */
  Parse       Arg a1 vmdbk_byte vmdbk_bit .
  Parse Upper Var a1 optnam .
  vmdbkn = '(VMD' || WORD(vmdbk_byte '?????',1)    ,
                  || WORD(',VMD'vmdbk_bit   ,1+(vmdbk_bit=''))  || ')'
  Return ABBREV(optnam,w1, ,
          LENGTH(SPACE(TRANSLATE(a1,,'abcdefghijklmnopqrstuvwxyz'),0)))

SET_BIT:  Procedure Expose  (expoze)
  Do a=1  For ARG()
    Parse Value  ARG(a)   With  byte bit rev .
    If \DATATYPE(byte,'W') | \DATATYPE(bit,'W')  Then Iterate
    If          (bit<0)    |          (bit>7)    Then Iterate
    If          (byte<1)                         Then Iterate
    rev    = WORD(rev '1',1)                   /* default value      */
    vmdbka = D2X(X2D(SPACE(vmdbk))+byte)       /* A(VMDBK)+disp      */
    Call DIAG_8 'D HS'vmdbka , '  (Err in "' , '")'
    xbyte  = X2B(LEFT(WORD(cp_msg,2),2))       /* byte w/ bit in BIN */
    xbit   = SUBSTR(xbyte,bit+1,1)             /* cur. value of  bit */
    If (WORDPOS(rev,'0 1')>0)                  /* new  value for bit */
      Then ?on = (rev = ?not)            /* If "0" or "1"; use as is */
      Else ?on = \xbit                   /*  else flip current value */
    xbdot  = OVERLAY(?on,'........'  ,1+bit)    /* Lookin' cute       */
    nubyt  = OVERLAY(?on,xbyte       ,1+bit)    /* NewByte in BIN     */
    nubytx = B2X(nubyt)                        /* NewByte in HEX     */
    If (?on=xbit)  Then Do  /* No SET needed, it's as we want it */
      Call SAY_IT 'No change needed to' vmdbka optnam ,
          SPACE(xbdot vmdbkn)
      End
    Else Do                    /* Let's flip the bit                */
      Call DIAG_8 'ST HS'vmdbka nubytx ,   '  (Err in "' , '")'
      ret_str = STRIP(ret_str byte || sep || bit || sep || xbit)
      Call SAY_IT 'Changed' vmdbka optnam SPACE(vmdbkn 'to')  ,
                  xbdot 'R="'ret_str'"'
      End
    End  /*  Do a=1  For ARG()  */
  Return 0

SET_BYTE: Procedure Expose  (expoze)
```

```
  Parse Upper Arg    byte nudata
  nudata = SPACE(TRANSLATE(nudata,,',./'),0)
  len2   = LENGTH(nudata)
  If \DATATYPE(nudata,'X')  |   (len2 <1)  Then Return 0
  If \DATATYPE(byte  ,'W')  |   (byte <1)  Then Return 0
  len2   = (len2%1) + ((len2//1)\=0)  /* round up to an even number */
  len    = (len2%2)                   /* true length of data        */
  nudata = LEFT(nudata, len2,'0')     /* pad data to same length    */
  vmdbka = D2X(X2D(SPACE(vmdbk))+byte)        /* A(VMDBK)+disp      */
  Call DIAG_8 'D HS'vmdbka'.'len , '  (Err in "' , '")'
  xbytes =  WORD(cp_msg,2)                    /* current value      */
  If (nudata=xbytes)  Then Do  /* No SET needed, it's as we want it */
    Call SAY_IT 'No change needed to' vmdbka optnam ,
         SPACE('('xbytes')' vmdbkn)
    End
  Else Do                     /* Let's alter the value             */
    Call DIAG_8 'ST HS'vmdbka nudata ,   '  (Err in "' , '")'
    ret_str = STRIP(ret_str byte || sep || xbytes)
    Call SAY_IT 'Changed' vmdbka'.'len optnam SPACE(vmdbkn 'to') ,
                 nudata 'R="'ret_str'"'
    End
  Return 0

SET_RST:  Procedure Expose  (expoze)
  Parse Arg a
  Parse Value TRANSLATE(a,,sep)  With w1 w2 w3 w4
  Select
    When \(w4='')          Then Call SAY_IT 'Too many subparms in "'a'"'
    When \DATATYPE(w1,'W') Then Call SAY_IT 'Invalid displacement in "'a'"'
    When (POS(w3,'01')>0)  Then Call SET_BIT  w1 w2 w3
    When  DATATYPE(w2,'X') Then Call SET_BYTE w1 w2
    Otherwise                   Call SAY_IT '???Unknown??? "'a'"'
    End  /*  Select  */
  Return

SET_EPOCH:              /* VMDEPOCH   312 138 */
    /* Similar to "CP SET VTOD DATE yyyy-mm-dd"                 */
    /*        and "CP SET VTOD SYSTEM";  but no reIPL is needed */
  Parse Upper Arg e
  Numeric Digits 33
  If (optnam='TOD')  Then ,  /* Calculate TOD displacement */
    e = D2X(54E8*(DATE('B',OVERLAY(e,DATE('S')),'S')-DATE('B')),12)
  Return SET_BYTE(312 LEFT(e,16,'0'))

SET_CPUID:              /* VMDCPUID   704 2C0 (8) */
  Return -3

SET_PRIVC:              /* VMDPCL     960 3C0 (4) */
    /* Similar to "CP SET PRIVCLAS x y";  but does not require  */
    /* SYSTEM.CONFIG to contain "FEATures ENABle SET_PRIVclass" */
  Parse Value DIAG(8,'QUERY PRIVCLAS' userid)  With ':' cur_pc '15'x
  Parse Upper Value SPACE(ARG(1),0) With new_pc
  If ABBREV(new_pc,'-') Then new_pc = TRANSLATE(cur_pc,,SUBSTR(new_pc,2))
  If ABBREV(new_pc,'+') Then new_pc =          cur_pc  SUBSTR(new_pc,2)
  Return SET_BYTE(960 C2X(PC_32TO4(new_pc)))

   /* The following 2 procedures process PRIVCLAS data, where each of */
   /* 32 bits indicates one of "ABCDEFGH IJKLMNOP QRSTUVWX YZ123456". */

PC_32TO4:  Procedure /* Cvt string of 0-32 classes into 4 bytes (hex) */
                     /* in-chars may be any order, invalid, or dupl.  */
  Parse Upper Value 'ABCDEFGHIJKLMNOPQRSTUVWXYZ123456' SPACE(ARG(1),0) ,
              With   a5                                c      o
  Do i=1 To 32   /* Now go thru each possible class; make it a 0 or 1 */
    o = o || (POS(SUBSTR(a5,i,1),c)\=0)
    End  /* Do i=1 To 32 ... */
  Return B2C(o)

PC_4TO32:            /* Cvt 4 bytes (32 bits) into 32-byte char-string*/
  Return TRANSLATE(BITAND('ABCDEFGHIJKLMNOPQRSTUVWXYZ123456' , ,
         TRANSLATE(C2B(ARG(1)),'00FF'X,'01')),'-','00'x)

DIAG_8:  /* return only if RC is zero;  else ABEND  */
  Parse  Arg   cp_cmd , a2 , a3
  cp_cmd = STRIP(TRANSLATE(cp_cmd))
  Parse Value SPACE(TRANSLATE(DIAGRC(8,cp_cmd),,'15'x))  ,
        With  cp_rc  cp_cc  cp_msg
  If (cp_rc=0)  Then Return
  If (a3 \='')  Then a2 = a2 || cp_cmd || a3  /* add CMD to message */
  Call ALL_DONE cp_rc cp_msg a2   ' (rc='cp_rc')'

SAY_IT:
  If ?cmd  Then Say ARG(1)
  Return 0

ALL_DONE:
  Parse Arg r1 m1
  If (m1\='')  Then Call SAY_IT   STRIP(m1)
  If \?cmd  Then Exit SPACE(calbak ret_str)
  Exit WORD('0' r1 , 1+DATATYPE(r1,'W'))

HELP_ME:
  'HELP' myfn
  Return 0

/*------------------------end of SETVMDBK.EXEC------------------------*/
```

SETVMDBK HELPCMS

```
********************************************************************************
** 192-line source for SETVMDBK HELPCMS X6 (2000-04-03 12:37:58) follows ... ***
********************************************************************************

.CS 2 ON
.cm Copyright Chuck Meyer Systems, Inc. ;  2000   (cmsi@attglobal.net)
.cm 2000-04-03 CHM Written

c|SETVMDBKc%

 Syntax for COMMAND invocation:

        SETVMDBK  userid  opt1  <opt2  <opt3 ...>>

 Syntax for CALL or FUNCTION (operate only on caller's VMBDK):

        Call SETVMDBK  opt1  <opt2  <opt3 ...>>

        xxx = SETVMDBK(opt1  <opt2  <opt3 ...>> )

 Each blank-delimited  option may be  either a single-word option;   or an
 assembler-like (keyword)  parameter, which  is a  keyword followed  by an
 equal-sign and the desired new value.

 Single-word  options  are words  exactly  like  the  words on  an  OPTION
 statement.  By  default the option is  turned ON;  to turn  it OFF, begin
 the word with a not-sign  or reverse-slash.  Abbreviations are allowed as
 shown by  lower-case letters (note  that many of these  abbreviations are
 shorter than DIRECTXA would allow).  Supported words are:
  ACct     APplmon  COmsrv    DEVInfo  DEVMaint DIAG98   D84nopas
  LNKExclv LNKNopas LNKStable MAIntccw NOMDcfs  NOPdata  NOVf
  Quickdsp Rmchinfo STDevopt  STGexemp SVC76vm  SVMstat  Todenable

 Supported keyword options are:
  ACCTUser=uuuuuuuu    (user to be charged with ACNT data)
  BITs=disp/bitnum/b   (set bit BITNUM in byte DISP to B)
  BYTEs=disp/hexval    (set bytes starting at DISP, to HEXVAL)
  CPuid=xxx            (full 8-byte CPUID)
  DIStrib=dddddddd     (Distribution-code for subsequently-added UR devs)
  Epoch=xxx            (SET VTOD,     w/o reIPL)
  MAXConn=nnnnnn       (Max number of IUCV & APPC connections)
  MAXVmcfi=nnnnnn      (Max number of VMCF input messages queued)
  PRIVclas=cccccccccc  (SET PRIVCLAS, w/o msgs, restrictions & AcntRecs)
  SECuser=uuuuuuuu     (SET SECUSER,  w/o msgs)
  Tod=ddd              (SET VTOD,     w/o reIPL)
 Not all possible DIRECTXA OPTIONs are supported, because either --

   1. The value for the corresponding option  is not kept in the VMDBK (the
      value for SPOOLMAX,  for example, must be available whether  or not a
      VMDBK exists;   so its value  is only  kept in the  object directory,
      from where it is read when needed);  or

   2. The option  is only used during  LOGON (the values for  V=R, SETORIG,
      and LOGON-VSIZE, for example).

.CS 2 OFF
.CS 1 ON
c|Purposec%

 SETVMDBK will  alter the  VMDBK ("VM  Block") of  any virtual  machine by
 setting ON  or OFF those  options which normally  can only be  altered by
 updating the OPTION statement of the user's directory entry.

 This program requires CP class C privileges (to alter main storage).

 In addition  to obviating directory  changes, this program  also obviates
 any LOGOFF/LOGON  sequence to bring  the option(s) into  effect.  Options
 such as  DEVINFO, MAINTCCW, LNKEXCLV, etc.,  may now be turned  ON or OFF
 dynamically.

 If  invoked as  a COMMAND,  then  ANY user's  VMDBK may  be altered.   If
 invoked via a REXX CALL or FUNCTION, then only the caller's own VMBDK may
 be altered, and a string (containing all of the before-alteration values)
 is returned to  the caller, so that a proper  restore may subsequently be
 done.

.CS 3 OFF
.CS 5 ON
c|Usage Notesc%

   1. User must have class-C privileges.

   2. EPOCH= and  TOD= each  alter the  same VMDEPOCH  field in  the VMDBK.
      This field  contains an  8-byte SIGNED  value which  is added  to the
      REAL, CPU-maintained  TimeOfDay clock,  to create the  user's VIRTUAL
      TOD.  This field is usually zero,  but can be officially altered with
      the "SET  VTOD" command (which  also issues  a "SYSTEM RESET"  to the
      guest OpSys -  requiring an IPL).  This field is  also altered when a
      guest OpSys sets  its TimeOfDay clock (when allowed  by the TODENABLE
      directory  option).  Using  a non-zero  VMDEPOCH for  a CMS  guest is
      completely useless, since the various  components of CMS will produce
      differing  dates and  times.  So  this is  normally used  immediately
      prior to IPLing  a guest VSE, MVS  or VM, to avoid  using the guest's
      method for setting  the date and time, or for  creating a predictable
```

TOD value for the guest.

EPOCH= and TOD= allow for two different methods of specifying the new value for VMDEPOCH;

EPOCH=xxx defines exactly the value for VMDEPOCH in HEX. It will then be RIGHT-padded with zeros, to a length of 8 bytes. This is a SIGNED binary number, so that values beginning with 8 thru F are negative. For example, to add exactly 24-hours to the virt-clock, specify "EPOCH=000141DD76" (24 hrs = 86400 seconds = 353,894,400,000,000 TU's or hex 0001,41DD,7600,0000)

TOD=ddd defines a desired virtual DATE value to be seen by the guest. SETVMDBK will then calculate the new value for VMDEPOCH using that date and the current time. The date is specified in "S" format (CCYYMMDD), but may be abbreviated to any length, the missing data defaults to that of the current date. For example, TOD=19981201 sets VMDEPOCH so that the guest sees a TOD value showing 1998-12-01 and the current time; TOD=199812 alters only the YEAR and MONTH; TOD=1998 alters only the YEAR.

(Use tool "QVTOD" to see the current value of any user's VMDEPOCH and virtual TOD in "English.")

3. CPUID=xxx allows control over the entire 8-byte CPUID. This is useful during disaster recovery, when running on a different CPU-type, and you have programs which are sensitive to that. Byte-0 normally contains 'FF', bytes 1 thru 3 the serial number (changeable by user with SET CPUID command), bytes 4+5 contain the CPU-type, e.g. "9021". XXX may be 2 to 16 hex characters, and will overlay the corresponding number of bytes in VMDCPVER (i.e., unspecified bytes are not altered).

4. PRIVCLAS=ccc is similar to the SET PRIVCLAS command, except it's dependent on privclas C, rather than on SYSTEM.CONFIG containing "FEATURES ENABLE SET_PRIVclas". The value of CCC may be any string of characters A-Z and 1-6 --- the specified characters REPLACE the target privclas. If CCC is prefaced with a "+" or "-", then the specified characters are added-to or subtracted-from the target privclas. Duplicate or invalid characters are bypassed without error. For example, "PRIVCLAS=-B*A" removes B and A, "*" is ignored.

```
.CS 5 OFF
.CS 5 ON
```

c|Examples of COMMAND invocationc%

```
    SETVMDBK LNKNOPAS
      Turns my LNKNOPAS option ON.   Note that when user is self-modifying
      a single option, then the userid is not needed.

    SETVMDBK * LNKNOPAS DEVINFO
      Turns my LNKNOPAS and DEVINFO options ON.  Note that since more than
      one option is specified, a userid ("*" in this case) is required.

    SETVMDBK * LNKN DEVI
      Same as preceding example, except showing minimum abbreviations.

    SETVMDBK VSEPROD1 \LNKNOPAS MAINTCCW NOPDATA PRIVCLAS=-AF
      For user  named VSEPROD1:  turn OFF LNKNOPAS,  turn ON  MAINTCCW (to
      allow VSE to run full DSF), turn ON NOPDATA (to allow spooled output
      to  contain NOPs  -  possibly for  microfiche  output), and  removes
      privilege classes A and F

    SETVMDBK FRED BIT=960/7/1
      In FRED's  VMDBK, sets  byte 960  (decimal) bit  7 (rel  0) to  a 1.
      (Currently that's the same as would be done by "PRIVCLAS=+H".)

c|Examples of FUNCTION-call invocation from a REXX programc%
    (This allows only the caller's own VMBDK to be altered)

  1------------------------------------------------------------------
    sav_vmdbk = SETVMDBK('DEVINFO')              /*#1 set DEVINFO ON */
    'PIPE CP Q MDISK USER PHRED 190-19F|STEM M.' /*#2 get MDISK info *
    Call SETVMDBK sav_vmdbk                      /*#3 restore VMDBK  */
      #1 Sets DEVINFO ON
      #2 Issues a CP command which requires DEVINFO
      #3 Rstores the DEVINFO option as it was found.
  2------------------------------------------------------------------
    sav_vmdbk = SETVMDBK('DEVM /NOMD DIST=DSF-LIST')    /*#1*/
    Call DIAG 8 , 'DEFINE 4248 60E'                     /*#2*/
    'PIPE COMMAND ICKDSF ... |URO 60E ...'              /*#3*/
    Call DIAG 8 , 'DETACH 60E'                          /*#4*/
    Call SETVMDBK sav_vmdbk                             /*#5*/
      #1 Sets DEVMAINT ON, NOMDCFS OFF, and DistCode to DSF-LIST
      #2 Defines a virtual 4248 (which will have a DIST of DSF-LIST)
      #3 Runs a PIPE which invokes DSF and places its output into
          the stream and routes it (with URO) to this 4248
      #4 DETACHes the printer (which also CLOSEs it)
      #5 Restores the 3 options in caller's VMDBK to their saved values.
  3------------------------------------------------------------------
    Call SETVMDBK 'TOD=1972' /*#1 set YEAR of TOD to 1972 (like 2000) */
    Call DIAG 8,'IPL 240'    /*#2 IPL a guest OpSys                   */

.CS 5 OFF
/*------------------------end of SETVMDBK HELPCMS-------------------------*/
```

DH EXEC

```
********************************************************************************
******* 276-line source for DH EXEC X6 (1997-04-07 18:19:20) follows ... *******
********************************************************************************

/* Display CP Host Storage ********************************************\

  Full-screen display of  CP Host (Real) Storage by creating  a CMS file
  ("DH RESULTS A"), and then BROWSing it.  (This means that the data may
  then be  easily printed.) This  display includes not  only "dump-like"
  output, but also  shows hex and decimal displacements  from the "base"
  address.

  Invoker must use one of these two forms:

            DH    addr  len

            DH    locate_type  <opt1  <opt2  ... > >

  The FIRST  FORM allows user to  specify address and length  in exactly
  the same  form as used  in the CP "DISPLAY  H" command (both  in hex).
  For example:

    DH 0 200          displays 512 bytes (x'200') starting at 0
    DH 0AA71 1000     displays 4096 bytes (x'1000') starting at x'AA71'
                      (Note that NO BOUNDARY-ALIGNMENT is done!)

  The SECOND FORM allows the user  to provide a "type" which corresponds
  to one of the  types allowed on the CP "LOCATE"  command.  This may be
  abbreviated  to the  minimum  needed for  uniqueness.  Following  this
  keyword  are  those  options  needed  by  the  corresponding  "LOCATE"
  command.  DH  then executes this  LOCATE command, extracts  the proper
  address, and displays storage starting at that address.  The supported
  keywords (followed by its options) are:

     Filid  SYSTEM|userid|*|*IMG|*NLS|*NSS|*SDF|*UCR
     Ldev  <ldevaddr>
     Rdev   rdevaddr
     SHpbk <system>
     SNabk <luname  VSM VTAM*>
     SPfbk  SYSTEM|userid|*|*IMG|*NLS|*NSS|*SDF|*UCR type spid
     SYmbol symbol
     VDev  <userid vdevaddr>
     VMdbk <userid>
     VSmbk <userid>

  The program also  displays (up to) the first 8  lines of response from
  the LOCATE command.  A limit was established because some forms of the
  LOCATE command can  create many lines of output.  And  8 seemed like a
  reasonable limit.

  Note that LENGTH is specified differently to -
     STORAGE(xxx,len) : "len" (length) is in DECIMAL
     CP  D  Hxxx.len  : "len" (length) is in HEX
  DH always assumes that length is specified in HEX.

  This program reads real storage 16 bytes at a time (GoodNews/BadNews)
    Advantages:
      No "suppressed line(s) same as above" messages
      No boundary-alignment requirements
      Able to show hex and decimal displacements
    Disadvantages:
      Synchronization/timing (possible inconsistent data)
      Browses "historical" ("snapshot") data, not real-time data.

################################################################################

1996/05/22 CHM/CMSi Copyright by Chuck Meyer Systems, Inc.
1996/09/16 CHM/CMSi Allow Function/Subroutine CALL,  by returning the
                    raw data to the caller <similar to STORAGE()>.
1997/04/07 CHM/CMSi made function-call work eggzactly like STORAGE(),
                    minus its masochistic storage-alteration ability.
\*********************************************************************/
copyright = '? Copyright: Chuck Meyer Systems, Inc.;  1997'
version   = '1997.04.07'

Address COMMAND
Trace   OFF
Parse Upper Source . how_called myfn myft myfm myfn2 .
out_file  = myfn 'RESULTS  A1'           /* Output file name */
hlp_file  = myfn 'HELP_ME  A3'           /* Help   file name */
my_fname  = myfn  myft     myfm          /* This   file name */
?cmd      = (how_called='COMMAND')       /* Is this a COMMAND?  */
?fun      = ??cmd                        /* Is this a FUN/SUBR? */
Parse Upper Arg   arg1  ,  olen
arg1      = SPACE(arg1)
olen      = SPACE(olen)
Parse Upper Var   arg1     typ op2
Parse Upper Var   op2          op21 op2x

     /* make sure we have proper authorities to run (usually C|E) */
Call CHECK_PRIV 'LOCATE VMDBK QQQQQQQQQ' , 20 , 'LOCATE'       , 'C|E'
Call CHECK_PRIV 'DISPLAY H*'             , 03 , 'DISPLAY HOST' , 'C|E'

If ABBREV('???',typ)  &  ??fun  Then Call HELP_ME

     /* create some variables used in output headers */
```

```
'PIPE (ENDCHAR $)'                            ,
     'CP D HS808.8     |SPEC W2|VAR  CPUID'     ,
    '$CP Q TIME OFFSET         |VAR  QTO'       ,
    '$CP Q CPLEVEL             |STEM CPLEVEL.'  ,
    '$CP Q STORAGE     |SPEC W3|VAR  RSIZE'
Parse Upper Var qto . . tz . qto1 qto2 .
Parse Value RIGHT(rsize,10,'0')  With rsiz1 +9 rsiz2 +1 .
Select       /* normalize the time */
  When  qto2 = 'WEST'  Then z = tz'(Z-'LEFT(qto1,2)/1')'
  When  qto2 = 'EAST'  Then z = tz'(Z+'LEFT(qto1,2)/1')'
  Otherwise                 z = tz'(Z)'
  End
If ?fun  &  (typ='')  Then Exit KNUMBER2HEX(rsize)  /* xx = DH() */
sep      =       COPIES('-',79)
h.1      = '            'myfn'.'myft'.'myfm '(Ver='version') run by'  ,
                   USERID() 'at' NODE_ID()
h.2      = '  Time=====' LEFT(DATE('W'),3)','                         ,
                  TRANSLATE('CcYy/Mm/Dd','-'DATE('S'),'/CcYyMmDd')    ,
               '('TRANSLATE(  'Yy/123'  ,'/'DATE('J'),'/Yy123'   )')' ,
               '(BaseDate='DATE('B')')'                               ,
                    'at' TIME() z'.'
h.3      = '  CP_Level=' cplevel.1
h.4      = '           ' cplevel.3
h.5      = '  Cpuid====' LEFT(cpuid,8) RIGHT(cpuid,8) '    RStor='rsize
h.6      = sep
h.7      = '?'
h.0      = 7
?raddr   = (op21 \= '')  &  DATATYPE(typ,'X')  &  DATATYPE(op21,'X')
_ldev    =       WORD(op21 'L0000-LOFFF' , 1)
_snabk   =       WORD(op21 '*'           , 1) op2x
_spf     =       WORD(op21 'SYSTEM'      , 1) op2x
_uid     =       WORD(op21 USERID()      , 1)
_vdev    = _uid  WORD(op2x '191'         , 1)
_vsmbk   =       WORD(op21 'VTAM'        , 1)

Select
  When   ?raddr                  Then x = ''
  When ABBREV('FILID'   ,typ,2)  Then x = 'FILID  2 2 0400' _spf
  When ABBREV('FRAMETBL',typ,2)  Then x = 'FRMTE  2 2 0400' op2
  When ABBREV('FRMTBL'  ,typ,2)  Then x = 'FRMTE  2 2 0400' op2
  When ABBREV('FRMTE'   ,typ,2)  Then x = 'FRMTE  2 1 0400' 'ENTRY' op2
  When ABBREV('LDEV'    ,typ,1)  Then x = 'LDEV   2 3 01F0' _ldev
  When ABBREV('RDEV'    ,typ,1)  Then x = 'RDEV   2 2 01F0' op2
  When ABBREV('SHPBK'   ,typ,2)  Then x = 'SHPBK  2 3 0200'
  When ABBREV('SNABK'   ,typ,2)  Then x = 'SNABK  2 3 0060' _snabk
  When ABBREV('SPFBK'   ,typ,2)  Then x = 'SPFBK  2 4 00C0' _spf
  When ABBREV('SYMBOL'  ,typ,2)  Then x = 'SYMBOL 1 3 0000' op2
  When ABBREV('VDEV'    ,typ,2)  Then x = 'VDEV   2 3 00C8' _vdev
  When ABBREV('VMDBK'   ,typ,2)  Then x = 'VMDBK  2 2 099F' _uid
  When ABBREV('VSMBK'   ,typ,2)  Then x = 'VSMBK  2 2 003F' _vsmbk
  Otherwise                           x = 'VMDBK  2 2 099F' USERID()
  End

Parse Upper Var x  loc1 linex wordx length loc2

If ?raddr  Then Do
  loc. = 0
  Call CP_DISPLAY_HT typ op2
  End
Else Do
  cpcmd = STRIP('LOCATE' loc1 loc2)
  'PIPE CP' cpcmd '| TAKE 08 | SPECS 1-* 10 | STEM LOC.'
  rc_loc = rc
  If (length=0)  Then Do
    wp     = WORDPOS('BYTES', TRANSLATE(loc.1) )
    If (wp>2)  Then Do
      ln = WORD(loc.1, wp-1)
      If DATATYPE(ln,'X')    Then length = ln
      End
    If (length=0)  Then length = '0400'
    End
  Call CP_DISPLAY_HT WORD(loc.linex , wordx) length
  End
If ?fun  Then Do
  If DATATYPE(olen,'X') & (LENGTH(olen)>0)
    Then olen = MAX(1, WORD(olen X2D(olen) , 1 + (ARG()>1)) )
    Else olen = X2D(length)
  'PIPE STEM HT. | JOIN * | CHOP' olen '| VAR EX_VALUE'
  Exit ex_value
  End
z = h.0
If ?raddr  ,
  Then h.z = '  "DISPLAY HT'typ'.'op21'"   has produced:'
  Else h.z = '  "'cpcmd'"   has produced the following results:'

'PIPE        STEM H.'    ,  /* headings                          */
   '| APPEND STEM LOC.'  ,  /* CP LOCATE + results;  or CP D H  */
   '| APPEND VAR  SEP'   ,  /* separator                         */
   '| APPEND STEM HT.'   ,  /* CP D HT results                   */
   '| APPEND VAR  SEP'   ,  /* separator                         */
   '| >'     out_file       /* out to the file                   */

'BROWSE'     out_file

Exit /*=========================================================*/

CP_DISPLAY_HT:  Procedure  Expose  ht. ?cmd ?fun
  Parse Upper Arg hex_addr hex_bytes .
  dec_addr  =       X2D(hex_addr )
  dec_bytes = MAX(1,X2D(hex_bytes))
  ht.       = ''
  ht.0      = CEILING(dec_bytes/16)      /* number of 16-byte chunks */
  non_disp  = XRANGE('00'x,'3F'x) || ,   /* non-display characters   */
              XRANGE('41'x,'49'x) || ,
              XRANGE('51'x,'59'x) || ,
              XRANGE('62'x,'69'x) || ,
              XRANGE('70'x,'78'x) || ,
                     '80'x        || ,
              XRANGE('8A'x,'90'x) || ,
              XRANGE('9A'x,'A1'x) || ,
              XRANGE('AA'x,'BF'x) || ,
              XRANGE('CA'x,'CF'x) || ,
              XRANGE('DA'x,'DF'x) || ,
                     'E1'x        || ,
              XRANGE('EA'x,'EF'x) || ,
              XRANGE('FB'x,'FF'x)
  Do i=1  By 1    For ht.0
    disp_d  = (i-1) * 16      /* displacement in dec */
    disp_x  = D2X(disp_d)     /* displacement in hex */
    Parse Value DIAGRC(8,'DISPLAY HS' || D2X(dec_addr+disp_d)'.10') ,
          With rc cc h_addr h_data h_key .
    If \(rc=0)  Then Leave
    Parse Var h_data   hd1 +8 hd2 +8 hd3 +8 hd4
    If ?cmd
      Then ht.i  = RIGHT(disp_d, 5) ,
                   RIGHT(disp_x, 4) ,
                   LEFT( h_addr,10) ,
                   hd1 hd2 hd3 hd4  ,
                   LEFT( h_key , 2) ,
                   '*' || TRANSLATE(X2C(h_data),,non_disp,'.') || '*'
      Else ht.i  =                  X2C(h_data)
    End
  ht.0 = MAX(ht.0,1)
  Return 0

CEILING:  Procedure
     /* Round a variable up to next higher integer                    */
     /*    CEILING(7.000)  ===>  7                                    */
     /*    CEILING(7.001)  ===>  8                                    */
   Parse Arg num__1 .
   If \DATATYPE(num__1,'N')   Then Return COPIES('9',DIGITS())
   Return TRUNC(num__1) + (TRUNC(num__1) <> num__1)

CHECK_PRIV:  Procedure
  Parse Arg cmd , rcx , cmd2 , class
  Parse Value DIAGRC(8,cmd,80)  With rc .
  If (rc=rcx)  Then Return
  'PIPE CP Q PRIVCLAS * | DROP 1 | TAKE 1 | SPECS W2 1 | VAR PR'
  Say 'You cannot execute the CP "'cmd2'" command   (rc='rc')'
  Call ALL_DONE '12 This usually requires CP Class' class ,
                  'authority;  you have "'pr'".'

ALL_DONE:
  Parse Arg r1 m1
  If m1 \= ''  Then Say m1
  Exit r1

HELP_ME:
  x  =  CENTRE(copyright'   Version='version,72)
  'PIPE <'      my_fname ,
      '|DROP    1'       ,
      '|TOLABEL ####' || ,
      '|APPEND  VAR X'   ,
      '|>'      hlp_file
  'BROWSE'      hlp_file
  Exit 0

NODE_ID: Procedure    /* Return name of current NodeID */
  'PIPE COMMAND IDENTIFY  | SPECS WORDS 3 1 | VAR NODEID'
  Return nodeid

KNUMBER2HEX:  Procedure /* cvt a nnn/nnnK/nnnM value to hex */
  Numeric Digits 15
  Parse Upper Arg v .
  Parse       Value RIGHT(v,10,'0')  With v1 +9 v2 +1 .
  Select       /* normalize the number (for algebra) */
    When ?DATATYPE(v1,'W')  Then d = 0
    When          (v2='K')  Then d = v1 * 1024
    When          (v2='M')  Then d = v1 * 1024 * 1024
    Otherwise                    d = 0
    End
  Return D2X(d)  /* in HEX, for function-call with no data */

/*---------------------------end of DH.EXEC---------------------------*/
```

QVTOD EXEC

```
********************************************************************************
***** 136-line source for QVTOD EXEC X6 (1999-12-03 16:35:51) follows ... ******
********************************************************************************

/**********************************************************************\
   Displays, for a specified user, the current VIRTUAL TOD-clock value.
   Needs PrivClas C or E;   and RXTOD.MODULE (from MAINT\193).

 1999-12-03 CHM Made public
\**********************************************************************/
copyright = 'Copyright: Chuck Meyer Systems, Inc.;  1999'
version   = '1999.12.03'

Parse Upper Value ARG(1) '*'  With uid .

uid    = WORD(uid USERID(),1+(uid='*'))

Parse Value DIAG_8('LOCATE' uid   )  With rc1 . . . vmdbk .
addr   =  D2X(312 + X2D(vmdbk),8)   /* Address of EPOCH in VMDBK */
Parse Value DIAG_8('D  HS'addr'.8')  With rc2 . . epochx .
Parse Value TIME('N') DATE('U') DATE('S') DATE('B')  With ,
              ktimn     kdatu     kdats     hdatb      .
kdati  =  TRANSLATE('CcYy/Mm/Dd','-'kdats,'/CcYyMmDd')
Numeric Digits 33
epochd =  X2D(epochx,16)
curtod =  TOD('T',kdatu||ktimn)
adjtod =  C2D(curtod) + epochd
Parse Value TOD('C',D2C(adjtod,8)) With dat +8 adjtim
ymd    =  TRANSLATE('Yy-Mm-Dd','-'dat,'-Mm/Dd/Yy')
adjdat =  19 + ((epochx>>'B')  &  (ymd<<'42-09-18'))  || ymd
epochx =  OVERLAY(STRIP(epochx,'T','0'),'0')
Say 'Currently' ':' kdati  ktimn  $(C2X(curtod))
Say LEFT(uid,9) ':' adjdat adjtim $(D2X(adjtod)) 'Offset(Epoch)='epochx

Exit 0 /*=============================================================*/

DIAG_8:
  Parse Value  SPACE(TRANSLATE(DIAGRC(8,TRANSLATE(ARG(1))),,'15'X)) ,
        With rc rr cpresp  1 totalresponse
  If (rc=0)  Then Return totalresponse
  Exit rc

$: Return TRANSLATE('abcd,efgh,ijkl,mnop',',',ARG(1),',abcdefghijklmnop')
```

```
/*********************************************************************\

  RXTOD is an IBM-supplied MODULE (initially provided on MAINT\193).

  It converts data in either direction, between -
     TOD-format  (8-bytes in the format returned by the STCK instruction
     USA-format  (16-characters in the format MM/DD/YYHH:MM:SS)

  It is intended to be called as  a REXX function only, and any other use
  will cause a  nasty CMS ABEND.  Even such  usually-benign problems such
  as  invalid-data  will  cause  a  CMS   abend.   This  is  not  a  real
  user-friendly program.  Nor was Y2K-compliancy  a high priority in this
  program -

    TOD2USA conversion is  OK, altho  a 2-digit  year is  returned, which
            must  be massaged  into a  4-digit  year, but  at least  it's
            do-able.

    USA2TOD conversion  is  restricted  to  the  100-year  interval  from
            1942-09-17@23:53:48 thru 2042-09-17@23:53:47, and all 2-digit
            years are squeezed into that interval.

  It  appears that  this module  is closely  related to  CP's "SET  VTOD"
  command;   except that  the documented  range of  valid dates  for "SET
  VTOD" is from 1942-01-01 thru 2041-12-31.

  To convert from TOD to SORTED (which has a 4-digit year, ccyymmdd) --
     tod_val = '8000000000000000'x          /* for example?????*/
     Parse Value TOD('C',tod_val)  With  mdy +8 hms +8
     ymd     = TRANSLATE('YyMmDd',mdy,'Mm/Dd/Yy')
     y4md    = 19 + ((tod_val>>'B0'x)  &  (ymd<<'420918'))  || ymd
     Say     y4md hms

  To convert from USA to TOD --
     tod_usa = DATE('U')  ||  TIME('N')     /* for example?????*/
     Parse Value TOD('T',tod_usa)  With  tod_valu
     Numeric Digits 20+3
     Say     '8 bytes hex =' C2X(tod_valu)  ,
             '    up to 20 decimal digits =' C2D(tod_valu)
```

::

The hardware's TOD-clock contains an 8-byte value indicating the elapsed time since 1999-01-01 at 00:00:00; and is expressed as a 64-bit number of "TOD Clock Units". A TOD Clock Unit is 1/4096000000 of a second - or; each micro-second contains 4096 TOD-CU's.

In converting this TOD value into "readable" format of date and time, the user must allow for a maximum of ten decimal places of accuracy in expressing the number of seconds. Note, tho, that most CPU's use something less than this degree of accuracy. The 9121, for example, seems to leave the right-most 12 bits as zero, which means that each CHANGE in the TOD-value represents a little less than 1/1000000 of a second (or a micro-second).

The maximum decimal number that may be expressed in 64 bits is (2**64)-1, or the 20-digit number 18446744073709551615. The time-range therefore, from 0 to all FF's, is slightly less than 52125 days. Since our base date is 1900-01-01, then the maximum date that may be expressed in this 8 bytes is 2042-09-17 at 23:53:47.3704959997.

Typically, (BUT NOT UNIVERSALLY), this time is a reflection of UTC (frequently called GMT , or "Z") time. It it YOUR responsibility to perform any necessary UTC-offset adjustment.

	seconds	TU's (dec)	TU's (hex)
TOD-Clock-Unit	0.0000000002	1	0001
MicroSecond	0.000001	4,096	1000
MilliSecond	0.001	4,096,000	3E 8000
Second	1.0	4,096,000,000	F424 0000
Minute	60.0	245,760,000,000	39 3870 0000
Hour	3,600.0	14,745,600,000,000	D69 3A40 0000
12Hours	43,200.0	176,947,200,000,000	A0EE BB00 0000
Day	86,400.0	353,894,400,000,000	1 41DD 7600 0000
Week	604,800.0	2,477,260,800,000,000	8 CD0E 3A00 0000
365Days	31,536,000.0	12,917,145,600,000,000	2D E413 5300 0000
366Days	31,622,400.0	12,952,535,040,000,000	2E 0443 1200 0000
Epoch	4,503,599,627._	18,446,744,073,709,551,615	FFFF FFFF FFFF FFFF

```
  A number  of calendaring  schemes give each  date a  sequential number,
  making it very  easy to do date calculations.  The  schemes differ only
  in where day "0" occurs;   REXX's BaseDate uses 0001-01-01, the 360-TOD
  epoch uses  1900-01-01, astronomer's Julian date  uses MINUS 4712-01-01
  (4713-01-01 BCE).  For  reference purposes, here are some  key dates in
  each "base":
                 yyyy-mm-dd     TOD   RxBase   AJulian
                 ----------   -----   ------   -------
                 1900-01-01       0   693595   2415021
                 1999-12-31   36523   730118   2451544
                 2000-01-01   36524   730119   2451545
                 2000-02-29   36583   730178   2451604
                 2042-09-17   52124   745719   2467145

\***********************************************************************/
/*--------------------------end of QVTOD.EXEC--------------------------*/
```

Chuck Meyer, Chuck Meyer Systems, Inc.
cmsi@attglobal.net
© Chuck Meyer

Check, find, & replace (CFR)

General description

CFR is a file support utility for professional use. It is not a replacement for XEDIT. Unlike XEDIT, which is used in conversational mode, CFR works in batch mode.

Mostly CFR is intended to manipulate very large CMS files with record lengths of up to 64 Kb.

CFR operates in three modes:

- *Check*: checks file contents to verify that a file contains only specified codes.
- *Find*: searches to determine if a file contains a given string.
- *Replace*: replaces occurrences of a search string in a file; if the replacement string is empty, the search string is excluded from the file.

CFR was written in Assembler and REXX. Its assembler code is optimized for speedy execution and high productivity.

Basic software

CFR was created in CMS with VM/SP Release 5.

Memory requirements

The size of CFR is 1912 bytes. To accelerate fixed-length file processing, a buffer area size of 1 Mb is allocated in execution time.

CFR EXEC usage

CFR EXEC has no parameters. The user selects mode and inputs parameters during interactive dialog.

The user is always prompted to enter a source file and the record number where processing should start. By default, processing will start with the first record.

Check

In Check mode, the EBCDIC code table is displayed by CFR. The user may set or unset chosen codes. Only set codes are considered valid when a file is verified.

Codes to be set may be entered as a characters or hexadecimal numbers from x'0' to x'FF'. Hexadecimal numbers must be preceded by X with no following apostrophe.

The syntax of a command line is

```
<operation> {<char> | <hex>} [<char> | <hex>]
```

where operation = S or U, char = a single character, and hex = a hexadecimal number in (x'0',x'ff').

Examples of setting and unsetting codes are as follows:

`S A`	set code A, C1 in hex
`U X0`	unset hex code 0
`S XF0 XF9`	set all codes in range 0-9, F0-F9 in hex
`U A Z`	unset all codes in range A-Z, C1-E9 in hex

Find

Find mode determines whether a file contains a given text or hexadecimal string. The search is terminated after finding the first occurrence of that string.

Examples of searching for a string are as follows:

`ABC`	char string
`X010203`	hex string

Replace

Replace mode replaces or excludes all occurrences of a given string. The search string and replacement string may differ in length. If the replacement is length zero, an empty string, then all occurrences of the search string are excluded from the source file.

An example of search and replace follows:

Search	Replacement	Action
123	X000102	replace, one-to-one, no file size changes
ABC	<none>	exclude, possible prompt for a pad
X00	X404040	replace, possible truncation of a records

If the replacement string is longer than the search string, the resulting record is truncated from the right, up to the record size of the source file.

If the replacement string is shorter than the search string and the source file has a fixed record format, then a padding character must be specified. The padding character is a single character or a hexadecimal number.

Examples of padding character declaration are:

<none>	blank, x'40'
0	0, x'F0'
X0	not displayable code, x'00'
X40	blank, x'40'

Note: Replace mode creates a new file containing the replacements. The new file may be larger than the source file. This fact must be considered when a minidisk is specified for the target file.

CFR getting ready

CFRINST EXEC should be used to generate the CFR MODULE on disk A.

CFR distribution material

CFR EXEC, INSTALLATION

```
/*****************************************************************/
/***                                             ***         ***/
/*** CFRINST           generate CFR MODULE       ***  DG"99  ***/
/***                                             ***         ***/
/*****************************************************************/
/***   SIZE 00048  VER 1.0 MOD 000  TIME 19:32:42 DATE 09/07/99   ***/
/*****************************************************************/

  HI = '1DF8'X
  LO = '1DF0'X
  CLRSCRN
  DO 11
    SAY
  END
  MESSAGE = 'user request'
SAY'--- Start CFR MODULE generation - reply Y or N'HI TIME(L)LO
  PULL REPLY
  IF REPLY ^= 'Y' THEN
  SIGNAL ERROR
  SET CMSTYPE HT
  STATE CFR MODULE A
  SAVE_RC = RC
  SET CMSTYPE RT
  IF SAVE_RC = 0 THEN
  DO
    SAY '--- CFR MODULE found on disk A'HI TIME(L)LO
    SAY '--- Replace CFR MODULE A - reply Y or N'HI TIME(L)LO
    PULL REPLY
    IF REPLY ^= 'Y' THEN
    SIGNAL ERROR
  END
  SET CMSTYPE HT
  SIGNAL ON ERROR
  MESSAGE = 'error when assemble' CFR
  ASSEMBLE CFR
  ERASE CFR LISTING A
  MESSAGE = 'error when load' CFR
  LOAD CFR '(' NOMAP NOLIBE
  MESSAGE = 'error when genmod' CFR
  GENMOD
  ERASE CFR TEXT A
  SIGNAL OFF ERROR
  SET CMSTYPE RT
  SAY '--- CFR MODULE generated successfully'HI TIME(L)LO
  EXIT
ERROR:
  SET CMSTYPE RT
SAY '--- CFR MODULE not generated due to' MESSAGE HI TIME(L)LO
```

CFR EXEC

```
/**********************************************************************/
/***                                                  ***         ***/
/*** CFR             check, find & replace            ***  DG"99  ***/
/***                                                  ***         ***/
/**********************************************************************/
/***   SIZE 00317  VER 1.0 MOD 000  TIME 19:54:24 DATE 09/07/99   ***/
/**********************************************************************/

  HI = '1DF8'X
  LO = '1DF0'X
  CLRSCRN
  DO 10
    SAY
  END
  SAY '>>>---> Check, find & replace -'
  SAY
  SAY '        Select mode ---------'
  SAY
  SAY '        1 - Check'
  SAY '        2 - Find'
  SAY '        3 - Replace'
  SAY '------- Enter 1, 2, or 3'
  PULL MODE .
  IF MODE = '' ! VERIFY(MODE, '123') ^= 0 THEN
  EXIT
  IF LEFT(MODE, 1) = '1' THEN
  MODE = 'C'
  ELSE
  IF LEFT(MODE, 1) = '2' THEN
  MODE = 'F'
  ELSE
  IF LEFT(MODE, 1) = '3' THEN
  MODE = 'R'
  FLR = ' '
  CLRSCRN
  DO 10
    SAY
  END
  DO FOREVER
    SAY '--- Enter source file - reply FN FT FM or 0/exit/'
    PULL FN FT FM
    IF FN = '0' THEN
    EXIT
    IF FM ^= '' THEN
    LISTFILE FN FT FM '(STACK ALL'
    IF QUEUED() = 1 THEN
    LEAVE
    SAY '--- File' FN FT FM 'not found'
  END
  PULL . . . RECFM . RECORDS .
  SAY '--- Enter start record number or none to process all records'
  PULL START .
  IF LENGTH(START) > 0 THEN
    IF VERIFY(START, '0123456789') > 0 THEN
  EXIT
  ELSE
  START = MIN(START, RECORDS)
  IF MODE ^= 'C' THEN
  DO FOREVER
    SAY '--- Enter search string'
    PULL FND
    IF LENGTH(FND) = 0 THEN
    ITERATE
    IF SUBSTR(FND, 1, 1) = 'X' THEN
    DO
      FND = SUBSTR(FND, 2)
      IF VERIFY(FND, '0123456789ABCDEF') > 0 THEN
      DO
        SAY '>>> Errors in hexadecimal data'
        ITERATE
      END
      ELSE
      FND = X2C(FND)
    END
    SAY 'Cha['FND']'
    HEX_R = C2X(FND)
    CALL HEX_GEN
    SAY '--- Enter 1/Yes/ to process'
    PULL ANS .
    IF ANS = 1 THEN
    LEAVE
  END
  IF MODE = 'R' THEN
  DO
    CLRSCRN
    DO 10
      SAY
    END
    DO FOREVER
      SAY '--- Enter replacement string'
      PULL REP
      IF SUBSTR(REP, 1, 1) = 'X' THEN
      DO
        REP = SUBSTR(REP, 2)
        IF VERIFY(REP, '0123456789ABCDEF') > 0 THEN
        DO
          SAY '>>> Errors in hexadecimal data'
          ITERATE
        END
        ELSE
        REP = X2C(REP)
      END
      IF LENGTH(REP) = 0 THEN
      SAY '    Found occurences will be excluded'
      ELSE
      DO
        SAY 'Cha['REP']'
        HEX_R = C2X(REP)
        CALL HEX_GEN
      END
      SAY '--- Enter 1/Yes/ to process'
      PULL ANS .
      IF ANS = 1 THEN
      LEAVE
    END
    CLRSCRN
    DO 10
      SAY
    END
    SAY '--- Enter target file - reply FN FT FM'
    PULL FN1 FT1 FM1
    IF FM1 = '' THEN
    EXIT
    IF LENGTH(FM1) ^= 0 THEN
    DO
      SET CMSTYPE HT
      MAKEBUF
      QUERY DISK FM1 '(' STACK LIFO
      PULL . . . STATUS .
      DROPBUF
      SET CMSTYPE RT
      IF STATUS ^= 'R/W' THEN
      DO
        SAY '--- Disk ' FM1 'is read/only'
        EXIT
      END
    END
    SET CMSTYPE HT
    LISTFILE FN1 FT1 FM1
    RC_SAVE = RC
```

```
    SET CMSTYPE RT
    IF RC_SAVE = 0 THEN
    DO
      SAY '--- File' FN1 FT1 FM1 'found - enter 1/Yes/ to erase'
      PULL ANS .
      IF ANS ^= '1' THEN
      EXIT
      ERASE FN1 FT1 FM1
    END
    IF RECFM = 'F' THEN
      IF LENGTH(FND) > LENGTH(REP) THEN
    DO FOREVER
      SAY '--- Enter padding char to fill record after replace'
      PULL FLR .
      IF FLR = '' THEN
      FLR = 'X40'
      IF SUBSTR(FLR, 1, 1) = 'X' THEN
      DO
        FLR = RIGHT(                                          ,
                 SUBSTR(FLR, 2, MIN(LENGTH(FLR)-1, 2)), 2, '0')
        FLR = X2C(FLR)
      END
      ELSE
      FLR = SUBSTR(FLR, 1, 1)
      SAY '    Fixed records will be filled with ' FLR '(char) ->',
                                      C2X(FLR) '(hex)'
      SAY '--- Enter 1/Yes/ to process with above setting'
      PULL ANS .
      IF ANS = 1 THEN
      LEAVE
    END
  END
  IF MODE = 'C' THEN
  DO
    MARK = COPIES('40'X, 256)
    HEX = '0123456789ABCDEF'
    SWITCH = 1
    DO FOREVER
      CALL SHOW
      IF SWITCH = 0 THEN
      DO
        SAY CENTER('Enter BLANK/continue/, 1/process/, 0/exit/',    ,
                    79, '+')
        PULL ANS .
        IF VERIFY(ANS, ' 10') = 0 THEN
        DO
          IF ANS = '0' THEN
          DO
            CLRSCRN
            EXIT
          END
          SWITCH = 1
          IF ANS = '1' THEN
          DO
            IF VERIFY(MARK, '40'X) = 0 THEN
            DO
              CLRSCRN
              DO 21
                SAY
              END
              SAY '--- Codes to check not found'
              SLEEP 5 SEC
              ITERATE
            END
            ELSE
            LEAVE
          END
        END
        ITERATE
      END
      SWITCH = 0
      SAY '--- Enter S/set/ or U/unset/ and CHAR/HEX or'              ,
               'range as CHAR CHAR or HEX HEX'LO
      PULL ACTION CODE_1 CODE_2
      IF ACTION = 'S' THEN
      SHOW_WITH = '+'
      ELSE
      SHOW_WITH = ' '
      I = 0
      IF LENGTH(CODE_2) > 0 THEN
      DO
        IF LENGTH(CODE_1) = 1 THEN
          IF LENGTH(CODE_2) = 1 THEN
        DO
          IF X2D(C2X(CODE_1)) ^> X2D(C2X(CODE_2)) THEN
          I = X2D(C2X(CODE_2)) - X2D(C2X(CODE_1)) + 1
          J = X2D(C2X(CODE_1)) + 1
        END
        IF LENGTH(CODE_1) > 1 THEN
          IF LENGTH(CODE_2) > 1 THEN
        DO
          CODE_1 = SUBSTR(CODE_1, 2)
          CODE_2 = SUBSTR(CODE_2, 2)
          IF DATATYPE(CODE_1 !! CODE_2, 'X') THEN
            IF X2D(CODE_1) ^> X2D(CODE_2) THEN
          I = X2D(CODE_2) - X2D(CODE_1) + 1
          J = X2D(CODE_1) + 1
        END
```

```
      END
      ELSE
      DO
        IF LENGTH(CODE_1) = 1 THEN
        DO
          I = 1
          J = X2D(C2X(CODE_1)) + 1
        END
        ELSE
        IF LENGTH(CODE_1) > 1 THEN
        DO
          CODE_1 = SUBSTR(CODE_1, 2)
          IF ^ DATATYPE(CODE_1, 'X') THEN
          I = 0
          ELSE
          DO
            I = 1
            J = X2D(CODE_1) + 1
          END
        END
      END
      IF I > 0 THEN
        IF J < 256 THEN
      MARK = OVERLAY(COPIES(SHOW_WITH, VALUE(I)), MARK, J)
    END
  END
  TAB = ''
  DO I = 1 TO 256
    IF SUBSTR(MARK, I, 1) = ' ' THEN
    TAB = TAB !! 'FF'X
    ELSE
    TAB = TAB !! '00'X
  END
  CLRSCRN
  DO 16
    SAY
  END
  CFR  MODE RIGHT(START, 8, '0') FN FT FM FN1 FT1 FM1
  IF RC ^= 0  THEN
  SAY '--- The above error caused CFR abend'
  SAY
  EXIT
HEX_GEN:
  REP_HEX_1 = ''
  REP_HEX_2 = ''
  DO I = 1 TO LENGTH(HEX_R) BY 2
    REP_HEX_1 = REP_HEX_1 !! SUBSTR(HEX_R, I, 1)
    REP_HEX_2 = REP_HEX_2 !! SUBSTR(HEX_R, I+1, 1)
  END
  SAY 'He1['REP_HEX_1'!'
  SAY 'He2['REP_HEX_2'!'
  RETURN
SHOW:
  CLRSCRN
  SAY 'Codes  >>> 1-63 <<<'
  SAY 'Hex 1' HI                                                   ,
    COPIES('0', 16)COPIES('1', 16)COPIES('2', 16)COPIES('3', 16)LO
  SAY 'Hex 2' HI COPIES(HEX, 4) LO
  SAY 'Chars'
  SAY LEFT('Check', 7) SUBSTR(MARK, 1, 64)
  SAY 'Codes  >>> 64-127 <<<'
  SAY 'Hex 1' HI                                                   ,
    COPIES('4', 16)COPIES('5', 16)COPIES('6', 16)COPIES('7', 16)LO
  SAY 'Hex 2' HI COPIES(HEX, 4) LO
  SAY 'Chars' HI XRANGE('40'X, '7F'X) LO
  SAY LEFT('Check', 7) SUBSTR(MARK, 65, 64)
  SAY 'Codes  >>> 128-191 <<<'
  SAY 'Hex 1' HI                                                   ,
    COPIES('8', 16)COPIES('9', 16)COPIES('A', 16)COPIES('B', 16)LO
  SAY 'Hex 2' HI COPIES(HEX, 4) LO
  SAY 'Chars' HI XRANGE('80'X, 'BF'X) LO
  SAY 'Codes  >>> 192-256 <<<'
  SAY LEFT('Check', 7) SUBSTR(MARK, 129, 64)
  SAY 'Hex 1' HI                                                   ,
    COPIES('C', 16)COPIES('D', 16)COPIES('E', 16)COPIES('F', 16)LO
  SAY 'Hex 2' HI COPIES(HEX, 4) LO
  SAY 'Chars' HI XRANGE('C0'X, 'FF'X) LO
  SAY LEFT('Check', 7) SUBSTR(MARK, 193, 64)
  RETURN
```

CFR ASSEMBLE

```
**********************************************************************
****                                                ***         ****
**** CFR            check, find & replace           ***  DG"99  ****
****                                                ***         ****
**********************************************************************
****   SIZE 00367  VER 1.0 MOD 000  TIME 19:23:59 DATE 09/07/99   ****
**********************************************************************
*                                                                    *
CFR      CSECT
         USING *,12
         ST    14,BACK2CMS
         MVC   REQ(1),8(1)
         LA    11,DCBREP
         LA    10,DCB
         USING FSCBD,10
         PACK  DOUBLE(8),16(8,1)
         CVB   15,DOUBLE
         ST    15,FSCBAITN
         MVC   DCB+8(24),24(1)
         CLI   REQ,C'R'
         BNE   GETREXX
         MVC   DCBREP+8(18),48(1)
GETREXX  EQU   *
         CLI   REQ,C'R'
         BNE   GETFND
         MVC   REXXID(3),=CL3'FLR'
         BAL   2,REXXDATA
         MVC   FILLER(1),REXXVAL
         MVC   REXXID(3),=CL3'REP'
         BAL   2,REXXDATA
         L     15,VALUELEN
         STH   15,LENREPL
         LTR   15,15
         BZ    GETFND
         BCTR  15,0
         STC   15,MVCR+1
         STC   15,MOVEREP+1
MVCR     MVC   REPLACE(64),REXXVAL
GETFND   EQU   *
         CLI   REQ,C'C'
         BE    GETTAB
         MVC   REXXID(3),=CL3'FND'
         BAL   2,REXXDATA
         L     15,VALUELEN
         STH   15,LENFIND
         BCTR  15,0
         STH   15,LENFJMP
         STC   15,MVCF+1
         STC   15,CLC+1
MVCF     MVC   FIND(64),REXXVAL
         B     OPEN
GETTAB   EQU   *
         MVC   REXXID(3),=CL3'TAB'
         BAL   2,REXXDATA
OPEN     EQU   *
         FSOPEN FSCB=DCB,ERROR=RET,FORM=E
         L     15,FSCBSIZE
         ST    15,LRECL
         MVC   DCBREP+X'24'(1),FSCBFV
         CLI   FSCBFV,C'F'
         BNE   CNV2DBL
         MVC   ALLOC(4),=A(128*512)
         B     COUNTBUF
CNV2DBL  EQU   *
         LA    15,7(15)
         SRL   15,3
         ST    15,ALLOC
COUNTBUF EQU   *
         SR    0,0
         L     1,ALLOC
         SLL   1,3
         D     0,LRECL
         ST    1,FSCBANIT
         USING FSCBD,11
         ST    1,FSCBANIT
         SR    0,0
         M     0,LRECL
         DROP  11
         USING FSCBD,10
         ST    1,FSCBSIZE
         USING FSCBD,11
         ST    1,FSCBSIZE
         D     0,=F'256'
         LTR   0,0
         BZ    SKIP
         BCTR  0,0
         STC   0,TRTREST+1
SKIP     EQU   *
         STM   0,1,REST
         L     0,ALLOC
         CLI   REQ,C'R'
         BNE   ASIS
         SLL   0,1
ASIS     EQU   *
         DMSFREE DWORDS=(0),TYPE=USER,AREA=HIGH,ERR=RET
         DROP  11
         USING FSCBD,10
         ST    1,FSCBBUFF
         L     0,FSCBSIZE
         AR    0,1
         USING FSCBD,11
         ST    0,FSCBBUFF
         DROP  11
         USING FSCBD,10
READNEXT EQU   *
         FSREAD FSCB=DCB,FORM=E
         CLI   FLAG,X'00'
         BE    JUMP
         MVI   FLAG,X'00'
         XC    FSCBAITN(4),FSCBAITN
JUMP     EQU   *
         LTR   15,15
         BNZ   CLOSE
         L     9,FSCBBUFF
         CLI   REQ,C'C'
         BNE   CHECKF
         SR    1,1
         CLI   FSCBFV,C'V'
         BE    RECALC
         C     0,FSCBSIZE
         BNE   RECALC
         LM    14,15,REST
         B     CUTDCMD
RECALC   EQU   *
         LR    15,0
         SR    14,14
         D     14,=F'256'
         LTR   14,14
         BZ    CUTDCMD
         BCTR  14,0
         STC   14,TRTREST+1
CUTDCMD  EQU   *
         LTR   15,15
         BZ    CHKREST
DOLOOP   EQU   *
         TRT   0(256,9),TAB
         LTR   1,1
         BNZ   DISPMSG
         LA    9,256(9)
         BCT   15,DOLOOP
CHKREST  EQU   *
         LTR   14,14
         BZ    READNEXT
TRTREST  TRT   0(0,9),TAB
         LTR   1,1
         BZ    READNEXT
DISPMSG  EQU   *
         WTO   '--- Invalid characters FOUND'
         B     FREEMAIN
CHECKF   EQU   *
         CLI   FSCBFV,C'V'
         BE    SETVLEN
         C     0,FSCBSIZE
         BNE   CNTRECS
         L     2,FSCBANIT
         B     STARTCYC
CNTRECS  EQU   *
         SRDL  0,32
         D     0,LRECL
         LR    2,1
         B     STARTCYC
SETVLEN  EQU   *
         LA    2,1
STARTCYC EQU   *
         USING FSCBD,11
         L     7,FSCBBUFF
NEXTREC  EQU   *
         LR    3,9
         LR    6,7
         LR    15,3
         LR    14,7
         ST    14,ORIGRG14
         A     6,LRECL
         CLI   FSCBFV,C'F'
         BNE   ITSVFMT
         L     0,LRECL
ITSVFMT  EQU   *
         LA    8,1
         AR    9,0
         SH    9,LENFIND
NEXTPOS  EQU   *
CLC      CLC   0(0,3),FIND
         BNE   FINDCYC
         CLI   REQ,C'R'
         BE    SUBST
         WTO   '--- Search string FOUND'
         B     FREEMAIN
SUBST    EQU   *
         LR    1,3
         SR    1,15
         LTR   1,1
         BZ    REPONLY
         LR    5,1
         LR    0,14
         LR    4,15
         AR    1,7
         LR    14,6
         SR    14,1
         BP    LENOK
         AR    5,14
         LTR   5,5
         BNP   CONTINUE
LENOK    EQU   *
         LR    1,5
         AR    7,1
         MVCL  0,4
         CR    6,7
         BE    CONTINUE
REPONLY  EQU   *
         AH    3,LENFJMP
         CLI   LENREPL+1,X'00'
         BE    NOTHTOMV
         CR    6,7
         BNH   NOTHTOMV
         LR    5,6
         SR    5,7
         CH    5,LENREPL
         BH    SPACEOK
         SH    5,LENREPL
         LTR   5,5
         BZ    CONTINUE
         LPR   5,5
         EX    5,MOVEREP
         AR    7,5
         B     CONTINUE
SPACEOK  EQU   *
MOVEREP  MVC   0(64,7),REPLACE
NOTHTOMV EQU   *
         AH    7,LENREPL
         LA    15,1(3)
         LR    14,7
FINDCYC  EQU   *
         BXLE  3,8,NEXTPOS
LEAVE    EQU   *
         AH    9,LENFIND
         CLI   REQ,C'R'
         BNE   READNEXT
         LR    5,9
         C     14,ORIGRG14
         BE    MOVEASIS
         SR    5,15
         LR    1,6
         SR    1,7
         CR    1,5
         BE    MOVELONG
         BH    CHECKFV
         LR    5,1
         B     MOVELONG
CHECKFV  EQU   *
         CLI   FSCBFV,C'F'
         BE    FILLIT
         LR    1,5
FILLIT   EQU   *
         ICM   5,8,FILLER
         B     MOVELONG
MOVEASIS EQU   *
         SR    5,15
         LR    1,5
MOVELONG EQU   *
         LR    0,14
         LR    4,15
         LR    15,1
         MVCL  0,4
         LR    7,0
         B     PROCNEXT
CONTINUE EQU   *
         AH    9,LENFIND
PROCNEXT EQU   *
         BCT   2,NEXTREC
         DROP  11
         USING FSCBD,10
         L     0,FSCBNORD
         CLI   FSCBFV,C'F'
         BE    CNTBLOCK
         S     7,ORIGRG14
         LR    2,7
         B     SETFSCB
CNTBLOCK EQU   *
         CLC   FSCBSIZE(4),FSCBNORD
         BE    WRITE
         LR    2,0
         SRDL  0,32
         D     0,LRECL
         USING FSCBD,11
         ST    1,FSCBANIT
SETFSCB  EQU   *
         ST    2,FSCBSIZE
WRITE    EQU   *
         FSWRITE FSCB=DCBREP,ERROR=FREEMAIN,FORM=E
         B     READNEXT
CLOSE    EQU   *
         CLI   REQ,C'C'
         BNE   FMSG
         WTO   '--- Invalid characters NOT FOUND'
         B     FREEMAIN
FMSG     EQU   *
         CLI   REQ,C'F'
         BNE   RMSG
         WTO   '--- Search string NOT FOUND'
         B     FREEMAIN
RMSG     EQU   *
         WTO   '--- All occurrences are replaced'
FREEMAIN EQU   *
         FSCLOSE FSCB=DCB
         CLI   REQ,C'R'
         BNE   DMSFREE
         FSCLOSE FSCB=DCBREP
DMSFREE  EQU   *
         L     0,ALLOC
         DROP  11
         USING FSCBD,10
         L     1,FSCBBUFF
         CLI   REQ,C'R'
         BNE   DMSFRET
         SLL   0,1
DMSFRET  EQU   *
         DMSFRET DWORDS=(0),LOC=(1)
RET      EQU   *
         L     11,BACK2CMS
         BR    11
REXXDATA EQU   *
         LA    0,REXXPARM
         LA    1,COMMAND
         ICM   1,8,=X'02'
         SVC   202
         DC    AL4(1)
         LTR   15,15
         BM    RET
         BR    2
DOUBLE   DS    D
REXXPARM DC    A(COMMAND)
         DC    8X'00'
         DC    A(REQBLOK)
COMMAND  DC    CL8'EXECCOMM'
REQBLOK  DC    2A(0)
REQUEST  DC    C'F'
RETCODE  DC    3X'00'
BUFSIZE  DC    F'256'
         DC    A(REXXID)
NAMELEN  DC    F'3'
         DC    A(REXXVAL)
```

```
VALUELEN DS    F
EXTPLIST EQU   *
         DC    A(COMMVERB)
         DC    3A(0)
COMMVERB DC    CL8'SUBCOM'
BACK2CMS DS    F
ALLOC    DS    F
LRECL    DS    F
REST     DS    2F
ORIGRG14 DS    F
LENFIND  DS    H
LENREPL  DS    H
LENFJMP  DS    H
FIND     DS    8CL8
REPLACE  DS    8CL8
REXXVAL  DS    16CL16
REXXID   DS    CL3
REQ      DS    CL1
FILLER   DS    CL1
FLAG     DC    X'FF'
TAB      EQU   REXXVAL
         LTORG
DCB      FSCB  FORM=E
DCBREP   FSCB  FORM=E
         FSCBD
         END   CFR
```

© Dobrin Goranov
dg099@hotmail.com

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