139 VM update

March 1998

In this issue

VM Update

Published by
Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38030
From USA: 01144 1635 38030
E-mail: xephon@compuserve.com

North American office
Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75067
USA
Telephone: 940 455 7050

Australian office
Xephon/RSM
GPO Box 6258
Halifax Street
Adelaide, SA 5000
Australia
Telephone: 088 223 1391

Contributions
Articles published in *VM Update* are paid for at the rate of £170 ($250) per 1000 words for original material. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our *Notes for Contributors*.

Editor
Robert Burgess

Disclaimer
Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon 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 Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever 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.

Subscriptions and back-issues
A year's subscription to *VM Update*, comprising twelve monthly issues, costs £170.00 in the UK; $255.00 in the USA and Canada; £176.00 in Europe; £182.00 in Australasia and Japan; and £180.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1990 issue, are available separately to subscribers for £14.50 ($21.50) each including postage.

VM Update on-line
Code from *VM Update* can be downloaded from our Web site at http://www.xephon.com; you will need the user-id shown on your address label.

Printed in England.

Interface to the MOVEFILE command

GENERAL DESCRIPTION

The FILETAPE EXEC provides an interactive interface to the CMS MOVEFILE command. It provides an easy way to set all parameter values when copying a file to or from tape. At the same time FILETAPE prevents the user from making simple errors in FILEDEF definitions, which are difficult to detect.

The CMS MOVEFILE uses common tape format for CMS, MVS, and VSE. So FILETAPE may be useful if production data needs to be interchanged between different sites.

FILETAPE is written in REXX.

FILETAPE EXEC USAGE

FILETAPE does not have set parameters. Users should follow the interactive dialogue to define the parameters for input and output file definitions. When data errors are entered, the state of the dialogue remains unchanged to prevent accidental termination and repeated invoking of the FILETAPE EXEC. To give the user the opportunity to cancel the dialogue and exit FILETAPE, the breakpoints are defined between different dialogue states.

To ensure error free FILEDEF declarations for MOVEFILE, the following limitations and built-in extensions are used in FILETAPE:

- Write-to-tape files have a blocksize between 20 and 32,760 bytes for fixed record format, and a recordsize between 20 and 32,752 bytes for variable record format.
- Read-from-tape non-standard labels (NL) files are always processed with maximum acceptable input buffers, to provide independence from the written records' blocksize.
- All header standard labels (SL) are processed to ensure read-by-name copying for SL tape files.

- A set of possible file record format conversions are supported.
- SL tapes can be read as NL tapes, and correct tape positioning is made to support this.
- Tape initialization is provided to change density and volume labels.

COPYING FILES FROM DISK TO TAPE

There are several hints to help when copying from disk to tape:

- Multi-volume tape files are supported only on SL tapes – SL labels are forced by default for these files.
- The density may be changed only when the tape is at load point – it may be set only when file number 1 is to be written.
- The tape should be reset if it is new, or density and volume labels are changed.
- The record format may be changed if necessary.
- Note that ‘max block’ should be selected when the output block size is specified – this option maximizes the tape capacity.
- If an empty line is entered as an answer to the message ‘— To accept the above - enter 1/OK/ (X-exit)’ the dialogue goes back to the preceding breakpoint and the user can correct the values set.

COPYING FILES FROM TAPE TO DISK

There are several hints to help when copying from tape to disk:

- For SL tapes, the volume labels are checked. If these do not match the values entered, copying will be terminated.
- SL files are read by name. File blocksize, record size, and file record format, are obtained from HDR2 label, and imported in file definition. After this, the tape is positioned correctly to allow MOVEFILE to process the HDR1 label.
- SL tapes may be read as NL tapes – the user selects ‘Ignore

standard labels' by entering 1.

- Generally, NL files are read without any information about their blocksize, and only record format and record size need be known. Even record size is not needed for variable record format files, which are accessed with specification V (32600, 32556).
- The truncation size of the record must be specified when variable record files are converted to fixed records.

FILETAPE EXEC

```
/*********************************************************************/
/***                                                        ***      ***/
/*** FILETAPE          copyfile to/from tape                 ***  DG'97  ***/
/***                                                        ***      ***/
/*********************************************************************/
/***   SIZE ØØ5Ø5  VER 1.Ø MOD ØØ                                    ***/
/*********************************************************************/

  HI = '1DF8'X
  LO = '1DFØ'X
  FI '*' CLEAR
  CLRSCRN
  DO 16
    SAY
  END
  SAY'— Copy'HI'TO'LO'tape/1/ or'HI'FROM'LO'tape/2/ -'              ,
                                    'select'HI'1'LO'or'HI'2'LO
  PULL MODE .
  IF VERIFY(MODE, '12') ¬= Ø | LENGTH(MODE) = Ø THEN
  EXIT
  DO FOREVER
    SAY'— Enter file name -'HI'FN FT FM'LO
    PULL FN FT FM
    IF FM ¬= '' THEN
    LEAVE
  END
  SET CMSTYPE HT
  STATE FN FT FM
  RC_SAVE = RC
  SET CMSTYPE RT
  IF MODE = 1 THEN
    IF RC_SAVE ¬= Ø THEN
  DO
    SAY'— Not found'HI FN FT FM LO
```

```
    EXIT
  END
  ELSE
  DO
    LISTFILE FN FT FM '(' STACK AL
    PULL . . . FMT RECS .
    IF RECS > 32760 & FMT = 'F' | RECS > 32752 & FMT = 'V' THEN
    DO
     SAY'    The record length of'HI||RECS 'is not supported'LO
     EXIT
   END
   TIT =  CENTER(HI||FN FT FM' >>>-> TAPE'LO, 52)
  END
  ELSE
  DO
    IF RC_SAVE = 0 THEN
    DO
      SAY'- Found'HI FN FT FM LO'-> enter'HI'1/Yes/'LO||           ,
                                              'to overwrite'
      PULL ANS .
      IF ANS ¬= 1 THEN
      EXIT
    END
    ELSE
    DO
      IF ¬ DATATYPE(FM, 'M') THEN
      EXIT
      MAKEBUF
      QUERY DISK FM '(' STACK LIFO
      PULL . . . MODE .
      DROPBUF
      IF MODE ¬= 'R/W' THEN
      DO
      SET CMSTYPE RT
      SAY '- Disk'HI FM LO'is'HI'read/only'LO'or'HI'not found'LO
      EXIT
    END
  END
  TIT = CENTER(HI'TAPE >>>-> 'FN FT FM||LO, 49)
END
DO FOREVER
  CLRSCRN
  SAY
  SAY CENTER(COPIES('-', 30), 50)
  SAY TIT
  SAY CENTER(COPIES('-', 30), 50)
  SAY
  SAY
```

```
IF MODE = '1' THEN
DO
  SAY'— To write'HI'MULTIVOLUME'LO'tape file enter'HI'1/0¬/'LO
   PULL LABEL .
 END
 ELSE
LABEL = ''
IF LABEL ¬= '1' THEN
DO
  SAY'— Standard tape label - enter'HI'1/Yes/'LO
  PULL LABEL .
END
ELSE
SAY'   'HI'SL'LO'forced to support multivolume tape file'
IF LABEL ¬= '1' THEN
SL = 'NL'
ELSE
DO
  IF MODE ¬= '1' THEN
  DO
    SAY'— Ignore standard labels - enter'HI'1/Yes/'LO
    PULL ANS .
    IF ANS = '1' THEN
    DO
      SL = 'BLP'
      SAY'    Standard labels to be bypassed'
    END
  END
  IF SL ¬= 'BLP' THEN
  DO
    SAY'— Enter tape volume label'
    PULL VOL1 .
  END
END
DO FOREVER
  SAY'— Enter file number on the tape'
  PULL FILE_NO .
  IF DATATYPE(FILE_NO, 'N') THEN
  LEAVE
END
FILE_NO = MAX(1, FILE_NO)
IF MODE = '1' THEN
DO
  IF FILE_NO = 1 THEN
  DO
    SAY'— Specify density - select'HI'1/1600/ or 2/6250/'LO
    PULL DN
    IF DN = '2' THEN
```

```
     DN = '6250'
    ELSE
    DN = '1600'
    IF MODE = '1' THEN
    DO
      SAY'- Does the tape reset as'HI||SL||LO||'with'HI||DN||LO,
                                      'bpi- enter'HI'1/Yes/'LO
      PULL ANS .
      IF ANS ¬= '1' THEN
      DO
        CLRSCRN
        DO 7
        SAY
        END
        SAY'    The tape is to be reset - enter'HI'1/OK/'LO
        PULL ANS .
        IF ANS ¬= '1' THEN
        EXIT
        SAY'   Wait for tape rewind      at' TIME('L')
        TAPE REW
        SAY'   Wait for write tape marks at' TIME('L')
        TAPE WTM 33
        SAY'   Wait for tape rewind      at' TIME('L')
        TAPE REW
        IF SL = 'SL' THEN
        DO
          SAY'   Wait for write VOL1 label at' TIME('L')
          TAPE WVOL1 VOL1
          SAY'   Wait for tape rewind      at' TIME('L')
          TAPE REW
        END
      END
    END
  END
END
IF MODE = '1' THEN
ACTION = 'write to'
ELSE
ACTION = 'read from'
IF SL = 'BLP' THEN
FILE_NO = (FILE_NO * 3) - 1
IF VOL1 = 'VOL1' THEN
VOL_TX = ' '
ELSE
VOL_TX = HI||VOL1||LO
 IF DN = 'DN' THEN
 DN_TX = ''
 ELSE
```

```
  DN_TX = ' with'HI||DN||LO'bpi'
  SAY COPIES('-', 79)
  SAY'    To be' ACTION||HI||SL||LO'tape'||VOL_TX||'file'||        ,
                                              HI||FILE_NO||LO||DN_TX
  SAY'— To accept the above enter'HI'1/OK/'LO'(X-exit)'
  PULL ANS .
  IF ANS = 'X' THEN
  EXIT
  IF ANS = '1' THEN
  LEAVE
END
IF SL = 'SL' & MODE ¬= '1' THEN
W_DEF = HI'SL defined'LO
ELSE
DO TRY_CONFIG = 1 BY 1
  CLRSCRN
  DO 2
    SAY
  END
  SAY CENTER(COPIES('-', 3Ø), 5Ø)
  SAY TIT
  SAY CENTER(COPIES('-', 3Ø), 5Ø)
  SAY
  IF MODE = '1' THEN
  DO
    FMT_O = FMT
    SAY '— Change record format - enter'HI'1/Yes/'LO
    PULL ANS .
    IF ANS = '1' THEN
    DO
      IF FMT = 'F' THEN
      SAY '— Write'HI'F/fixed/'LO'as'HI'V/variable/'LO||         ,
                                        '- enter'HI'1/OK/'LO
      ELSE
      SAY '— Write'HI'V/variable/'LO'as'HI'F/fixed/'LO||         ,
                                        '- enter'HI'1/OK/'LO
      PULL ANS .
      IF ANS ¬= '1' THEN
      SAY '    The record format is not changed'
  ELSE
      DO
        IF FMT = 'F' THEN
        FMT_O = 'V'
        ELSE
        FMT_O = 'F'
      END
    END
    SAY'— Enter'HI'block size'LO'or'||HI'empty line/max block/'LO
```

```
    PULL BLK .
    IF BLK ¬= '' THEN
      IF ¬ DATATYPE(BLK, 'N') THEN
    ITERATE TRY_CONFIG
    ELSE
    IF FMT_O = 'V' THEN
    BLK = MAX(MIN(32760, MAX(RECS + 8, BLK)), 20)
    ELSE
    BLK = MAX(MIN((32760 % RECS) * RECS, RECS * BLK) % RECS,       ,
                20 % RECS + MIN(1, 20 // RECS))
    ELSE
    IF FMT_O = 'V' THEN
    BLK = 32760
    ELSE
    BLK = (32760 % RECS) * RECS
    IF FMT_O = 'V' THEN
    DO
      RECS_O = BLK - 4
      W_DEF_D = HI||'V ('BLK','||RECS_O||')'LO
      O_TXT = 'VARIABLE'
    END
    ELSE
    DO
      RECS_O = RECS
      W_DEF_D = HI||'F ('BLK','||RECS_O||')'LO
      O_TXT = LEFT('FIXED', 8)
    END
    IF FMT = 'V' THEN
    I_TXT = 'VARIABLE'
    ELSE
    I_TXT = LEFT('FIXED', 8)
    W_DEF = HI||FMT '('||RECS||')'LO
    SAY'          To be  read from disk'HI||I_TXT||LO'records -'||  ,
                                                          W_DEF
  SAY'    Configure to write to tape'HI||O_TXT||LO||             ,
                                          'blocks  -'|| W_DEF_D
  SAY'— To accept the above enter'HI'1/OK/'LO'(X-exit)'
  PULL ANS .
  IF ANS = 'X' THEN
  EXIT
  ELSE
  IF ANS = '1' THEN
  LEAVE TRY_CONFIG
  ELSE
  ITERATE TRY_CONFIG
END
ELSE
DO
```

```
 SAY'— Enter tape record format -'HI'F/fixed/'LO'or'          ,
                                              ||HI'V/variable/'LO
 PULL ANS .
 IF ANS = 'F' THEN
 FMT = 'F'
 ELSE
 FMT = 'V'
 DO CHECK_REQ = 1 BY 1
   SAY'— Enter tape 'HI'record size'LO
   PULL RECS .
   IF RECS ¬= '' THEN
     IF DATATYPE(RECS, 'N') THEN
       IF FMT = 'F' THEN
   DO
     RECS = MIN(32760, MAX(RECS, 1))
     BLK = (32760 % RECS) * RECS
     FMT = 'FB'
     FMT_O = 'F'
     RECS_O = RECS
     LEAVE CHECK_REQ
   END
   ELSE
   DO
     RECS = 32752
     BLK = 32760
     FMT = 'VB'
     FMT_O = 'V'
     RECS_O = RECS
LEAVE CHECK_REQ
   END
 END
 SAY '— Change disk file record format - enter'HI'1/Yes/'LO
 PULL ANS .
 IF ANS = '1' THEN
 DO
   IF FMT_O = 'F' THEN
   SAY '— Write'HI'F/fixed/'LO'as'HI'V/variable/'LO||          ,
                                    '- enter'HI'1/OK/'LO
   ELSE
   SAY '— Write'HI'V/variable/'LO'as'HI'F/fixed/'LO||          ,
                                    '- enter'HI'1/OK/'LO
   PULL ANS .
   IF ANS ¬= '1' THEN
   SAY '    The record format is not changed'
   ELSE
   DO
     IF FMT_O = 'F' THEN
     DO
```

```
        FMT_O = 'V'
        RECS_O = 32752
      END
      ELSE
      DO
        FMT_O = 'F'
        DO FOREVER
          SAY '- Enter record size/record truncate size/'
          PULL ANS .
          IF DATATYPE(ANS, 'N') THEN
          LEAVE
        END
        RECS_O = MIN(RECS, ANS)
      END
    END
  END
  IF FMT = 'FB' THEN
  DO
    W_DEF = HI||'F ('BLK',')||RECS||')'LO
    I_TXT = LEFT('FIXED', 8)
  END
  ELSE
  DO
    W_DEF = HI||'V (32760,32756)'LO
    I_TXT = 'VARIABLE'
  END
  W_DEF_D = HI||FMT_O '('||RECS_O||')'LO
  IF FMT_O = 'F' THEN
  O_TXT = LEFT('FIXED', 8)
  ELSE
  O_TXT = 'VARIABLE'
  SAY COPIES('-', 79)
  SAY'    Configure to -'
  SAY'         read from tape'HI||I_TXT||LO'blocks  -'|| W_DEF
  SAY'          write to disk'HI||O_TXT||LO'records -'|| W_DEF_D
  SAY'- To accept the above enter'HI'1/OK/'LO'(X-exit)'
  PULL ANS .
  IF ANS = 'X' THEN
  EXIT
  IF ANS = '1' THEN
  LEAVE TRY_CONFIG
  ELSE
  ITERATE TRY_CONFIG
END
END
DO FOREVER
  CLRSCRN
  DO 7
```

```
      SAY
    END
    IF MODE = 1 THEN
    DO
      MSG = 'Ready to write'DN_TX||                                    ,
             HI||SL||LO||'tape'||VOL_TX||'file'HI||FILE_NO||LO
      MSG_EXT = 'on disk' W_DEF'>>>>>>> on tape' W_DEF_D
    END
    ELSE
    DO
      MSG = 'Ready to read 'HI||SL||LO||'tape'||VOL_TX||'file'         ,
                                              ||HI||FILE_NO||LO
      IF SL = 'SL' THEN
      MSG_EXT = 'on tape'HI'by labels'LO'>>>>>>>'                      ,
                                         'on disk'HI'by labels'LO
      ELSE
      MSG_EXT = 'on tape' W_DEF'>>>-> on disk' W_DEF_D
    END
  SAY COPIES('-', 78)
  SAY TIT
  SAY
  SAY MSG
  SAY MSG_EXT
  SAY COPIES('-', 78)
  SAY
  SAY'- To start copying enter'HI'1/OK/'LO'(X-exit)'
  PULL ANS .
  IF ANS = 'X' THEN
  EXIT
  IF ANS = '1' THEN
  LEAVE
END
IF MODE = 1 THEN
DO
  IF SL = 'SL' THEN
  DO
    SET CMSTYPE HT
    PUSH FN FT
    LABELDEF OUTMOVE FID '?' VOLID VOL1
    SET CMSTYPE RT
  END
  IF DN = 'DN' THEN
  DEN = ''
  ELSE
  DEN = 'DEN' DN
  FI OUTMOVE TAP1 SL FILE_NO '(' BLOCK BLK LRECL RECS_O RECFM FMT_O,
                                                        ||'B' DEN
  FI INMOVE DISK FN FT FM
```

```
  TXT = 'WERE COPIED TO TAPE'
  ERR_TXT = 'WERE NOT COPIED TO TAPE'
END
ELSE
DO
  IF SL = 'SL' THEN
  DO
    SET CMSTYPE HT
    ERASE @TAPE@ @LABELS@ A
    SET CMSTYPE RT
    FSF_NO = (FILE_NO - 1) * 3 - 1
    TAPE REW
  FI IN  TAP1 '(' LEAVE
  FI OUT DISK @TAPE@ @LABELS@ A
  MOVEFILE IN OUT
  EXECIO 1 DISKR @TAPE@ @LABELS@ A '(VAR BUF'
  DO FOREVER
    IF SUBSTR(BUF, 1, 4) = 'VOL1' THEN
      IF SUBSTR(BUF, 5, 6) = VOL1 THEN
    DO
      IF FILE_NO > 1 THEN
      DO
        TAPE FSF FSF_NO
        SET CMSTYPE HT
        ERASE @TAPE@ @LABELS@ A
        SET CMSTYPE RT
        MOVEFILE IN OUT
      END
      EXECIO 1 DISKR @TAPE@ @LABELS@ A '(VAR BUF'
      IF SUBSTR(BUF, 1, 4) = 'HDR1' THEN
        IF SUBSTR(BUF, 5, 17) = FN FT THEN
      DO
        EXECIO 1 DISKR @TAPE@ @LABELS@ A '(VAR BUF'
        IF SUBSTR(BUF, 1, 4) = 'HDR2' THEN
        DO
          FMT_O = SUBSTR(BUF, 5, 1)
          FMT = FMT_O'B'
          BLK = SUBSTR(BUF, 6, 5)
          RECS = SUBSTR(BUF, 11, 5)
          RECS_O = RECS
          ERASE @TAPE@ @LABELS@ A
          IF FSF_NO > Ø THEN
          DO
            SET CMSTYPE HT
            TAPE BSF 2
            TAPE FSR 1
            SET CMSTYPE RT
          END
```

```
          ELSE
          TAPE REW
          LEAVE
        END
        ELSE
        SAY'    Tape'HI'HDR2'LO'label'HI'not found'LO
      END
      ELSE
       SAY'    Tape'HI'HDR1'LO'label'HI'not matched'LO
       ELSE
       SAY'    Tape'HI'HDR1'LO'label'HI'not found'LO
     END
     ELSE
     SAY'    Tape'HI'VOL1'LO'label'HI'not matched'LO
     ELSE
     SAY'    Tape'HI'VOL1'LO'label'HI'not found'LO
     ERASE @TAPE@ @LABELS@ A
     EXIT
   END
END
ELSE
LEAVE = ''
ADD_OUT = '(' BLOCK RECS_O RECFM FMT_O
ADD_IN = '(' BLOCK BLK LRECL RECS RECFM FMT
FI OUTMOVE DISK FN FT FM ADD_OUT
FI INMOVE TAP1 SL FILE_NO ADD_IN LEAVE
TXT = 'WERE COPIED TO DISK'
ERR_TXT = 'WERE NOT COPIED TO DISK'
SET CMSTYPE HT
ERASE FN FT FM
SET CMSTYPE RT
END
MOVEFILE
IF RC = Ø THEN
SAY '>>>—>' FN FT FM||HI||TXT LO
ELSE
SAY '>>>—> error ->' FN FT FM||HI||ERR_TXT LO
SAY
SAY
```

Dobrin Goranov
Information Services Co (Bulgaria)

Poor man's CMS monitor

There were occasions in the past when I had no VM monitor available, but our operating staff needed to know whether or not a user was working (especially before system shutdown). To handle this situation, I wrote the following program.

On each occasion that the enter key is hit, the program reads the current values from CP IND USER. The new values are compared with the old values of CP IND USER, and the calculated differences are displayed.

This program uses IOS3270 (5785-HAX) and has been tested on VM/ESA 2.1.0.

The EXEC shows the following columns:

- ORIGINAL – values after the start of the program or reset.
- CURRENT – values retrieved after you pressed enter.
- CALCULATED – some calculated values. These are:
  - RES – the number of resident pages multiplied by 4K.
  - WS – the working set pages multiplied by 4K.
  - VTIME – the virtual time as a percentage of total time.
  - IO – I/Os per second.
- MAX – the maximum value since the start of the program or reset.
- MIN – the minimum value since the start of the program or reset.
- CHANGES – the difference between the original and the current values.

Note:

- Changed values are displayed in white and reverse. If a field is white and reverse, but the difference is 0, the value has changed and is now equal to the original value.

- You can enter an update interval after which the display is automatically refreshed.
- You can enter the name of a different CMS user for observation.
- If you press PF2, all values are reset.
- Press PF3 to exit.

```
Company Name              INDICATE for User: STEUER2                  INDF
———————————————————————— 14/11/97 16:25:04 -
STEUER2: MACH=XC  STOR=0003M VIRT=V XSTORE=NONE IPLSYS=CMS      DEVNUM=00015
             original      current computed     max         min         changes
Zeit:        16:24:04     16:25:04                                      0:01:00
RES:              155          155 620k         155         155               0
WS:               148          148 592k         148         148               0
READS:          91916        91948                                         32
WRITES:         66027        66041                                         14
LOCK:               0            0                0           0               0
RESVD:              0            0                0           0               0
NPREF:            359          359              359         359               0
PREF:               0            0                0           0               0
VTIME:         028:19       028:20 71.34%                                 0:01
TTIME:         039:41       039:43                                        0:02
IO:            159714       159821 1.78/sec                               107
RDR:           886303       886923                                        620
PRT:            65355        65400                                         45
PCH:            13987        13987                                          0
Automatic Update after 0    seconds. User: STEUER2
————————————————————————————————MAINT——

==> ____________________________________________________________________________
PF1=HLP 2=RES 3=END 4=    5=    6=    7=    8=    9=    10=     11=     12=
```

Figure 1: Display of INDF

A display of INDF is shown in Figure 1.

INDF EXEC

```
/* INDF: CP IND with fullscreen display */
         parse upper arg user .;
```

```
          /* Set variables to default values */
          ut = Ø;        /* Seconds between automatic redisplay */
          /* If no user supplied, display own userid */
          if user = ''
            then user = userid();
user:
          /* get current values */
          first = 1;
          call ind;
          if rc ¬= Ø
            then signal exit;
reset:
          /* Set attributes for all IOS327Ø fields to default */
          drop a4f.;
          /* Set variables to start values */
          parse value  res  ws  reads  writes  npref  pref  vtime,
             ttime  sio  rdr  prt  pch  lock  resvd  time(),
                       res  ws  lock  resvd  npref  pref,
                       res  ws  lock  resvd  npref  pref,
             with,
                      ores ows oreads owrites onpref opref ovtime,
            ottime osio ordr oprt opch olock oresvd otime,
                      mres mws mlock mresvd mnpref mpref,
                      ires iws ilock iresvd inpref ipref;
          /* Drop calculated values */
          drop cres cws cvtime csio;
          /* Set variables to Ø */
          parse value,
                      Ø    Ø   Ø      Ø       Ø      Ø     Ø,
            Ø      Ø    Ø    Ø    Ø    Ø     Ø      Ø with,
                      dres dws dreads dwrites dnpref dpref dvtime,
            dttime dsio drdr dprt dpch dlock dresvd dtime;
          pname = 'INDF';
          title = center('INDICATE for User:' user,4Ø);
loop:
          /* format variables                        */
          /*  xxx = current variable (new value)    */
          /* oxxx = original variable (old value)   */
          /* dxxx = difference between xxx and oxxx */
          /* mxxx = maximum value for xxxx          */
          /* ixxx = minimum value for xxxx          */
          /* cxxx = calculated value based on xxxx  */
           res    = format( res,7);
          ores    = format(ores,7);
          dres    = format(dres,7);
          mres    = format(mres,7);
          ires    = format(ires,7);
           ws     = format( ws,7);
```

```
ows     = format(ows,7);
dws     = format(dws,7);
mws     = format(mws,7);
iws     = format(iws,7);
 reads  = format( reads,7);
oreads  = format(oreads,7);
dreads  = format(dreads,7);
 writes = format( writes,7);
owrites = format(owrites,7);
dwrites = format(dwrites,7);
 npref  = format( npref,7);
onpref  = format(onpref,7);
dnpref  = format(dnpref,7);
mnpref  = format(mnpref,7);
inpref  = format(inpref,7);
 pref   = format( pref,7);
opref   = format(opref,7);
dpref   = format(dpref,7);
mpref   = format(mpref,7);
ipref   = format(ipref,7);
 vtime  = right(strip(vtime),7);
ovtime  = right(ovtime,7);
dvtime  = right(dvtime,7);
 ttime  = right(strip(ttime),7);
ottime  = right(ottime,7);
dttime  = right(dttime,7);
 sio    = format( sio,7);
osio    = format(osio,7);
dsio    = format(dsio,7);
 rdr    = format( rdr,7);
ordr    = format(ordr,7);
drdr    = format(drdr,7);
 prt    = format( prt,7);
oprt    = format(oprt,7);
dprt    = format(dprt,7);
 pch    = format( pch,7);
opch    = format(opch,7);
dpch    = format(dpch,7);
 lock   = format( lock,7);
olock   = format(olock,7);
dlock   = format(dlock,7);
mlock   = format(mlock,7);
ilock   = format(ilock,7);
 resvd  = format( resvd,7);
oresvd  = format(oresvd,7);
dresvd  = format(dresvd,7);
mresvd  = format(mresvd,7);
iresvd  = format(iresvd,7);
```

```
          dtime   = right(dtime,8);
          if ut ¬= Ø
            then cursor = 'TIME' ut;    /* Update after n seconds   */
            else cursor = '';
          nu = user;
          /* display screen */
disp:     call ios pname 'EXITØ';
          if IOSK = 'PFØ2'              /* reset values to Ø        */
            then signal reset;
          ut = strip(ut);
          if datatype(ut) ¬= 'NUM'
            then ut = Ø;
          /* Display values for a new user */
          if nu ¬= '' & nu ¬= user
            then
              do;
                upper nu;
                user = strip(nu);
                signal user;
              end;
          /* get current values */
          call ind;
          parse value  Ø    Ø   Ø      Ø       Ø      Ø      Ø    ,
            Ø     Ø    Ø    Ø    Ø    Ø    Ø   Ø with,
                     dres dws dreads dwrites dnpref dpref dvtime,
            dttime dsio drdr dprt dpch dlock dresvd dtime;

          /* Process RES */
          if res ¬= ores
            then
              do;
                A4F.$dres = "%" || x2c('ØØ');
                dres      = res - ores;
              end;
          mres = max(mres,res);
          ires = min(ires,res);
          cres = res*4'k';

          /* Process time */
          if time ¬= otime
            then
              do;
                A4F.$dtime = "%" || x2c('ØØ');
                parse var otime ohour ':' omin ':' osec;
                ozf   = ohour * 36ØØ + omin * 6Ø + osec;
                parse var  time  hour ':'  min ':'  sec;
                zf    = hour  * 36ØØ + min  * 6Ø +  sec;
                zf1   = (zf-ozf)%36ØØ;
```

```
      dsec  = zf-ozf;
      dtime = right(zf1,2) || ':' ||,
        right((zf-ozf-zf1*3600)%60,2,'0') || ':' ||,
        right((zf-ozf)//60,2,'0');
    end;

/* Process WS */
if ws ¬= ows
  then
    do;
      A4F.$dws = "%" || x2c('00');
      dws      = ws - ows;
    end;
mws = max(mws,ws);
iws = min(iws,ws);
cws = ws*4'k';

/* Process READS */
if reads ¬= oreads
  then
    do;
      A4F.$dreads = "%" || x2c('00');
      dreads      = reads - oreads;
    end;

/* Process WRITES */
if writes ¬= owrites
  then
    do;
      A4F.$dwrites = "%" || x2c('00');
      dwrites      = writes - owrites;
    end;

/* Process NPREF */
if npref ¬= onpref
  then
    do;
      A4F.$dnpref = "%" || x2c('00');
      dnpref      = npref - onpref;
    end;
mnpref = max(mnpref,npref);
inpref = min(inpref,npref);

/* Process PREF */
if pref ¬= opref
  then
    do;
```

```
      A4F.$dpref = "%" || x2c('00');
      dpref      = pref - opref;
    end;
mpref = max(mpref,pref);
ipref = min(ipref,pref);

/* Process VTIME */
parse var vtime min ':' sec;
zf  = min * 60 + sec;
zfv = zf;
if vtime ¬= ovtime
  then
    do;
      parse var ovtime omin ':' osec;
      ozf         = omin * 60 + osec;
      A4F.$dvtime = "%" || x2c('00');
      dvtime      = right((zf-ozf)%60,2) || ':' ||,
        right((zf-ozf)//60,2,'0');
    end;

/* Process TTIME */
parse var ttime min ':' sec;
zf = min * 60 + sec;
if ttime ¬= ottime
  then
    do;
      A4F.$dttime = "%" || x2c('00');
      parse var ottime omin ':' osec;
      ozf         = omin * 60 + osec;
      dttime      = right((zf-ozf)%60,2) || ':' ||,
        right((zf-ozf)//60,2,'0');
    end;
if zf > 0
  then cvtime = format(zfv*100/zf,,2)'%';

/* Process SIO */
if sio ¬= osio
  then
    do;
      A4F.$dsio = "%" || x2c('00');
      dsio      = sio - osio;
    end;
if dsec > 0
  then csio = format(dsio/dsec,,2)'/sec';

/* Process RDR */
if rdr ¬= ordr
```

```
            then
              do;
                A4F.$drdr = "%" || x2c('00');
                drdr      = rdr - ordr;
              end;

          /* Process PRT */
          if prt ¬= oprt
            then
              do;
                A4F.$dprt = "%" || x2c('00');
                dprt      = prt - oprt;
              end;

          /* Process PCH */
          if pch ¬= opch
            then
              do;
                A4F.$dpch = "%" || x2c('00');
                dpch      = pch - opch;
              end;

          /* Process LOCK */
          if lock ¬= olock
            then
              do;
                A4F.$dlock = "%" || x2c('00');
                dlock      = lock - olock;
              end;
          mlock = max(mlock,lock);
          ilock = min(ilock,lock);

          /* Process RESVD */
          if resvd ¬= oresvd
            then
              do;
                A4F.$dresvd = "%" || x2c('00');
                dresvd      = resvd - oresvd;
              end;
          mresvd = max(mresvd,resvd);
          iresvd = min(iresvd,resvd);

          signal loop;
exit0:    rc = 0;
exit:     exit rc;
/*********************************
* IND: Issue 'CP INDICATE USER' *
*********************************/
```

```
ind:
         'EXECIO * CP ( STEM IND. STRING IND USER' user;
         if rc ¬= 0
           then
             do;
               /* On initial call or if there is more than   */
               /* one message display them at the console,   */
               /* else set variable MESSAGE.                 */
               if first | ind.0 > 1
                 then
                   do i = 1 to ind.0;
                     say ind.i;
                   end;
                 else message = ind.1;
               signal exit;
             end;
         parse var ind.1 . ind.1;
         ind.1 = strip(ind.1);
         machine = user':' ind.1 ind.2;
         parse var ind.3 'RES=' res 'WS=' ws 'LOCK=' lock,
           'RESVD=' resvd;
         parse var ind.4 'NPREF=' npref 'PREF=' pref 'READS=' reads,
           'WRITES=' writes;
         parse var ind.5 'VTIME=' vtime 'TTIME=' ttime 'IO=' sio;
         parse var ind.6 'RDR=' rdr 'PRT=' prt 'PCH=' pch;
         /* Remove + for machines using the SIE assist */
         sio = strip(sio,'T','+');
         time = time();
         first = 0;
         return;

/************************
* IOS - Show the Panel *
************************/
ios:
         'NUCXLOAD IOS3270';
         parse upper arg i1 i2 '(' i4 . ;
         wer   = userid();
         date  = date('E');
         time  = time();
         pname = i1
         'IOS3270' i1 '( PA2 SUBSET' cursor clear ')'
         clear = 'NOCLEAR';
         if rc = 1 | rc = 2 | rc = 3 | rc = 5
           then
             do;
```

```
            say 'The panel' i1 'is not available.';
            say 'Please press the ENTER key';
            'CP SLEEP';
            exit;
          end;
        cursor = '0001';
        message = ' ';
        if IOSK  = 'PF03'
          then
            if i2 = '*'
              then return;
              else signal value strip(i2);
        if IOSK  = 'PF01'
          then
            do;
              'IOS3270' i1 'IOSHELP (' clear;
              signal 'IOS';
            end;
        if input = ''
          then return;
        input = strip(input,'T');
        upper input;
        interpret 'input';
        message = 'Returncode' rc 'from' input;
        input = '';
        signal 'IOS';
```

MAP INDF IOS3270

```
.TC
Company Name      %&TITLE.
&PNAME
_____________________________%&DATE-&TIME ____
.TC
.CJX SET CTL | TYPE=(SKIP PROT) DYNAMIC
.CJX SET CTL % H=REVERSE TYPE=(HIGH)
&machine
.ch
             original    current  computed  max      min      changes

Time:        &otime.      &time.                               |&dtime
RES:         &ores.       &res.    &cres.  &mres.    &ires.   |&dres
WS:          &ows.        &ws.     &cws.   &mws.     &iws.    |&dws
READS:       &oreads.     &reads.                              |&dreads
WRITES:      &owrites.    &writes.                             |&dwrites
LOCK:        &olock.      &lock.           &mlock.   &ilock.  |&dlock
RESVD:       &oresvd.     &resvd.          &mresvd.  &iresvd. |&dresvd
NPREF:       &onpref.     &npref.          &mnpref.  &inpref. |&dnpref
```

```
PREF:        &opref.      &pref.           &mpref.   &ipref.  |&dpref
VTIME:       &ovtime.     &vtime.  &cvtime.                    |&dvtime
TTIME:       &ottime.     &ttime.                              |&dttime
IO:          &osio.       &sio.    &csio.                      |&dsio
RDR:         &ordr.       &rdr.                                |&drdr
PRT:         &oprt.       &prt.                                |&dprt
PCH:         &opch.       &pch.                                |&dpch
Automatic update after#2&ut seconds. User:#8&nu
.YBHCF > > QUIT
PF1=HLP 2=RES 3=END 4=    5=    6=    7=    8=    9=    1Ø=     11=
12=
.CJ         _
.CL21
.&PZEILE
.C
————————————————————————————————&WER———
.CH3
&MESSAGE
.CN
==>¬75&INPUT
```

Thomas Rupp
Vorarlberger Illwerke AG (Austria)

VM Update, Issue 137, January 1998

Because of an error by our printers, certain copies of *VM Update* for January 1998 have been misnumbered as Issue 146. The correct issue number should be 137.

If you have a misnumbered copy, please contact Xephon at any of the addresses shown on page 2. We will replace the issue free of charge – our apologies for the mistake.

IBM's VM Web site

This review begins a series which will describe World Wide Web sites of interest to VMers (and readers are encouraged to suggest relevant sites by contacting Xephon at any of the addresses shown on page 2 or the author at gabe@acm.org). As we've all seen, VM is a powerful engine for creating Web sites, powering back-end functions such as calendar maintenance and database searches; it also serves Web content quickly and reliably. In addition, it's a robust client platform, allowing users to integrate Internet functions with other mainframe applications. Therefore, it's no surprise that there's an abundance of Web sites dedicated to enhancing VM operation, support, development, and use. Some of these are mainstream and easily found, while some are more obscure, perhaps dealing with VM or related topics, but not identified as VM-centric and so not appearing on VM-search results or linked from VM pages.

Even easily found sites sometimes have valuable nuggets beneath the surface – perhaps several clicks away from the main page. These reviews will highlight individual sites and drill down to expose resources. It's worth remembering a Web caveat: Web sites often change, so sites reviewed may not remain precisely as described. However, their fundamental mission and outlook should continue.

The first VM site to tour is IBM's VM operating system home page at http://www.vm.ibm.com/. IBM's Endicott VM laboratory takes justifiable pride in this Web page as a banner and news/information vehicle available around the world. It helps position VM in numerous niches – for example, by highlighting the availability of year 2000-ready OfficeVision/VM Release 4, describing the long-standing value of VM to VSE enterprises, and announcing VM support for the emerging technology of IBM's Network Station.

The page opens with the proud declaration "Served to you by VM/ESA Version 2.2.0", which is followed by a "Powered by System/390" logo. The specifications given for the site engine are:

- We're running VM/ESA Version 2.2.0 on an ES/9000 9221-200 having 128MB of main storage.

- Our Web server is Beyond Software's EnterpriseWeb/VM.
- E-Web is running in eight 32MB, XC-mode virtual machines.
- All of this site's data resides in the CMS Shared File System.

This indicates that it is an industrial-strength configuration, capable of serving large quantities of data to many viewers. Web page http://www.vm.ibm.com/siteinfo/gdlvmweb.gif illustrates how the VM system serving the Web pages fits into IBM's overall network. VM is perhaps unique in its ability to run multiple TCP/IP stacks that do not know about one another. This allows the safe connection of a single VM system to both the Internet and IBM's corporate intranet simultaneously. VM's TCP/IP even lets one specify virtual machines permitted to use a given stack. Brian Wade, who manages the server and download library, used this feature to define boundaries so that only the EnterpriseWeb/VM servers for www.vm.ibm.com can use the Internet stack. He notes that this feature is important for VM customers who might not have his luxury of dedicating a VM system to serving pages to the Internet.

The network connection is made through an IBM 3172-003 with a 66MHz Pentium CPU and 8MB of memory. The 3172 contains IBM TokenRing Busmaster Server adapters, which connect to the laboratory's Internet Token Ring loop. Eventually that loop leads to a T1 line out of the site. Even though the dedicated 9221 runs about 10% utilized on average, it is getting old and is to be replaced with a dedicated 9672. Brian has found System/390, VM/ESA, and EnterpriseWeb/VM to be a robust, nearly maintenance-free combination for serving Web pages, and he does no day-to-day rote work to keep the site running. During the recent holiday break he left the site unattended for 13 days; when he returned, he found that it had run the entire time without error. The only downtime in recent memory occurred when a squirrel jumped into a nearby power substation, killing itself and knocking out power to the lab and portions of surrounding Endicott. Most of Brian's work is longer-term, such as writing CGI scripts or helping users with HTML or site policy questions.

Web sites can be annoying by not being frequently updated, or by not

explicitly indicating when they were last changed. This site wins points – the first link listed being *What's new – Site change summary*, dated just four days before my visit. In January 1998 (only two-thirds through the month!) already included were:

- *European VM and VSE Technical Conference – brochure and enrollment information.*
- *Head to Dayton, OH for Midwest VM Regional Users Group (MVMRUG) on January 23.*
- *Read the value of VM to VSE enterprises, a new paper from IBM S/390.*
- *Sterling Software, Inc helps the US Defense Information Systems Agency enhance customer service with VM Web servers.*
- *S/390 Application of the month: Beyond Software, Inc. helps Ducks Unlimited Canada work the Web.*
- *General availability of Release 1 of S/390 service update facility.*
- *New VM service page which includes RSU and service FTP link.*
- *New VM batch facility page.*
- *VM/ESA spec sheet.*
- *New VM courses from IBM Education and Training:*
  - *BG93A – Creating the ideal Web server with VM/ESA*
  - *BG940 – Managing the year 2000 transformation with VM/ESA*

Each item links to detailed information. For example, the MVMRUG line jumps to http://miamiu.muohio.edu/~mvmrug/, the MVMRUG home page, which offers *About MVMRUG*, *Meeting schedules/meeting recaps*, *Upcoming meeting details*, *MVMRUG-L LISTSERV discussion list*, and *Other VM resources*. The last item listed illustrates the power and serendipitous nature of the Web, since it offers a jumping off point to still more VM resources, identified and aggregated by the MVMRUG people.

It's often worth browsing Web sites for other/related links, even if

you've already found the specific information you sought, since the site's owner thought the additional links would be valuable to visitors.

Other late breaking news, and an illustration of why a topical site like this is so valuable, is the description of the class *Managing the year 2000 transformation with VM/ESA* – which is so new that we're advised "This is hot off the press so it may not appear on the IBM training Web site yet, but the folks at IBM Teach do know about it and are ready to process your enrollment by phone today".

The page is divided into brief list items on the left – in categories *VM community, VM events, VM base product, VM related products, VM information and resources, Site information*, and (tantalizing but unreachable) *IBM internals only*. These groupings lend themselves to detailed and specific searching, being goal-oriented, and solving specific problems.

Long-time VMers know that the VM community has been one of VM's strengths, and these links illustrate that the community spirit remains strong. Clicking *Customer references* retrieves a page packed full of technical, organizational, and strategic success stories that can solve problems or bolster a case for continuing or expanding VM services. Resources as diverse as *Information Week, Forbes ASAP,* and *PC Week* are cited, along with one of my favourite author's thoughts, Jeff Savit's *A Strategic Outlook for VM*. Be sure to visit the VM developers' page – it links to the Web pages of many IBM Endicott staffers who play key roles in VM development and support. For example, Pam Christina, the editor responsible for VM Web site content, offers a glimpse into her work and interests, including a page of ramblings and sayings. My favourite is "VM: When you need more than what reality can offer", quoting Richard Ross of IBM Rochester NY. The page of Bill Bitner, a VM performance specialist, has a drawing of Bill which must be observed carefully for the performance wisdom it imparts, along with links to VM performance documents and downloadable tools.

The *VM events* category includes a VM events calendar, information on IBM's technical conferences, user group location/contact information, and information on past and planned teleconference calls. *VM base product* information includes the meat one would

expect: *VM V2 R3.0 announce, VM V2 R2.0 information, VM business value, VM & Year 2000, VM technical resources, performance report, and service*. The *VM-related products* area provides information on wares from IBM and non-IBM sources, valuable software available for download (from IBMers, customers, and developers), products which are open for beta test, and the page for *IBM global software solutions*, which I didn't actually drill into because it opens with a bloated and pointless graphic map of the world with which one can select the language in which to read. Even worse, the page captured my browser – clicking BACK had no effect, the same page kept loading. Fortunately, I'd already bookmarked the VM site – as readers should too.

Brian Wade notes that the download area is perhaps the most visited area of the site; it may be unique as a library to which both IBMers and non-IBMers can submit content. The server posts IBMers' submissions without manual intervention and alerts Brian when a submission arrives from outside IBM. This page serves as a repository for tools, documentation, and other nifty gadgets of specific interest to VMers, offering more than 300 packages available for download, representing a wide variety of content. The opening download page describes terms of use (very reasonable, no fee, etc) and methods of downloading and installing software; the content page lists packages available. Representative entries – illustrating items installed the day before this writing – include:

- PIPEDOCS 1998-01-21 VERY FAST/FUZZY PIPELINES SEARCH OF OV/VM DataBase.
- PIPEFIND 1998-01-21 VERY FAST/FUZZY PIPELINES SEARCH OF FILES.
- WINMAKER 1998-01-21 FULLSCREEN MENU/TOOLBAR/ SCROLL, VERSION 3.
- DATECNVT 1997-12-24 DATECNVT EXEC – Converts/ validates date input to another date format.
- SETDATE 1997-12-24 SETDATE EXEC – Set date of last update on CMS/SFS files.

VM info & resources links to *VM education, VM redbooks* (tutorial and cookbook publications slightly less formal that traditional product manuals), *Forums/discussions/lists* (communications glue that bonds the VM community), and *VM interested parties* (which in fact lists more categories of news and information). Finally, *Site information* provides interesting infrastructure insights, including site visitor statistics.

The body of the main page is formatted rather like newspaper headlines, with brief descriptions of linked articles. Don't miss the link next to the green VM bear at the page's bottom, providing abundant information on VM's role in intranets and the Internet.

Brian Wade collects EnterpriseWeb/VM-generated NCSA-standard log records in an SFS directory, one file for each day of operation. Each night the server reduces data in the previous day's log, translating IP addresses to hostnames and producing the hit summary available through the *Site information general info* and *Visit statistics* links. On the first day of each month, the server produces a similar report, using the previous month's data as input; the report is used to decide which parts of the site deserve attention. After producing the monthly report, the server uses VMARC to compress the previous month's files into a single file for archiving.

The philosophy on hit counting has always been to generate reports that reflect the number of visitors to the site rather than the HTTP transaction rate. The pipeline that counts site hits ignores log records that refer to page ornaments (such as masthead graphics) and other illustrations. Using this technique reveals that the machine serves about 2,500 complete pages daily via about 500,000 HTTP transactions per month.

Clearly, a Web site must be experienced to be fully understood and enjoyed – so I strongly suggest that readers pack their browsers and a snack, and (at least virtually) visit Endicott, where VM happens.

Gabe Goldberg
Computers and Publishing (USA)

Editor's note: The code from articles in VM Update is available free to subscribers from Xephon's Web site at www.xephon.com.

A calendar generator

The following program is a calendar covering all years from year 1 to year 9999, following the Gregorian conventions about leap years. It presents you with a screen as shown in Figure 1.

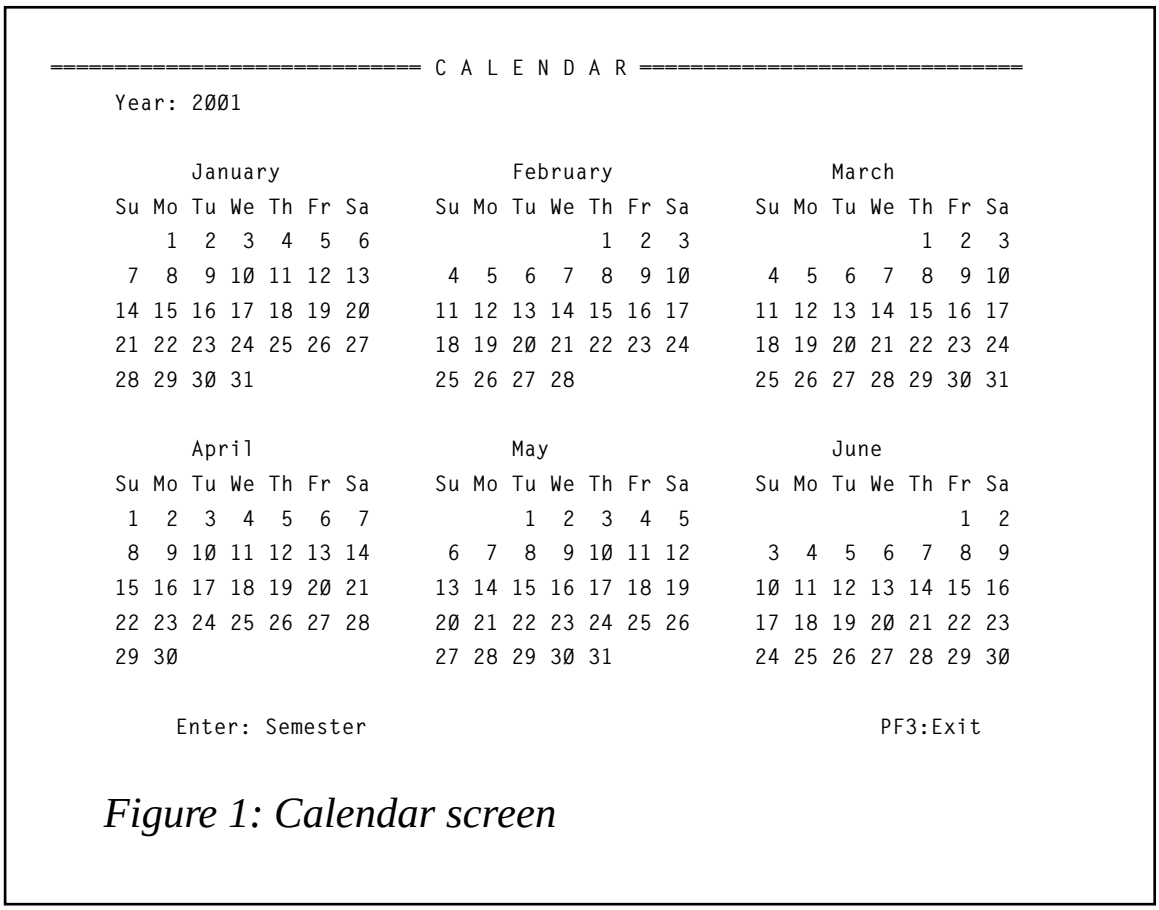

Figure 1: Calendar screen

You can choose the year you want, but if an invalid year is entered CALENDAR defaults to the present date. The enter key switches the semester (half year).

HOW TO RUN CALENDAR

CALENDAR works in an XEDIT environment, making use of its capabilities to create a formatted fullscreen. This means that to run CALENDAR, you need to launch CALENDAR XEDIT as a profile for a dummy edition. The easiest way to do this is to create a small

CALENDAR REXX file containing something like:

```
/**/
"xedit dummy dummy a (profile calendar"
```

This way, invoking CALENDAR in the CMS prompt, you get the XEDIT macro running.

USING XEDIT TO CREATE FULLSCREEN PANELS

There are several ways to create panels, or fullscreens, under CMS. My preferred method, however, is using the XEDIT command SET RESERVED, in conjunction with the READ command. This way, I can do everything with the same REXX program, without having to call external modules. The technique for this is quite simple, and I will explain it briefly, using CALENDAR XEDIT as a practical example.

If you look at my program, you will see that it has the following structure:

```
.....
DO ALPHA = Ø   (same as do forever, but with a control name)
   .....
   CALL DISPLAY_SCREEN
   READ NOCHANGE TAG
   DO QUEUED()
      PULL KEY LINE COLUMN STRING
         (select action according to KEY, LINE ...)
         WHEN KEY="PFK" & LINE=3 then leave alpha
         WHEN KEY="RES" & LINE=2 & COLUMN=12 then variable = STRING
   END
   ...
END
QUIT (leave xedit environment)
EXIT (exit this program).
...
(subroutines, including Display_screen).
```

You can see that it all turns around a perpetual loop that goes on displaying a screen until it detects a PF3, leaves the loop, and quits XEDIT. After the screen is displayed, when you hit any return key, the command READ is executed. This means that XEDIT reads whatever you have written on the screen. The NOCHANGE option means you

don’t need to update your file buffer in memory (don’t forget you are editing a file named ‘dummy dummy a’ or whatever). The TAG option means you want XEDIT to add an identifier to the STRING read to make it easier for you to know what is what.

The READ command places several lines on the stack, so the next logical thing to do is to PULL everything, with a DO QUEUED() loop. You can use PULL to separate the tags previously added (KEY, LINE, COLUMN) from the STRING, or data.

The KEY tag indicates which type of key was pressed (a PF key, ET key, PA key), or which area of the screen the STRING came from (CMDline, PRFix area, REServed line, FILe line). The LINE and COLUMN variables depend on the type of KEY. If KEY= ‘KEY’, LINE will be the PF key number. In my example, I am only concerned about 3 or 15, but I could test for other keys and assign them actions.

If I hit enter, instead of a PF key, variable key equals ‘ETK’. Because there is no need to change the program flow, this is ignored and allowed to fall under the ‘otherwise nop’ selection. The other relevant test for me is for KEY= ‘RES’, which means that an input originated on a REServed line. My screen fields are all located on reserved lines. LINE and COLUMN allow me to match screen positions with variables, and so retrieve the values typed on the fields by the user. CALENDAR only has one typeable (unprotected) field, that being the year you wish to see. This is located on row 2, column 12. Since it is the only one, only a RES tag will appear, which means there is no need to test for line and column positions.

Let’s look now at the contents of the ‘Display_screen’ paragraph. A panel in XEDIT is created through the ‘SET RESERVED line number text’ command. This means you want that line number to be excluded from the regular editing screen, and you want the line to appear with a fixed text (XEDIT’s SCALE line, STATUS line, and others, are reserved lines). What I do is reserve the entire screen with the text I choose. Furthermore, I disable features like the ‘Top of file’, message line, and command line, since I don’t need them.

The ‘text’ I choose to display on a reserved line need not consist solely of text. It may contain REXX variables and, more importantly, it may

contain control characters for the screen. The use of control characters (CTLCHARS) allows me to modify attributes such as colour, highlight, or the protect/unprotect status. Control characters are inserted as text and are made up of two bytes:

- The first byte is the escape character, and its only function is to indicate that the next byte is to be interpreted as a control character.
- The second byte is the control itself. Escape and CTLCHARS need to be defined with the 'SET CTLCHAR' command. In my program, I use '&' as an escape character, and I define other controls at will. The escape character must not occur on the screen itself.

If you look at my program, you will see that the text for the first reserved line, which I put in a variable called 'line1', begins with a '$*' sequence. This means that I want the screen to start as protected, highlighted, and with the colour red. Next, for the word 'CALENDAR', I insert the '&?' sequence to change the colour to pink. Note that the escape sequences are inside quotes, since they are text. Also, any escape sequence only occupies one byte on the physical screen.

On line 2, I have the text 'Year:', followed by an 'unprotect' sequence '&_', followed by a variable named YEAR, obviously outside quotes because I want REXX to replace the variable by its value. The remaining lines on the screen contain mostly escape sequences and variables.

For ease of printing, I have chosen to split each line in two and glue them together. On a real program it's easier to define them in full length.

A word of caution about variables – when you define a screen, you expect a variable or an input field to have a certain length, justified left or right, and padded with spaces. You must ensure that variables are of the correct length, otherwise your screen will appear distorted. For example, on line 4, I have three variables for month names. Since month names have different lengths, line 4 would end up with a different appearance depending upon the month. One way around this is to declare, before the definition of the lines, something like:

```
MTH1 = left(MTH1,9)
```

for all variables on the screen. Some people insert this type of thing within the line definition itself, but I prefer to do it outside, making the lines less confusing.

However, you will not find this trick of the left() or right() function in my program, except for input variable year. This is because I ensure that all variables are correctly created with the desired length.

Finally, how difficult is it to create the lines with text and variables in the desired positions?

First of all, I only have a few different lines in my screen, repeated several times. So I just need to set one correct for each type and repeat it. How is this done in practical terms?

Firstly, create a file that looks like the desired screen, with text as text and with variables as anything filling up their correct length. Also include the control characters, having decided previously what these are to be. Don't forget they take one byte of physical space.

Let's use some lines of my screen as an example:

```
*==================?C A L E N D A R*=================
    -Year:_XXXX+
+
      *AAAAAAAAA          *BBBBBBBBB          *CCCCCCCCC
 ?DD/EEEEEEEEEE%FF  ?GG/HHHHHHHHHH%HH  ?II/JJJJJJJJJJ%KK
```

Once you have defined the entire screen, issue CHANGE commands to insert the escape character, and to replace variable positions by their names:

```
c/-/&-/**
c/_/&_/**
c/*/&*/**
c/+/&+/**
c/XXXX/"year"/**
c/AAAAAAAAA/"mth1"/**
c/BBBBBBBBB/"mth2"/**
c/CCCCCCCCC/"mth3"/**
    .....
```

the lines above would become:

```
&*==================&?C A L E N D A R&*=================
    &-Year:&_"year"&+
&+
```

```
      &*"mth1"       &*"mth2"        &*"mth3"
 &?"a1"&/"b1"&%"c1"  &?"d1"&/"e1"&%"f1"  &?"g1"&/"h1"&%"i1"
```

Finally, insert quotes for everything except variables, give a name to each line, repeat them as necessary, and you are ready to issue the SET RESERVED commands. Note that I have used the CHANGE command to put quotes for the variables, thus reducing the work needed for this final touch:

```
line1="&*================&?C A L E N D A R&*================="
line2="    &-Year:&_"year"&+"
line3="&+"
line4="     &*"mth1"       &*"mth2"       &*"mth3
line5=" &?"a1"&/"b1"&%"c1"  &?"d1"&/"e1"&%"f1"  &?"g1"&/"h1"&%"i1
....
```

At the end of my program, I have taken the precaution of reserving lines from line 22 up to the end of the screen. This is especially useful for screens with more than 24 lines, and means there is no need to worry about different screen length.

For a complete description of the XEDIT commands, refer to the *XEDIT Command & Macro Reference* or the *User's Guide* manuals. *The REXX VM User's guide* also contains a description of this method of creating fullscreens, in a chapter dedicated to XEDIT, although using a quite different example.

CALENDAR XEDIT

```
/* REXX XEDIT ***********************************************************/
/*                                                                      */
/*                Calendar from year 1 to year 9999                     */
/*              Launch this macro with a dummy xedit                    */
/*                                                                      */
/************************************************************************/
call inicializa_ano
call inicializa_tabelas
extract "/lscreen"
do alpha = 0
   year = space(year,0)
   if ¬datatype(year,"W") then call inicializa_ano
   if year < 0 then call inicializa_ano
   year = right(year,4,"0")
   if year ¬= year_ant then sem = 1
```

```
   leap = leap_year(year)
   if sem = 1 then ddd = 0
   else ddd = 181 + leap
   days = year*365 + year%4 - year%100 + year%400 + ddd - leap
   avanco = days//7
   if avanco < 0 then avanco = avanco+7
   do mmm = 1 to 6
      posicao = (mmm-1)*6+1
      mes = mmm+sem-1
      call fill_month substr(tab,posicao,5)
   end
   year_ant = year
   call display_screen
   read nochange tag
   do queued()
      pull key line col string
      select
         when key = "RES" & line = 2 then year = string
         when key = "PFK" & (line=3|line=15) then leave alpha
         otherwise nop
      end
   end
   if sem=1 then sem=7
   else sem=1
end alpha
dropbuf
command quit
exit
/*********************************************************************/
/*                            Subroutines                            */
/*      fill_month creates the variable names for each month         */
/*                 and assigns the values to them.                   */
/*********************************************************************/
fill_month:
 arg xx yy zz .
 mes6 = mes//6
 if mes6 = 0 then mes6=6
 interpret "MTH"||mes6"='"month.mes"'"
 select
    when mes=4|mes=6|mes=9|mes=11 then lastday = 30
    when mes=2 then lastday = 28 + leap
    otherwise lastday = 31
 end
dia = -avanco
 if dia = -7 then dia = 0
 do semana = 1 to 6
    do x = 1 to 7
       d.x = (semana-1)*7 + dia + x
       if d.x < 1 then d.x = ""
```

```
      d.x = right(d.x,2)
    end
    if semana > 4 then do x = 1 to 7
      if d.x > lastday then d.x = "  "
    end
    interpret xx||semana"='"d.1"'"
    interpret yy||semana"='"d.2 d.3 d.4 d.5 d.6"'"
    interpret zz||semana"='"d.7"'"
 end
 avanco = (avanco+lastday)//7
return
/*********************************************************************/
/*     inicializa_ano sets present year and semester as default      */
/*********************************************************************/
inicializa_ano:
 year = left(date("S"),4)
 this_month = left(date("U"),2)
 if this_month > 6 then sem = 7
 else sem = 1
 year_ant = year
return
/*********************************************************************/
/*  leap_year returns 1 if year has 366 days, returns 0 otherwise.  */
/*********************************************************************/
leap_year:  procedure
arg ano .
bisexto = 0
if ano//4 = 0 then do
   if ano//100 = 0 then do
      if ano//400 = 0 then bisexto = 1
   end
   else do
      bisexto = 1
   end
end
return bisexto
/*********************************************************************/
inicializa_tabelas:
 tab = "A B C D E F G H I J K L M N P Q R S"
 month.1 = "January  "
 month.2 = "February "
 month.3 = "March    "
 month.4 = "April    "
 month.5 = "May      "
 month.6 = "June     "
 month.7 = "July     "
 month.8 = "August   "
 month.9 = "September"
```

```
 month.1Ø= "October  "
 month.11= "November "
 month.12= "December "
 weekdays = "Su Mo Tu We Th Fr Sa     "
return
/*********************************************************************/
display_screen:
 command set cmdline off
 command set msgline off
 command set tofeof  off
 command set ctlchar "&" escape
 command set ctlchar "_" noprot yellow high
 command set ctlchar "?" prot   pink   high
 command set ctlchar "%" prot   yellow
 command set ctlchar "/" prot   white
 command set ctlchar "*" prot   red    high
 command set ctlchar "<" prot   turq   high
 command set ctlchar "$" prot   blue
 command set ctlchar "+" prot   green
 command set ctlchar "-" prot   yellow high
 command cursor screen 2 12
 line1 = "&*============================&?C A L E N D A R"
 line1 = line1||"&*============================"
 line2 = "    &-Year:&_"YEAR"&+"
 line3 = "&+"
 line4 = "          &*"MTH1"               &*"MTH2
 line4 = line4||"               &*"MTH3
 line5 = "    &<"||copies(weekdays,3)
 line6 = "    &?"A1"&/"B1"&%"C1"&+   &?"D1"&/"E1
 line6 = line6||"&%"F1"&+   &?"G1"&/"H1"&%"I1"&+"
 line7 = "    &?"A2"&/"B2"&%"C2"&+   &?"D2"&/"E2
 line7 = line7||"&%"F2"&+   &?"G2"&/"H2"&%"I2"&+"
 line8 = "    &?"A3"&/"B3"&%"C3"&+   &?"D3"&/"E3
 line8 = line8||"&%"F3"&+   &?"G3"&/"H3"&%"I3"&+"
 line9 = "    &?"A4"&/"B4"&%"C4"&+   &?"D4"&/"E4
 line9 = line9||"&%"F4"&+   &?"G4"&/"H4"&%"I4"&+"
 line1Ø= "    &?"A5"&/"B5"&%"C5"&+   &?"D5"&/"E5
 line1Ø= line1Ø||"&%"F5"&+   &?"G5"&/"H5"&%"I5"&+"
 line11= "    &?"A6"&/"B6"&%"C6"&+   &?"D6"&/"E6
 line11= line11||"&%"F6"&+   &?"G6"&/"H6"&%"I6"&+"
 line12= "&+"
 line13= "          &*"MTH4"               &*"MTH5
 line13= line13||"               &*"MTH6
 line14= "    &<"||copies(weekdays,3)
 line15= "    &?"J1"&/"K1"&%"L1"&+   &?"M1"&/"N1
 line15= line15||"&%"P1"&+   &?"Q1"&/"R1"&%"S1"&+"
 line16= "    &?"J2"&/"K2"&%"L2"&+   &?"M2"&/"N2
 line16= line16||"&%"P2"&+   &?"Q2"&/"R2"&%"S2"&+"
```

```
 line17= "    &?"J3"&/"K3"&%"L3"&+   &?"M3"&/"N3
 line17= line17||"&%"P3"&+   &?"Q3"&/"R3"&%"S3"&+"
 line18= "    &?"J4"&/"K4"&%"L4"&+   &?"M4"&/"N4
 line18= line18||"&%"P4"&+   &?"Q4"&/"R4"&%"S4"&+"
 line19= "    &?"J5"&/"K5"&%"L5"&+   &?"M5"&/"N5
 line19= line19||"&%"P5"&+   &?"Q5"&/"R5"&%"S5"&+"
 line2Ø= "    &?"J6"&/"K6"&%"L6"&+   &?"M6"&/"N6
 line2Ø= line2Ø||"&%"P6"&+   &?"Q6"&/"R6"&%"S6"&+"
 line21= "    &+   Enter: Semestre               "
 line21= line21||"                        PF3:Exit"
 command set reser  1 noh line1
 command set reser  2 noh line2
 command set reser  3 noh line3
 command set reser  4 noh line4
 command set reser  5 noh line5
 command set reser  6 noh line6
 command set reser  7 noh line7
 command set reser  8 noh line8
 command set reser  9 noh line9
 command set reser 1Ø noh line1Ø
 command set reser 11 noh line11
 command set reser 12 noh line12
 command set reser 13 noh line13
 command set reser 14 noh line14
 command set reser 15 noh line15
 command set reser 16 noh line16
 command set reser 17 noh line17
 command set reser 18 noh line18
 command set reser 19 noh line19
 command set reser 2Ø noh line2Ø
 command set reser 21 noh line21
 do extra = 22 to lscreen.1
    command set reser extra noh
 end
return
```

Luis Paulo Figueiredo Sousa Ribeiro
Systems programmer
Edinfor (Portugal)

Subscribers who want copies of the code from this issue can visit our Web site – www.xephon.com – and ask for the article they require. The article will then be e-mailed to them. This service is free to subscribers.

Transferring files between ICCF and CMS

GENERAL DESCRIPTION

Many shops have changed their primary development environment from ICCF to VM/CMS, but they may still have source code in ICCF that is required from time to time. Furthermore, they may require some of the skeletons or sample jobs that are provided with an original VSE/ESA system within ICCF libraries. Occasionally they may need to transfer back some source files from CMS to an ICCF library.

To this end, I have developed some REXX and ICCF procedures for moving files from ICCF to CMS, and *vice versa*, initiated from the ICCF side or from CMS.

These perform the following functions:

- To CMS – fetching an ICCF member and punching it to the CMS reader.
- In CMS – reading the punched files into CMS files.
- From CMS – moving files to ICCF using the VSE punch queue and GETP in ICCF.
- From CMS – moving a file to ICCF via a VSE batch job that catalogues it in an ICCF library.
- From ICCF – moving one, several, or all files in an ICCF library to the CMS reader of a VM user.

MOVING AN ICCF MEMBER TO THE CMS READER

To move files from ICCF to CMS, using the VSE punch queue and GETP in ICCF, a batch job PUNMEM is submitted to the VSE reader that calls the ICCF utility program DTSUTIL with the PUNCH MEMBER command. Preceding this command is a REPRO command that punches a ':READ' card in front of the ICCF member. The latter is used to automatically name the file in CMS, when received from the CMS reader. The convention used is that the file gets the same filename as in ICCF, and I use 'LIBxx' as filetype, where 'xx' is the

ICCF library number of the original ICCF member. Only one member can be transferred at a time.

The procedure is called by:

```
PUNMEM <vse> member lib
```

where:

- 'vse' is the VSE machine where the batch job is to be submitted.
- 'member' is the ICCF member to be moved.
- 'lib' is the ICCF library where the member resides.

To submit the job to VSE, using variable parameters, some utility procedures were developed, which could also be used in other contexts:

- SUB EXEC submits a job to VSE.
- EXXX EXEC uses skeleton jobs with variable parts and produces a job stream that is finally submitted by calling SUB EXEC. Variables '*xxxxxx*' within the job can be replaced by real values.

All procedures use the construct of global variables set by a procedure SETGLOBL EXEC, published in *Back-up of CMS mini-disks and SFS directories, VM Update*, Issue 99, November 1994. SUB, EXXX, and INCLUDE were published in *Saving all relevant VM/VSE data – part 2, VSE Update*, Issue 26, June 1997.

INSTALLATION-SPECIFIC CONFIGURATION

The SUB and EXXX procedures automatically transfer files back to the CMS user that initiated the submit.

Permanent GLOBALV variables

Permanent GLOBALV variables are set by procedure SETGLOBL EXEC. This creates a file INITIAL GLOBALV which should be copied to the A disk of all users who want to work with PRFOLIE EXEC.

Some of the variables in SETGLOBL that are used by PRFOLIE and its child procedures are:

- $vse1 – first VSE system.
- $vse2 – second VSE system.
- $vse3 – third VSE system.
- $vsedef – default VSE system.

PUNMEM EXEC

```
/*********************************************************************/
/* Punch Member from ICCF into CMS reader                            */
/*********************************************************************/
/* Call:   PUNMEM <vse> member lib                                   */
/*               vse                 : VSE system                    */
/*                                   :   default: global variable    */
/*               member              : ICCF member                   */
/*               lib                 : ICCF library                  */
/*********************************************************************/
/* When arrived in CMS reader it can be transferred to a CMS file    */
/* with READCARD, RECEIVE, or READPUN (see below)                    */
/*********************************************************************/
trace off
'GLOBALV SELECT $$GLOB$$ GET $vse1    $vse2    $vse3    $vsedef'
parse upper arg vse member lib .
if vse = '' | vse = '?' then signal hilfe
if vse = substr($vse1,4,3) then vse = $vse1
if vse = substr($vse2,4,3) then vse = $vse2
if vse = substr($vse3,4,3) then vse = $vse3
if vse ¬= $vse1 & vse ¬= $vse2 & vse ¬= $vse3 then do
   vse = $vsedef
   parse upper arg member lib .
end

/* A file PUNMEM JOB must exist, that contains the job skeleton */

dummy = 'DUMMY'      /* no /INCLUDE */
'EXEC EXXX' vse member dummy dummy 'PUNMEM MEMBER' member 'LIB' lib

/*********************************************************************/
ende:
exit
/*********************************************************************/
/* Help                                                              */
/*********************************************************************/
```

```
hilfe:
'VMFCLEAR'
address cms 'type punmem  exec * 1 12'
```

PUNMEM JOB

```
// JOB PUNMEM
// DLBL DTSFILE,'ICCF.LIBRARY',,DA
// EXTENT SYSØ1Ø,SYSWK1
// ASSGN SYSØ1Ø,DISK,VOL=SYSWK1,SHR
// EXEC DTSUTIL
REPRO
:READ   *MEMBER* LIB*LIB*
PUN MEM(*LIB* *MEMBER*)
/*
/&
```

TRANSFERRING FILES FROM THE CMS READER TO A CMS FILE

This utility procedure simplifies the transfer of files from the CMS reader to a CMS file. It takes all reader files with a class of A and a type of PUN, and invokes READCARD to move them to CMS. The CMS filename and type is determined by the ':READ' card in front of the punched files. This means that they have to be created by PUNMEM or SUBVM ICCF procedures, mentioned below, because these put the ':READ' card in front of the files. Transferring is performed automatically (by invoking READCARD with the NOPROMPT parameter).

If the 'Prompt' option is specified, then the READCARD command is invoked with the FULLPROMPT parameter, which allows you to decide whether you really want this file moved, or wish to rename a file.

The procedure is called by:

```
READPUN <fm> <(Prompt>
```

where:

- 'fm' is the filemode, where the files should be placed in CMS.
- 'Prompt' means READ with prompting.

READPUN EXEC

```
/*********************************************************************/
/* Read RDR files with type PUN into CMS file                        */
/*********************************************************************/
/* Call:   READPUN <fm> <(Prompt>                                    */
/*                 fm   (default A)   : filemode, where the files    */
/*                                    : should be moved in CMS       */
/*                 Prompt             : READ with prompting          */
/*********************************************************************/
/* The RDR files must have been created by PUNMEM or come from the   */
/* ICCF procedures SUBVM/SUBVMALL/SUBVMFIL                           */
/* that means they must have a :READ card at the front               */
/*********************************************************************/
trace off
parse upper arg fm '(' prompt
if fm = '?' then signal hilfe
if fm = '' then fm = 'A'
if prompt = '' then prompt = 'NOPROMPT'
else do
   if abbrev(prompt,'P') > 0 then prompt = 'FULLPROMPT'
                             else signal hilfe
end
'EXECIO * CP (BUF 64000 STRING QUERY RDR * ALL'
pull .
do queued()
   pun = ''
   pull . num . pun . . . . . name .
   if pun = 'PUN' then do
     'CP SET IMSG OFF'
     'CP ORDER RDR' num
     'READCARD * *' fm '('prompt
     'CP SET IMSG ON'
   end
end

/*********************************************************************/
ende:
exit
/*********************************************************************/
/* Help                                                              */
/*********************************************************************/
hilfe:
'VMFCLEAR'
address cms 'type readpun exec * 1 12'
```

MOVING FILES TO ICCF

To move files to ICCF using the VSE punch queue and GETP, a batch job is created in file '$$PUN$$ $$PUN$$ A' that executes the VSE DITTO utility program with the CC (card to card) function. The file

to be moved is placed behind the CC as an input card deck. If there are '* $$' commands within the file (perhaps it's a job source) they are changed to '* $X' so that the VSE job doesn't interpret them as POWER commands. Only one file can be moved at a time.

Finally the punched file can be moved to an ICCF library by the GETP ICCF command:

```
GETP fn MEM=xxxxxxxx
```

where 'fn' is the name of the POWER punch queue entry, which is the same as the name of the CMS file to be transferred, and 'xxxxxxxx' is the name to be given to the ICCF member.

The procedure is called by:

```
PUN <vse> fm ft fm
```

where:

- 'vse' is the VSE machine to which the batch job is to be submitted.
- 'fn ft fm' is the CMS file to be moved.

Permanent GLOBALV variables

Some of the variables of SETGLOBL which are used by PRFOLIE and its child procedures are:

- $vse1 – first VSE system.
- $vse2 – second VSE system.
- $vse3 – third VSE system.
- $vsedef – default VSE system.
- $sysid1 – SYSID of the 1st VSE system.
- $sysid2 – SYSID of the 2nd VSE system.
- $sysid3 – SYSID of 3rd VSE system.

Hardcoded values

The variable 'cl' is used to specify the VSE POWER reader class where the job should be run. This can be changed to your requirements.

The file is punched to class Q in the VSE POWER punch queue. The job is built in a temporary file named $$PUN$$ $$PUN$$ on the A disk.

PUN EXEC

```
/*************************************************************/
/*  Punch a CMS file into the POWER reader queue             */
/*  from there it can be fetched into ICCF with the ICCF GETP command */
/*  (without disconnecting the DTSFILE)                      */
/*  * $$ -cards are automatically changed to * $X            */
/*************************************************************************/
/* Call:   PUN <vse> fn ft fm                                */
/*             vse                    : VSE                  */
/*                                    :   default: global variable    */
/*             fn ft fm               : File                 */
/*************************************************************************/
trace off
'GLOBALV SELECT $$GLOB$$ GET $vse1    $vse2    $vse3    $vsedef',
                           '$sysid1  $sysid2  $sysid3         '
parse upper arg vse fn ft fm .
if vse = '?' then signal fehler
if vse = substr($vse1,4,3) then vse = $vse1
if vse = substr($vse2,4,3) then vse = $vse2
if vse = substr($vse3,4,3) then vse = $vse3

address xedit 'EXTRACT /FN /FT /FM /ALT'
if rc = Ø & ¬(FTYPE.1 = 'FILELIST' & right(FMODE.1,1) = Ø) then do
   if vse = '' then vse = $vsedef
   if vse ¬= $vse1 & vse ¬= $vse2 & vse ¬= $vse3 then do
      vse = $vsedef
      parse upper arg fn ft fm .
   end
   if fn = '' then fn = FNAME.1
   if ft = '' & fn ¬= 'SEL' then ft = FTYPE.1
   if fm = '' & fn ¬= 'SEL' then fm = FMODE.1
   if alt.1 > Ø then address xedit 'SAVE'
   end
else do
   if vse = '' then signal fehler
   if vse ¬= $vse1 & vse ¬= $vse2 & vse ¬= $vse3 then do
      vse = $vsedef
      parse upper arg fn ft fm .
   end
end

'CP SET IMSG OFF'
address command 'ESTATE' fn ft fm
if rc ¬= Ø then do
   say fn ft fm 'does not exist'
   signal ende
```

```
end
cl = '4'
if vse = $vse1 then sys = $sysid1
if vse = $vse2 then sys = $sysid2
if vse = $vse3 then sys = $sysid3
queue 'MSGMODE OFF'
queue 'INPUT * $$ JOB JNM='fn',CLASS='cl',SYSID='sys
queue 'INPUT * $$ PUN CLASS=Q,DISP=K'
queue 'INPUT * $$ LST CLASS=Q,DISP=D'
queue 'INPUT // JOB PUNCH'
queue 'INPUT // UPSI 1'
queue 'INPUT // EXEC DITTO'
queue 'INPUT $$DITTO SET EOD=ı$ı$ı$ı$'
queue 'INPUT $$DITTO CC'
queue 'GET' fn ft fm
queue 'INPUT ı$ı$ı$ı$'
queue 'TOP'
queue 'N 4'
queue 'CH/* $$ /* $X /* *'
queue 'BOT'
queue 'INPUT $$DITTO EOJ'
queue 'INPUT /*'
queue 'INPUT /&'
queue 'INPUT * $$ EOJ'
queue 'FILE'
address command 'ERASE $$PUN$$ $$PUN$$ A'
'XEDIT $$PUN$$ $$PUN$$ A'
'CP SPOOL PUNCH' vse
'PUNCH UASS JOB A (NOH'
'CP SPOOL PUNCH SYSTEM'
say 'The file can be moved in ICCF via GETP fn MEM=xxxxxxxx'
/*********************************************************************/
/* End                                                               */
/*********************************************************************/
ende:
'CP SET IMSG ON'
exit
/*********************************************************************/
/* Errors/Help                                                       */
/*********************************************************************/
fehler:
'VMFCLEAR'
address cms 'type pun     exec * 1 11'
```

MOVING A FILE TO ICCF BY SUBMITTING A VSE BATCH JOB

When moving a file to ICCF by submitting a VSE batch job, the user is prompted for the ICCF library number to which the files are to be moved. The batch job is created in file '$$PUN$$ $$PUN$$ A' that executes the VSE DTSUTIL utility program with the ADD MEMBER

function. The file to be moved is placed behind this as an input card deck, terminated by an 'END OF MEMBER' card. If there are any '*$$' commands within the file, they are changed to '..$$', '/*' is changed to '../*', and '/&' is changed to '../&'. DTSUTIL reverses these changes when cataloguing members in ICCF. The advantages of this procedure are that you can transfer many files at a time and that they are catalogued automatically (without GETP). The disadvantage is that you must close the ICCF library with /DISC DTSFILE, so that ICCF users can't work in the meantime.

The procedure is called by:

```
PUNX <vse> fm ft fm
```

where:

- 'vse' is the VSE machine where the batch job is to be submitted.
- 'fn ft fm' is the CMS file to be moved.

Permanent GLOBALV variables

Permanent GLOBALV variables are the same as in the previous example.

Hardcoded values

As before, the variable 'cl' is used to specify the VSE POWER reader class where the job should be run and should be changed to meet your requirements.

The job is built in a temporary file named $$PUN$$ $$PUN$$ on the A disk.

The ICCF user 'AAAA' is used to catalogue the members. You must have this user defined, or change the procedure. The user must have the privilege to create files in the specified library.

Editor's note: this article will be continued next month.

Dr Reinhard Meyer (Germany)

VM news

For VM users of Adabas database and Natural application development products, Software AG has announced that it now provides full year 2000 support. As well as VM, other platforms on which the products are available include OS/390, VSE, and BS2000 mainframes. For Natural, the supported version for mainframes is 2.2.8 and for Adabas, Version 6.2 is required.

The company points out that client application programs and database structures must be adapted to meet year 2000 requirements when the new versions of the application programs are installed.

For further information contact:
Software AG of North America, 11190 Sunrise Valley Drive, Reston, VA 22091, USA.
Tel: (703) 860 5050.
Software AG (UK), Charter Court, 74-78 Victoria Street, St Albans, AL1 3XH, UK.
Tel: (01727 844 455.

IBM has announced Version 1.2 of its OV/VM Support for Lotus Notes clients, designed to migrate users to Notes clients without affecting existing OV/VM infrastructures. The new release supports Windows 95 and NT, host calendar resources when scheduling a meeting, time zone and daylight savings time adjustments for OV/VM meeting, and CMS notes in the in-basket.

Other features include an option to compact data transferred via native TCP/IP or IBM Personal Communications emulator, calendar support for Lotus Notes 4.6, and back-up of host in-basket to more than one notelog.

For further information contact your local IBM representative.

* * *