151

VM update

March 1999

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 75077-2150
USA
Telephone: 940 455 7050

Editorial panel
Articles published in *VM Update* are reviewed by our panel of experts. Members of the panel include John Illingworth (UK), Reinhard Meyer (Germany), Philippe Taymans (Belgium), Romney White (USA), Martin Wicks (UK), and Jim Vincent (USA).

Subscriptions and back-issues
A year's subscription to *VM Update*, comprising twelve monthly issues, costs £180.00 in the UK; $275.00 in the USA and Canada; £186.00 in Europe; £192.00 in Australasia and Japan; and £190.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1990 issue, are available separately to subscribers for £16.00 ($23.00) each including postage.

Editor
Robert Burgess

***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.

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*.

Purging VM spool files

This procedure purges RDR, PRT, and PUN files that are older than a specified date, for all CMS users in the system. You can specify whether the files to be purged are displayed before purging. This will enable you to decide whether each file really should be deleted.

The syntax is:

```
PURSPOOL date <(AUTO>
```

where:

- ‘date’ has the format ‘dd.mm’ (where ‘dd’ = day, ‘mm’ = month).
- ‘AUTO’ is the option to purge all files without displaying the files prior to purging.

If the day and month specified is before the current date, then the current year is assumed; if the day and month specified is the same as the current date, then the previous year is assumed.

Note: this procedure needs access to all spool files for all users and should be used by authorized personnel only!

PURSPOOL EXEC

```
/*******************************************************************/
/* Purging RDR/PRT/PUN-Files older than a specified date           */
/* (for all users)                                                 */
/* Unless specifying (AUTO, the files to be deleted are shown to   */
/* decide whether they should really be purged.                    */
/*******************************************************************/
/* PURSPOOL date  <(AUTO>                                          */
/*                 date              : Format: dd.mm               */
/*                                   : if dd.mm < current date ->  */
/*                                   :  current year               */
/*                                   : otherwise -> last year      */
/*                 AUTO              : purging without displaying  */
/*                                   :  files in advance           */
/*******************************************************************/
trace off
parse upper arg datum . '(' auto .
if datum = '' | datum = '?' then signal help
if auto = 'AUTO' then do
 call sayrt 'Do you really want to purge files automatically? (Y/N)'
 pull antwort
 if antwort ¬= 'Y' then exit
end
parse var datum tt '.' mm '.'
if datatype(tt,'W') = Ø | datatype(mm,'W') = Ø then signal help
if length(tt) = 1 then tt = 'Ø' || tt
if length(mm) = 1 then mm = 'Ø' || mm
if length(tt) ¬= 2 | length(mm) ¬= 2 then signal help
aktjjmmtt = date('S')
aktmmtt = substr(aktjjmmtt,5,4)
aktjj   = left(aktjjmmtt,4)
mmtt = mm || tt
if mmtt <= aktmmtt then jjmmtt = aktjj || mmtt
                   else jjmmtt = (aktjj - 1) || mmtt
'QUERY ALLOC SPOOL'
'MAKEBUF'
'SET CMSTYPE HT'
/**********************************************************************/
/* Selecting spool files with fitting date                            */
/**********************************************************************/

spool.1 = 'RDR'
spool.2 = 'PRT'
spool.3 = 'PUN'

do j = 1 to 3
   call sayrt '-------'
   call sayrt spool.j' files ...'
   call sayrt '-------'
   say ' '
   'EXECIO * CP (STEM LINE. BUF 64ØØØ STRING Q' spool.j 'ALL'
   do i = 1 to line.Ø
      if i = 1 then do; call sayrt line.i; iterate i; end
      parse var line.i user spoolno . . . . . smm '/' stt .
      if left(smm,5) = 'OPEN-' then iterate i
      smmtt = smm || stt
      if smmtt <= aktmmtt then sjjmmtt = aktjj || smmtt
                          else sjjmmtt = (aktjj - 1) || smmtt
      if sjjmmtt >= jjmmtt & sjjmmtt <= aktjjmmtt then iterate i
      call sayrt line.i
      if auto = 'AUTO' then antw = 'Y'
      else do until antw = 'Y' | antw = 'N' | antw = ''
         call sayrt 'purge? (Y=yes  N=no  blank=yes)'
         pull antw
      end
      if antw = '' | antw = 'Y' then do
         'EXECIO Ø CP (STR PURGE' user spool.j spoolno
         call sayrt '->' user spool.j spoolno 'purged'
      end
   end
end
```

```
/*********************************************************************/
/* End                                                               */
/*********************************************************************/
ende:
'SET CMSTYPE RT'
'QUERY ALLOC SPOOL'
exit
/*********************************************************************/
/* SAY with HT and RT                                                */
/*********************************************************************/
sayrt:
parse arg text
'SET CMSTYPE RT'
say text
'SET CMSTYPE HT'
return
/*********************************************************************/
/* Help                                                              */
/*********************************************************************/
help:
'VMFCLEAR'
address cms 'type purspool exec * 1 14'
```

Dr Reinhard Meyer (Germany) © Xephon 1999

A full screen console interface – part 8

Editor's note: this month we continue the code for the full screen console interface for Disconnected Service Machines (DSM). This article is an extensive piece of work which will be published over several issues of VM Update. *It was felt that readers could benefit from the entire article and from the individual sections. Any comments or recommendations would be welcomed and should be addressed either to Xephon or directly to the author at fernando_duarte@vnet.ibm.com.*

CSCSCN ASSEMBLE

```
         TITLE 'CSCSCN - CSC Scan data'
CSCSCN   START X'Ø17FFØ'
         PRINT NOGEN
         CSCHDR                         Scan data
*
```

```
* Scan, extract, translate, validate data and search command tables
*
*
         USING UIDSECT,R8              UID (user) Block
         USING CMDSECT,R2              CMD Commands Table
         SPACE
*
* Scan, extract, translate data and optionally search a table
*
*        Input R6 points to first byte of buffer or last word
*              RØ addresses the Command Table to search or zero
*              SCANLEN contains the length of last word
*              CSCBUFFE addresses the end of data to scan
*       Output R1 contains the real length of scanned word
*              R2 addresses CMD entry if table search was successful
*              R6 points to first byte of scanned word
*              R15 addresses of the processing routine or zero
*                 .For (E)xternal routines R15 address the first byte
*                  after the timestamp message
*                 .For (I)nternal routines R15 is shifted by 2Ø bytes
*                  to compensate for the (E)xternal save areas
*              SCANUPP contains word in uppercase (maximum 16 bytes)
*              A cc not zero is returned if no data was found
*   Separators Three separators are accepted: " ", "/" and ":"
*              Scanning the first word a numeric digit also accepted
*
*
         A     R6,SCANLEN              Skip previous word
         C     R6,CSCBUFFE             Anything left
         BNL   SCAN9ØØ                 No, done
SCAN1ØØ  CLI   Ø(R6),C' '              Skip all blanks between words
         BNE   SCAN2ØØ
         LA    R6,1(,R6)
         C     R6,CSCBUFFE             But check for end of data
         BL    SCAN1ØØ
         B     SCAN9ØØ                 Only spaces left, return
         SPACE
SCAN2ØØ  LR    R1,R6                   We found something
         SR    R2,R2
SCAN3ØØ  LA    R1,1(,R1)
         C     R1,CSCBUFFE             Look for end of data...
         BNL   SCAN4ØØ
         CLI   Ø(R1),C' '              ... or first blank...
         BE    SCAN4ØØ
         CLI   Ø(R1),C'/'              ... or "/" or...
         BE    SCAN4ØØ
         CLI   Ø(R1),C':'              ... or "/" or...
         BE    SCAN4ØØ
         C     R2,SCANLEN              Is it first word?
         BNE   SCAN3ØØ
         CLI   Ø(R1),C'Ø'             Yes, check also for numerics
         BL    SCAN3ØØ
         CLI   Ø(R1),C'9'
         BH    SCAN3ØØ
SCAN4ØØ  SR    R1,R6                  Length of word
         ST    R1,SCANLEN             Store it
         LR    R2,R1
         C     R2,SCANMAX             Check length
         BNH   SCAN5ØØ
         L     R2,SCANMAX             Too big, truncate
SCAN5ØØ  BCTR  R2,Ø                   Adjust for EXecute
         MVC   SCANUPP,BLANKS         Clear field
         EX    R2,SCANMVC             Move data
         OC    SCANUPP,BLANKS         Convert to uppercase
SEARCH   LTR   R15,RØ                 Any table to search?
         BZ    SCAN8ØØ                No, done
         LA    R2,CMDSIZEB            Length of table entry
         SR    R15,R2                 Prepare to loop
         LR    R2,R15
         SR    R15,R15                Routine not found yet
         SR    R3,R3                  Required by next IC
SCAN6ØØ  LA    R2,CMDSIZEB(,R2)       Advance pointer
         CLI   CMDSECT,X'FF'
         BE    SCAN8ØØ                End of table, invalid command
         IC    R3,CMDMIN              Minimum abbreviation
         CR    R3,R1
         BNL   SCAN6ØØ                Not enough input data, next...
         LR    R3,R1                  Copy length
         BCTR  R3,Ø                   Adjust for EXecute
         EX    R3,SCANCLC             Compare
         BNE   SCAN6ØØ
         L     RØ,CMDCLASS            Load command class
         N     RØ,UIDCLASS            Compare with user classes
         CL    RØ,CMDCLASS            Is it valid
         BNE   SCAN8ØØ                No, abandon search
         L     R15,CMDADDR            Command found, address routine
         CLI   CMDTYPE,C'E'           Is it an (E)xternal label?
         BNE   SCAN7ØØ                No, check for internal
         A     R15,Ø(,R15)            Skip timestamp
         B     SCAN8ØØ
         SPACE
SCAN7ØØ  CLI   CMDTYPE,C'I'           Is it an (I)nternal label?
         BNE   SCAN8ØØ                No...
         LA    RØ,2Ø                  Compensate (E)xternal save area
         SR    R15,RØ
SCAN8ØØ  CR    R14,R14                Generate cc of zero
         BACK                         Return
         SPACE
SCAN9ØØ  LTR   R14,R14                Generate non-zero cc
         BACK
         SPACE
SCANMVC  MVC   SCANUPP(*-*),Ø(R6)      Move data
SCANCLC  CLC   SCANUPP(*-*),CMDNAME    Compare name
         SPACE 3
*
* Search Commands Table
*
*
CSCSCNSC RELOC                         Search Command Table
         B     SEARCH                  Search and return to caller
         SPACE 3
*
* Verify if last scanned word is numeric
*
*
         BACK
         SPACE 3
*
* Verify if scanned data is numeric
*
*        Input R6 addresses the first byte of word to verify
*              SCANUPP contains the word to convert (from SCAN)
*              SCANLEN Contains the length of the word to verify
*       Output R2 contains the value in binary
*              A cc not zero is returned if data not numeric
*
*
CSCSCNVN RELOC                         Verify if numeric
         L     R1,SCANLEN              Load length
         C     R1,SCANMAXN             Check with maximum
         BH    NUM9ØØ                  Too big, done
         LR    R2,R6                   Copy address of first byte
         LR    R3,R1                   Copy length
         BCTR  R2,Ø                    Make next loop easier
NUM1ØØ   LA    R2,1(,R2)               Advance pointer
         CLI   Ø(R2),C'Ø'              Check for Ø-9 digits
         BL    NUM9ØØ
         CLI   Ø(R2),C'9'
         BH    NUM9ØØ                  No good, invalid character
         BCT   R3,NUM1ØØ               Check all bytes
         LR    R2,R1                   Copy length
         BCTR  R2,Ø                    Adjust for EXecute
         EX    R2,NUMPACK              Pack data
         CVB   R2,SCANDEC              Convert to binary
         CR    R14,R14                 Generate cc of zero
         BACK
         SPACE
NUM9ØØ   LTR   R14,R14                 Generate non-zero cc
         BACK
         SPACE
```

```
NUMPACK  PACK  SCANDEC,SCANUPP(*-*)
         SPACE 3
         CSCDATA
         CSCDS (UID,CMD)
         REGEQU
         END
```

CSCBLD ASSEMBLE

```
         TITLE 'CSCBLD - CSC Build user screen (3270 Data Stream)'
CSCBLD   START X'01A070'
         PRINT NOGEN
         CSCHDR                        Build user screen
*
* Build user screen (3270 Data Stream)
*
*        Input R8 addresses UIB block
*
*
         USING UIDSECT,R8              UID (user) Block
         USING CCHSECT,R7              CCH (cache) Block
         SPACE
         TM    UIDOPT1,UIDSEND         Is last Send still in progress?
         BO    BLD990                  Yes, let's wait
         TM    UIDOPT3,UIDPPROG        Are we printing / writing?
         BZ    BLD100                  No, process request
         L     R0,UIDPWREM             Print records to process
         LTR   R0,R0                   Anything left
         BNZ   BLD990                  Yes, wait
         NI    UIDOPT3,X'FF'-UIDPPROG  No, reset Print In Progress
         B     BLD990                  Next time we are back to work
         SPACE
BLD100   L     R3,UIDSCRN              Address user buffer
         TM    UIDOPT4,UIDBALM         ALARM requested
         BZ    TTL
         NI    UIDOPT4,X'FF'-UIDBALM   Yes, reset option
         MVC   4(L'COMMALM,R3),COMMALM Move ALARM command
         LA    R0,4+L'COMMALM          Length including prefix
         ST    R0,0(,R3)               Store prefix length
         AR    R3,R0                   Advance buffer pointer
         SPACE
TTL      TM    UIDOPT4,UIDBTTL         TITLE change
         BZ    HDR
         NI    UIDOPT4,X'FF'-UIDBTTL   Reset option
         MVC   4(L'COMMTTL,R3),COMMTTL Move TITLE command
         L     R1,SCRTTLL              Length of new title
         LTR   R1,R1                   Is it zero?
         BZ    TTL100                  Yes, skip data move
         BCTR  R1,0                    Adjust for EXecute
         L     R2,SCRTTL
         EX    R1,TTLMVC              Move data
TTL1ØØ   LA    RØ,4+L'COMMTTL
         A     RØ,SCRTTLL
         ST    RØ,Ø(,R3)              Create prefix
         AR    R3,RØ                  Advance buffer pointer
         SPACE
HDR      TM    UIDOPT4,UIDBHDR        HEADER overlay
         BZ    MCL
         NI    UIDOPT4,X'FF'-UIDBHDR  Reset option
         MVC   4(L'COMMHDR,R3),COMMHDR Move HEADER command
         L     R1,SCRHDRL             Length of overlay
         BCTR  R1,Ø                   Adjust for EXecute
         L     R2,SCRHDR
         EX    R1,HDRMVC              Move data
         LA    RØ,4+L'COMMHDR
         A     RØ,SCRHDRL
         ST    RØ,Ø(,R3)              Create prefix
         AR    R3,RØ                  Advance buffer pointer
         SPACE
MCL      TM    UIDOPT4,UIDBMCL        Move data to command line
         BZ    MSG
         NI    UIDOPT4,X'FF'-UIDBMCL
         MVC   4(L'COMMMCL,R3),COMMMCL
         L     R1,SCRMCLL             Length of data
         BCTR  R1,Ø                   Adjust for EXecute
         L     R2,SCRMCL
         EX    R1,MCLMVC              Move data
         LA    RØ,4+L'COMMMCL
         A     RØ,SCRMCLL
         ST    RØ,Ø(,R3)
         AR    R3,RØ
         SPACE
MSG      TM    UIDOPT4,UIDBMSG        Screen Message
         BZ    SCREEN
         NI    UIDOPT4,X'FF'-UIDBMSG
         MVC   4(L'COMMMSG,R3),COMMMSG
         L     R1,SCRMSGL             Length of message
         BCTR  R1,Ø                   Adjust for EXecute
         L     R2,SCRMSG
         EX    R1,MSGMVC              Move data
         LA    RØ,4+L'COMMMSG
         A     RØ,SCRMSGL
         ST    RØ,Ø(,R3)
         AR    R3,RØ
         SPACE
SCREEN   TM    UIDOPT4,UIDBSCR        User SCREEN
         BZ    RESET                  Not required, at least reset KB
         NI    UIDOPT4,X'FF'-UIDBSCR  Reset option
         LR    R4,R3                  Screen starting address
         MVC   4(L'COMMSCR,R3),COMMSCR Move SCREEN command
         LA    R3,4+L'COMMSCR(,R3)
         L     R7,UIDBUFF1             Address first detail line
         TM    UIDOPT3,UIDWRAP         Is WRAP switch On?
         BZ    SCR1ØØ                  No, check something else
         GO    CSCWRPGS                Get number of display columns
         ST    R5,BLDLSIZE             Save for later
         SR    R6,R6                   Detail line to build
         SR    RØ,RØ                   Required by next IC
SCRØ1Ø   IC    RØ,CCHLINE2             Last line number for this msg
         CR    RØ,R6                   Anything to display?
         BP    SCR1ØØ                  Yes, do it
         L     R7,CCHFWD               No, address next buffer record
         B     SCRØ1Ø
         SPACE
SCR1ØØ   TM    UIDOPT2,UIDEDS          EDS supported?
         BZ    SCR11Ø
         MVC   Ø(L'SAPREF,R3),SAPREF   Yes, set attributes for prefix
         LA    R3,L'SAPREF(,R3)
SCR11Ø   MVC   Ø(1,R3),CCHPREF         Move prefix
         MVI   1(R3),C' '              Separator
         TM    CCHOPTS,CCHHOLD         Message on Hold?
         BZ    SCR12Ø
         MVI   1(R3),C'>'              Yes, display indicator
SCR12Ø   LA    R3,2(,R3)               Adjust pointer
         TM    UIDOPT2,UIDEDS          EDS supported?
         BZ    SCR2ØØ
         MVC   Ø(L'SANORM,R3),SANORM   Yes, reset extended
         LA    R3,L'SANORM(,R3)
SCR2ØØ   LA    RØ,77(,R3)              Address end of line
         TM    UIDOPT2,UIDDATE         DATE to be displayed?
         BZ    SCR21Ø
         MVC   Ø(L'CCHDATE,R3),CCHDATE Yes, move it to the screen
         MVI   L'CCHDATE(R3),C' '      Separator
         LA    R3,L'CCHDATE+1(,R3)     Adjust pointer
SCR21Ø   TM    UIDOPT2,UIDTIME         TIME to be displayed?
         BZ    SCR22Ø
         MVC   Ø(L'CCHTIME,R3),CCHTIME
         MVI   L'CCHTIME(R3),C' '
         LA    R3,L'CCHTIME+1(,R3)
SCR22Ø   TM    UIDOPT2,UIDUSER         USER to be displayed?
         BZ    SCR3ØØ
         MVC   Ø(L'CCHUSER,R3),CCHUSER
         MVI   L'CCHUSER(R3),C' '
         LA    R3,L'CCHUSER+1(,R3)
SCR3ØØ   CLI   CCHATTR,X'ØØ'           Now, let's look at the message
         BE    SCR4ØØ                  No special attributes
         CLC   CCHUSER,BLANKS          Is it TOF, EOF, or Blank
         BH    SCR31Ø
         TM    UIDOPT2,UIDUSER         Yes, is user on screen?
         BZ     SCR31Ø
         LA     R1,9                     Yes, back-up user-id, separator
         SR     R3,R1
SCR31Ø   TM     UIDOPT2,UIDEDS           EDS supported?
         BO     SCR32Ø                   Yes, check colours and ext attr
         TM     CCHATTR,EDSHIGH          Highlight requested?
         BZ     SCR4ØØ
         BCTR   R3,Ø                     Yes, backup separator
         MVC    Ø(2,R3),SFHIGH           Move Highlight start field
         LA     R3,2(,R3)                Advance pointer
         LR     R1,RØ
         LA     RØ,1(,R1)                Adjust End-Of-Line by one byte
         B      SCR4ØØ
         SPACE
SCR32Ø   TM     CCHATTR,EDSEXT           Any extended attributes
         BZ     SCR34Ø                   No, check for colours
*        CLC    CCHUSER,BLANKS           Is it TOF, EOF, or Blank
*        BH     SCR33Ø
*        LA     R1,59                    Address column 21 (8Ø - 59)
*        LNR    R3,R1
*        AR     R3,RØ                    Buffer address for column 21
SCR33Ø   IC     R1,CCHATTR               Get attributes byte
         SLL    R1,27                    Remove all others bits
         SRL    R1,3Ø
         IC     R1,EXTATTR(R1)           Get 327Ø attribute byte
         MVC    Ø(3,R3),SAEXT            Move SA order
         STC    R1,2(,R3)                Store attribute byte
         LA     R1,3
         AR     R3,R1                    Advance pointer and End-Of-Line
         AR     RØ,R1
SCR34Ø   TM     CCHATTR,EDSCLR           Any colour requested?
         BZ     SCR4ØØ
         IC     R1,CCHATTR               Yes, get attributes byte
         SLL    R1,29                    Remove all other bits
         SRL    R1,29
         LA     R1,X'FØ'(,R1)            Get 327Ø attribute byte
         MVC    Ø(3,R3),SACLR            Move SA order
         STC    R1,2(,R3)                Store attribute order
         LA     R1,3
         AR     R3,R1                    Advance pointer and End-Of-Line
         AR     RØ,R1
SCR4ØØ   CLC    CCHUSER,BLANKS           Is this a TOF, EOF, or Blank?
         BH     SCR5ØØ                   No, must be a normal message
         LR     R1,RØ                    Address End-Of-Line
         SR     R1,R3                    Length of message
         EX     R1,SCRXC                 Clear all text field
         LA     R1,59                    Address column 21 (8Ø - 59)
         LNR    R3,R1
         AR     R3,RØ                    Buffer address for column 21
         IC     R1,CCHRLEN               Get length of message
         BCTR  R1,Ø                   Adjust for EXecute
         LA    R2,CCHDATA             Address message text
         EX    R1,SCRMVC              Move message text
*        LA    R3,1(R1,R3)            Address end of msg for ext attr
         LA    R6,1(,R6)              Ajust line for WRAP option
         B     SCR72Ø
         SPACE
SCR5ØØ   TM    UIDOPT3,UIDWRAP        Is WRAP switch On?
         BZ    SCR6ØØ                 No, process normal screen
SCR51Ø   LA    R6,1(,R6)              Next line to build
         SR    R2,R2                  Offset for first partial line
         ICM   R1,B'1ØØØ',CCHLINE1    Get first line on screen
         SRA   R1,24                  Convert to full word
SCR52Ø   CR    R1,R6                  Should we display it
         BNL   SCR61Ø                 Yes, do it
         LA    R1,1(,R1)              Check next partial line
         A     R2,BLDLSIZE            Adjust first display offset
         B     SCR52Ø                 Check it
         SPACE
SCR6ØØ   SR    R2,R2                  Required by next IC
         IC    R2,UIDCOL1             Offset of first col to display
SCR61Ø   SR    R5,R5                  Required by next IC
         IC    R5,CCHRLEN             Get length of message text
         SR    R5,R2                  Adjust message length
         BNP   SCR68Ø                 Nothing left, blank line
         LA    R2,CCHDATA(R2)         Address first column to display
         AR    R5,R3                  Check against space available
         CR    R5,RØ                  Is it too much?
         BNH   SCR62Ø
         LR    R5,RØ                  Yes truncate message
SCR62Ø   LR    R1,R5                  Copy to R1
         SR    R1,R3                  Length to move
         BCTR  R1,Ø                   Prepare to EXecute
         EX    R1,SCRMVC              Move message text
         NI    BLDOPTS,X'FF'-BLDRHIGH Reset Highlight option
SCR63Ø   EX    R1,SCRTRT              Check for non-displayable data
         BZ    SCR67Ø                 Nothing found
         CLI   Ø(R1),X'1D'            Is it a Start Field order?
         BE    SCR64Ø
         MVI   Ø(R1),X'ØØ'            No, replace with null
         B     SCR65Ø
         SPACE
SCR64Ø   LA    R1,1(,R1)              Allow Start Field orders
         OI    Ø(R1),X'2Ø'            But make sure field is protected
         LA    R2,1                   Remember X'1Dxx' uses one byte
         AR    RØ,R2                  Compensate for Start Field order
         OI    BLDOPTS,BLDRHIGH       Remember to reset Highlight
SCR65Ø   LA    R1,1(,R1)              Skip bad byte
         CR    R1,R5                  Anything left to check
         BNL   SCR66Ø                 No, all done for now
         LR    R3,R1                     New byte to start scanning
         LR    R1,R5                     Address end of data again
         SR    R1,R3                     Length left to check
         BCTR  R1,Ø                      Prepare to EXecute
         B     SCR63Ø                    Do it
         SPACE
SCR66Ø   BCTR  R1,Ø                      Go back to attributes byte
         MVI   Ø(R1),X'6Ø'               Restore normal attributes
         LA    R5,1(,R1)                 Do not truncate byte after SF
SCR67Ø   LR    R3,R5                     Restore regular pointer
SCR68Ø   CR    R3,RØ                     Any space at the right end?
         BNL   SCR7ØØ
         LR    R1,RØ                     Yes, address End-Of-Line
         SR    R1,R3                     Length of not used space
         EX    R1,SCRXC                  Blank it
SCR7ØØ   TM    UIDOPT3,UIDWRAP           Is WRAP switch On?
         BZ    SCR72Ø                    No, process normal screen
         ICM   R1,B'1ØØØ',CCHLINE2       Get last partial line to display
         SRA   R1,24                     Convert to full word
         CR    R1,R6                     Is it processed?
         BNP   SCR72Ø                    Yes, done with this message
         LR    R3,RØ                     No, address end of display line
         TM    CCHATTR,EDSEXT            Any extended attributes
         BZ    SCR71Ø
         MVC   Ø(3,R3),SAEXT             Yes, reset them
         LA    R3,L'SAEXT(,R3)
         LA    RØ,8Ø(,R3)                End address of next line
         XC    Ø(3Ø,R3),Ø(R3)            Clear prefix, date, time, user
         LR    R3,RØ                     End address of display line
         S     R3,BLDLSIZE               Address to move message text
         IC    R1,CCHATTR                Get attributes byte
         SLL   R1,27                     Remove all others bits
         SRL   R1,3Ø
         IC    R1,EXTATTR(R1)            Get 327Ø attribute byte
         MVC   Ø(3,R3),SAEXT             Move SA order
         STC   R1,2(,R3)                 Store attribute byte again
         LA    R1,L'SAEXT
         AR    R3,R1                     Adjust message address
         AR    RØ,R1                     Adjust end address of line
         B     SCR51Ø
         SPACE
SCR71Ø   LA    RØ,8Ø(,R3)                End address of next line
         XC    Ø(3Ø,R3),Ø(R3)            Clear prefix, date, time, user
         LR    R3,RØ                     End address of display line
         S     R3,BLDLSIZE               Address to move message text
         B     SCR51Ø                    Process next partial line
         SPACE
SCR72Ø   CLI   CCHATTR,X'ØØ'             Any attributes?
         BZ    SCR8ØØ
         TM    UIDOPT2,UIDEDS            Yes, is EDS supported?
         BO    SCR73Ø
         TM    CCHATTR,EDSHIGH         No, was message highlighted?
         BZ    SCR8ØØ
         LR    R3,RØ                   Yes, address End-Of-Line
         MVC   Ø(2,R3),SFNORM          Reset attribute
         LA    R3,2(,R3)               Advance pointer
         B     SCR9ØØ
         SPACE
SCR73Ø   TM    CCHATTR,EDSEXT          Extended attributes used?
         BZ    SCR75Ø                  No, check colours
*        CLC   CCHUSER,BLANKS          Is this a TOF, EOF or Blank?
*        BNH   SCR74Ø                  Yes, do not expand RevVideo
         LR    R3,RØ                   Address End-Of-Line
SCR74Ø   MVC   Ø(3,R3),SAEXT           Reset Extended attributes
         LA    R1,3
         AR    R3,R1                   Adjust pointer and End-Of-Line
         AR    RØ,R1
SCR75Ø   TM    CCHATTR,EDSCLR          Colours used?
         BZ    SCR8ØØ                  No, done
         LR    R3,RØ
         MVC   Ø(3,R3),SACLR           Reset colours
         LA    R1,3
         AR    R3,R1                   Adjust pointer and End-Of-Line
         AR    RØ,R1
SCR8ØØ   LR    R3,RØ                   Address End-Of-Line
         TM    BLDOPTS,BLDRHIGH        SF orders on this line?
         BZ    SCR81Ø                  No, just blank column 8Ø
         MVC   Ø(2,R3),SFNORM          Yes, reset attributes
         LA    R3,2(,R3)               Advance pointer
         B     SCR9ØØ
         SPACE
SCR81Ø   MVI   Ø(R3),C' '              Clear last byte (column 8Ø)
         LA    R3,1(,R3)               Adjust pointer
SCR9ØØ   L     R7,CCHFWD               Address next line
         LTR   R7,R7                   Anything left?
         BZ    SCR91Ø                  No, all done with screen DS
         TM    UIDOPT3,UIDWRAP         Is WRAP switch On?
         BZ    SCR1ØØ                  No, process new line
         ICM   R1,B'1ØØØ',CCHLINE2     Get last partial line
         SRA   R1,24                   Convert to full word
         CR    R1,R6                   Anything to display
         BH    SCR1ØØ                  Yes, do it
         B     SCR9ØØ                  No, do all lines
         SPACE
SCR91Ø   LR    R1,R3                   Next available byte
         SR    R1,R4                   Length of Data Stream
         ST    R1,Ø(,R4)               Store it
RESET    C     R3,UIDSCRN              Anything generated?
         BNE   RES1ØØ
         MVC   4(L'COMMRSK,R3),COMMRSK No, at least reset keyboard
         LA    RØ,4+L'COMMRSK          Length including prefix
         ST    RØ,Ø(,R3)               Store prefix length
         AR    R3,RØ                   Advance buffer pointer
RES1ØØ   S     R3,UIDSCRN              Length of generate Data Stream
         ST    R3,UIDSCRNL
BLD99Ø   BACK
         SPACE
         DS    ØD
TRTTABLE DC    64AL1(*-TRTTABLE)
         DC    192X'ØØ'
         SPACE
TTLMVC   MVC   4+L'COMMTTL(*-*,R3),Ø(R2)
HDRMVC   MVC   4+L'COMMHDR(*-*,R3),Ø(R2)
MCLMVC   MVC   4+L'COMMMCL(*-*,R3),Ø(R2)
MSGMVC   MVC   4+L'COMMMSG(*-*,R3),Ø(R2)
SCRMVC   MVC   Ø(*-*,R3),Ø(R2)
SCRTRT   TRT   Ø(*-*,R3),TRTTABLE
SCRXC    XC    Ø(*-*,R3),Ø(R3)
         SPACE 3
BLDLSIZE DS    F
         SPACE
SAPREF   DC    X'2842F6'               Attributes for PREFIX
SANORM   DC    X'2842ØØ'               Normal attributes
SFHIGH   DC    X'1DF8'                 Highlight Start Field
SFNORM   DC    X'1D6Ø'                 Normal Start Field
SAEXT    DC    X'2841ØØ'               SA for Extended attributes
SACLR    DC    X'2842ØØ'               SA for Colour
EXTATTR  DC    X'ØØF1F2F4'             327Ø bytes for Extended attr
EDSHIGH  EQU   B'ØØ1ØØØØØ'             Bits used by Highlight
EDSEXT   EQU   B'ØØØ11ØØØ'                  Extended attributes
EDSCLR   EQU   B'ØØØØØ111'                  Colours
BLDOPTS  DC    X'ØØ'                   Build options
BLDRHIGH EQU   X'8Ø'                   Reset Highlight on line
         SPACE 3
         CSCDATA
         CSCDS (CCH,UID)
         REGEQU
         END
```

CSCUSC ASSEMBLE

```
         TITLE 'CSCUSC - CSC Process User commands (IUCV)'
CSCUSC   START X'Ø18698'
         PRINT NOGEN
         CSCHDR                        Process User commands
*
* Process user requests
*
*
```

```
         USING UIDSECT,R8             UID (user) Block
         USING CCHSECT,R7             CCH (cache) Block
         SPACE
         TM    UIDOPT1,UIDCONN        Is user connected
         BZ    USER1ØØ                No, process data
         GO    CSCUSASD               Send data to destination node
         B     USERBYE                Wait for return
         SPACE
USER1ØØ  CLC   CSCBUFF(L'COMMINI),COMMINI
         BNE   USER2ØØ
         GO    CSCUIN                 Process Initial request
         B     USERSND                Refresh user screen
         SPACE
USER2ØØ  CLC   CSCBUFF(L'COMMCMD),COMMCMD
         BE    USERKEY
         MSG   Ø3ØØ                   Invalid data
         L     RØ,UIDPID              Get PATHID (first two bytes)
         GO    CSCSEV                 Terminate session
         B     USERBYE                Just return
         SPACE
USERKEY  SR    RØ,RØ                  Process interrupt key
         IC    RØ,CSCBUFF+L'COMMCMD   Get key code
         LA    R1,PFTABLE-8
USER3ØØ  LA    R1,8(,R1)              Address table entry
         CLI   Ø(R1),X'FF'
         BE    USER4ØØ                End of table, invalid PA/PF
         CLM   RØ,B'ØØØ1',Ø(R1)
         BNE   USER3ØØ
         L     RØ,Ø(,R1)              Found it, check user classes
         SLL   RØ,8                   Drop key code
         N     RØ,UIDCLASS            Compare with user classes
         CLM   RØ,B'111Ø',1(R1)       Is user authorized?
         BNE   USER4ØØ                No, invalid PA/PF key
         L     R15,4(,R1)             Yes, load routine address
         BASR  R14,R15                Execute routine
         B     USERCMD                Now process input command
         SPACE
USER4ØØ  MSG   Ø3Ø1,(USER,NOCMD)      Invalid PA/PF key
         B     USERSND                Update User screen
         SPACE
USERCMD  LA    R6,CSCBUFF+L'COMMCMD+6  First data byte
         SR    RØ,RØ
         ST    RØ,SCANLEN             Start new scan
         LA    RØ,USCTABLE
         GO    CSCSCN                 Scan command name
         BNZ   USERSND                Nothing, update user screen
         LTR   R15,R15                Is command valid?
         BNZ   USER6ØØ                Yes, process it
         CLI   SCANUPP,CSCLOCCH       Is it a default Locate ("/")
         BE    USER5ØØ                Yes, process it
         CLI   SCANUPP,CSCMATCH         Try also default Match ("\")
         BNE   USER7ØØ                  No good, invalid command
         LA    R2,MATCH                 Address Match entry in table
         B     USER51Ø
         SPACE
USER5ØØ  LA    R2,LOCATE                Address Locate entry in table
         USING CMDSECT,R2
USER51Ø  L     RØ,CMDCLASS              Load command class
         N     RØ,UIDCLASS              Compare with user classes
         CL    RØ,CMDCLASS              Is it valid?
         BNE   USER7ØØ                  No, process as invalid command
         L     R15,CMDADDR              Load routine address
         A     R15,Ø(,R15)              Skip timestamp
         SPACE
USER6ØØ  MVC   CSCCOMM,CMDNAME          Save command name
         GO    ,                        Execute processing routine
         B     USERSND
         DROP  R2
         SPACE
USER7ØØ  MSG   Ø3Ø2,USER                We got an invalid command
*        B     USERSND
         SPACE
USERSND  TM    UIDOPT1,UIDCONN          Is user connected?
         BO    USERBYE                  Yes, just return
         CLI   UIDOPT4,X'ØØ'            Any data to send to the user
         BE    USER9ØØ                  No, reset keyboard and return
         TM    UIDOPT4,UIDBHDR          Did Header change?
         BZ    USER9ØØ
         BAS   R14,ADDHDR               Yes, create new Header line
         TM    UIDOPT4,UIDBSCR          Was the screen rebuilt
         BO    USER9ØØ
         BAS   R14,REBUILD              No, do it now
USER9ØØ  TM    UIDOPT1,UIDRMTE          Is user remote?
         BO    USERBYE                  Yes, do not display data
         GO    CSCBLD                   Build data stream
         LINK  SEND                     Send data to the user
USERBYE  BACK
         SPACE 3
*
* Process TOP command   (PFØ4 or PF16)
*
*
TOPCMD   EQU   *                        Top (input command)
         ST    R14,CMDSV14
         SR    RØ,RØ                    No table to search
         GO    CSCSCN
         BNZ   TOPC1ØØ                  Nothing found, that's good news
         MSG   Ø312,USER                No parameters allowed
         B     TOPC9ØØ
         SPACE
```

```
TOPC1ØØ  BAS   R14,TOP                 Execute command
TOPC9ØØ  L     R14,CMDSV14
         BR    R14
         SPACE
TOP      EQU   *                       PFØ4 Top
         ST    R14,PFKSV14
         TM    UIDOPT2,UIDAUTO         Is user in Refresh mode
         BZ    TOP1ØØ                  No, do it
         NI    UIDOPT2,X'FF'-UIDAUTO   Reset AUTO Refresh option
         OI    UIDOPT4,UIDBHDR         Remember to refresh Header line
         B     TOP2ØØ                  Refresh the screen, no beeps
         SPACE
TOP1ØØ   GO    CSCWRPGT                Locate top line on screen
         SR    RØ,RØ
         C     RØ,CCHRECNO             Is record number valid?
         BNE   TOP2ØØ                  Yes, process command
         OI    UIDOPT4,UIDBALM         No, TOF or blank, sound alarm
         B     TOP9ØØ
         SPACE
TOP2ØØ   BAS   R14,TOPSCR              Build TOP screen
*        B     TOP9ØØ
         SPACE
TOP9ØØ   L     R14,PFKSV14
         BR    R14
         SPACE 3
CSCUSCTL RELOC                         TOPLINE (external call)
         BAS   R14,TOPLINE
         BACK
         SPACE
TOPSCR   EQU   *
         ST    R14,TOPSV14
         L     R7,UIDBUFF1             Address top line
         LINK  DELETE                  Delete it
         LINK  ADDTOFB                 Add TOF after last record
         ST    R7,NEWTOP               Save as new top line
         L     R7,UIDBUFF1             Delete the first line
         LINK  DELETE
         GO    CSCRDFFT                Read first line from file
         BNZ   TOPBLANK                Not found, add blank lines
         L     R1,UIDBUFF2             Add as last line
         LINK  ADD
         B     TOPL1ØØ                 Move line to the top
         SPACE
TOPLINE  EQU   *
         ST    R14,TOPSV14
         ST    R7,NEWTOP               Save as new top line
TOPL1ØØ  L     R7,UIDBUFF1             Address top line
         C     R7,NEWTOP               Is it the required new top line
         BE    TOPEND                  Yes, screen completed
         LINK  DELETE                  No, delete top line
         L     R7,UIDBUFF2              Address bottom line
         GO    CSCRDFNT                 Get next record
         BNZ   TOPBLANK                 Not found, add blank lines
         L     R1,UIDBUFF2              Add as last line
         LINK  ADD
         B     TOPL1ØØ
         SPACE
TOPBLANK EQU   *
         LINK  ADDEOFB                  Add EOF after last record
TOPB1ØØ  L     R7,UIDBUFF1              Address top line
         C     R7,NEWTOP                Is it the TOF message
         BE    TOPEND                   Yes, screen completed
         LINK  DELETE                   No, delete top line
         LINK  ADDBLKB                  Add blank line after last record
         B     TOPB1ØØ
         SPACE
TOPEND   EQU   *
         OI    UIDOPT4,UIDBSCR          Option to build user screen
         MVI   CCHLINE2,X'FF'           Make Top and Bottom lines valid
         L     R7,UIDBUFF2                 in case WRAP is turned Off
         MVI   CCHLINE2,X'FF'
         TM    UIDOPT3,UIDWRAP          Is WRAP switch On?
         BZ    TOPE9ØØ                  No, done
         GO    CSCWRPTP                 Yes, build partial lines
TOPE9ØØ  L     R14,TOPSV14
         BR    R14
         SPACE 3
*
* Process BOTTOM command   (PFØ5 or PF17)
*
*
BOTCMD   EQU   *                        Bottom (input command)
         ST    R14,CMDSV14
         SR    RØ,RØ                    No table to search
         GO    CSCSCN
         BNZ   BOTC1ØØ                  Nothing found, that's good news
         MSG   Ø312,USER                No parameters allowed
         B     BOTC9ØØ
         SPACE
BOTC1ØØ  BAS   R14,BOTTOM               Execute command
BOTC9ØØ  L     R14,CMDSV14
         BR    R14
         SPACE
```

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

Fernando Duarte
Analyst (Canada)

Managing chronological commands

Using PROP, there is a way to execute commands every minute, hourly, daily, every Monday, or every first or last day of the month.

To do this, copy both PROPREPT EXEC and PROPCOMM EXEC onto an accessed PROP disk, and set up a file called PROPCOMM ORIGINAL, as shown by PROPCOMM SAMPLE (keep this example file with the format information – the heading comments will be deleted in the working version). Finally add entries to the routing table.

Manage your command list file, PROPCOMM ORIGINAL, with the new commands:

- ‘msg operator send propcomm original a’.
- ‘msg operator file propcomm original 191 = = a rdpw’.

Editor’s note: the character ➤ denotes a formatting line break not present in the original code; it does not appear in the code downloadable from Xephon’s Web site.

COMMAND FILE

```
* Check communication with distributed node(s)
PROPCHK 1 1 DUMMYØØØ
 ...

* Programmable operator commands
**********************************************************************
*Comparison text      *Starting column
*                     *   *Ending column
*                     *   *   *Iucv message class          *Para-
*                     *   *   *  *Userid  *Nodeid  *Action  * meter
*------------*        *-* *-* ** *------*  *------*  *------*  *------*
$ DUMMYØØØ $           1  23                       PROPCOMM
 ...

* privileged user for repetition commands
**********************************************************************
*Comparison text      *Starting column
*                     *   *Ending column
*                     *   *   *Iucv message class          *Para-
*                     *   *   *  *Userid  *Nodeid  *Action  * meter
```

```
*——————————————*          *-* *-* ** *——————*   *——————*   *——————*   *——————*
/FILE /                    1   5      ATA              PROPREPT
/SEND /                    1   5      ATA              PROPREPT
 ...

* All remaining requests and commands
*************************************************************************
 ...
```

PROPCOMM EXEC

```
/*******************************************************************/
/* PROPCOMM -      EXEC for handling of repeatable commands          */
/*                                                                   */
/* Scan the file   PROPCOMM LISTmmdd A for actual commands           */
/*                                                                   */
/*******************************************************************/
hour_i = 3  /* time(hours) interval PROPCOMM LISTmmdd file is created */
RTable_call? = queued() = 2;
if RTable_call? then do;
   test? = Ø;
   arg req_id req_node lop_id lop_node msg_type prop_id
   ➤ prop_node net_id rtable_fn;
   pull;
   pull;
   end;
else do;
   test? = 1;
   arg . "(" Options;
   parse var Options "LIST" Test_dd "." Test_mm "." Test_yy
   ➤  Test_time Test_Week_Day Test_cur_day .;
   if Test_time = "" then do;
      parse source . . Source .;
      say "Format: " Source "(LIST dd. mm. yyyy
      ➤ hh:mm week_day days_in_year"
      exit 9;
      end;
   else do;
      Test_dd = right(strip(Test_dd),2,"Ø");
      Test_mm = right(strip(Test_mm),2,"Ø");
      Test_yy = "19"right(strip(Test_yy),2);
      Test_time = right(Test_time,5,"Ø");
      Test_Week_Day = bitor(substr(Test_Week_Day,1,2),"4ØØØ"x);
      end;
   end;
if test? then do;
   year = Test_yy;
   m_day = Test_mm || Test_dd;
   date = Test_mm"/"Test_dd;
   time = Test_time;
   cur_day = Test_cur_day;
   Week_Day = Test_Week_Day;
   end;
else do;
   year = word(date(),3)
   m_day = substr(date("S"),5,4)
   date = substr(date("O"),4,5)
   time = substr(time(),1,5)
   cur_day = date("Days")
   Week_Day = substr(date("Weekday"),1,2);
   end;
if year//4 = Ø then day_in_year =
➤ 'Ø  31  6Ø  91 121 152 182 213 244 274 3Ø5 335 366'
else day_in_year = 'Ø  31  59  9Ø 12Ø 151 181 212 243 273 3Ø4 334 365'
month = substr(m_day,1,2)
c_date = substr(date,1,3)
day_time = date'.'time
List_name = "LE" || right(year,2) || m_day "EXECUTED A";
'set cmstype ht'
'state propcomm list'm_day 'a'
state_rc = rc
'set cmstype rt'
already_done = Ø;
if state_rc = Ø then do
   do i = 1 by 1
      'execio 1 diskr propcomm list'm_day 'a' i '(fifo'
      rc_io = rc
      if rc_io = Ø then do
         pull . exec_time command
         if exec_time <= day_time then do;
            LINE.i = exec_time "("time()")" command;
            if ¬test? then command;
            if rc ¬= Ø then LINE.i = LINE.i "; rc=" rc;
            end;
         else do;
            if i > 1 then do;
               LINE.Ø = i-1;
               'execio' LINE.Ø 'diskw' List_name '(finis STEM LINE.';
               'execio Ø diskr propcomm list'm_day 'a (finis';
               'copyfile propcomm list'm_day 'a = = = (replace from' i;
               end;
            exit;
            end;
         end
      else do;
         'execio Ø diskr propcomm list'm_day 'a (finis';
         'erase propcomm list'm_day 'a'
         already_done = 1;
         leave i;
         end;
      end
   end
hour_c = substr(time,1,2)
if substr(time,4,2) > 5Ø then hour_c = hour_c + 1;
hour = min(24,hour_c % hour_i * hour_i + hour_i) - 1
last_minute = date'.'right(hour,2,"Ø")':59 '
'execio 1 diskw propcomm original a (finis string' right("*Ø*",8)
last_minute 'ØØ.ØØ:Ø1 ' last_minute '=* DUMMY COMMAND *'
parse var time c_h ":" c_m
min_c = (cur_day*24+c_h)*6Ø+c_m
executed = Ø;
do i = 1 by 1
   'execio 1 diskr propcomm original a (lifo'
   if rc ¬= Ø then leave
   pull cmd_nr first increment last subsequent sub_nr . "=" command
   if substr(cmd_nr,1,1) ¬="*" & substr(cmd_nr,1,1)¬="%" then iterate i
   parse var first f_month '/' f_day '.' f_h ':' f_m .
   fst_day = word(day_in_year,f_month) + f_day
   parse var increment i_day '.' i_h ':' i_m .
   if i_day = Ø & i_h = Ø then i_m = max(1,i_m)
   if datatype(i_day,"upper") then do
      To_Day = find("Mo Tu We Th Fr Sa Su",Week_Day)
      Lk_Day = find("MO TU WE TH FR SA SU",i_day)
      Last_Day = cur_day-To_Day+Lk_Day
      if Last_Day > cur_day then Last_Day = Last_Day-7
      fst_day = Last_Day-((Last_Day-fst_day)%7)*7
      i_day = 7; end
   if i_day = "3X" then i_day = max(3Ø,word(day_in_year,month));
   parse var last l_month '/' l_day '.' l_h ':' l_m .
   min_f = (fst_day*24+f_h)*6Ø+f_m
   min_i = (i_day*24+i_h)*6Ø+i_m
   min_delta = (max(Ø,min_c-min_f)%min_i)*min_i
   dt_day = min_delta%144Ø
   if fst_day+dt_day > word(day_in_year,month+1) then iterate
   c_day = fst_day+dt_day - word(day_in_year,month)
   min_delta = min_delta-dt_day*144Ø
   d_h = min_delta%6Ø
   d_m = min_delta-d_h*6Ø
   c_m = f_m+d_m
   if c_m >= 6Ø then do; c_m = c_m-6Ø; d_h = d_h+1; end
   c_h = f_h+d_h
   if c_h >= 24 then do; c_h = c_h-24; c_day = c_day+1; end
   if subsequent ¬= "" then do;
      parse var subsequent s_day '.' s_h ':' s_m .;
      if s_day = Ø & s_h = Ø then s_m = max(1,s_m)
      if sub_nr = "" then sub_nr = 1;
      end;
   if c_day < Ø then temp_day = Ø;
   else temp_day = min(c_day,99);
   current = c_date||right(temp_day,2,"Ø")".
```

```
    ➤ "right(c_h,2,"Ø")":"right(c_m,2,"Ø");
    if substr(first,1,5) <= date then do j = 1 by 1
    ➤ while current <= last & current <= last_minute
       if current < day_time then nop
       else if current = day_time then do;
          executed = executed+1;
          LINE.executed = day_time "("time()")" command;
          if ¬test? then command
          if rc ¬= Ø then LINE.i = LINE.i "; rc=" rc;
          end;
          else 'execio 1 diskw propcomm dumy'm_day
          ➤ 'a (finis string' right(cmd_nr,8) current command
       if subsequent ¬= "" then do;
          t_day = c_day; t_h = c_h; t_m = c_m;
             do k = 1 to sub_nr;
             t_m = t_m + s_m
             if t_m >= 6Ø then do; t_h = t_h + 1; t_m = t_m - 6Ø; end
             t_h = t_h + s_h
             if t_h >= 24 then do; t_day = t_day+1; t_h = t_h-24; end
             t_day = t_day + s_day
             if t_day < Ø then temp_day = Ø;
             else temp_day = min(t_day,99);
             current = c_date||right(temp_day,2,"Ø")".
             ➤  "right(t_h,2,"Ø")":"right(t_m,2,"Ø");
             if current > last | current > last_minute then leave k;
             if current < day_time then nop
             else if current = day_time then if already_done then nop
                   else do;
                      executed = executed+1;
                      LINE.executed = day_time "("time()")" command;
                      if ¬test? then command
                      if rc ¬= Ø then LINE.i = LINE.i "; rc=" rc;
                      end;
                else 'execio 1 diskw propcomm dumy'm_day
                ➤ 'a (finis string' right(cmd_nr,8) current command
             end
          end
       c_m = c_m + i_m
       if c_m >= 6Ø then do; c_h = c_h + 1; c_m = c_m - 6Ø; end
       c_h = c_h + i_h
       if c_h >= 24 then do; c_day = c_day+1; c_h = c_h-24; end
       c_day = c_day + i_day
       if c_day < Ø then temp_day = Ø;
       else temp_day = min(c_day,99);
       current = c_date||right(temp_day,2,"Ø")".
       ➤ "right(c_h,2,"Ø")":"right(c_m,2,"Ø");
       end
    end
'execio Ø diskr propcomm original a (finis'
LINE.Ø = executed;
if executed > Ø then 'execio' executed 'diskw'
```

```
➤ List_name '(finis STEM LINE.';
'set cmstype ht'
'sortv propcomm dumy'm_day 'a propcomm list'm_day 'a 10 20'
'erase propcomm dumy'm_day 'a'
'set cmstype rt'
if hour = hour_i-1 then do
   'execio * diskr propcomm original a (finis fifo'
   'erase propcomm original a'
   orig_nr = queued()
   new_lines? = 0
   do k = 1 to orig_nr
      pull CMD_Nr . . last . 1 LINE
      if last < date".00:00" & substr(CMD_Nr,1,1) = "%" then do;
         'execio 1 diskw propcomm obsolete a (finis var LINE';
         iterate k;
         end;
      if new_lines? & CMD_Nr = "*0*" then iterate k;
      queue LINE
      new_lines? = 1
      end
   'execio' queued() 'diskw propcomm original a (finis'
   end
exit
```

PROPCOMM SAMPLE

```
*  FORMAT:
* */¬ID* MM/DD.HH:MM  DD.HH:MM  MM/DD.HH:MM [DD.HH:MM #] =COMMAND
*
*   *ID* - active line
*   ¬ID* - inactive line
*
*    MM/DD.HH:MM - Date and time of first execution
*
*       DD.HH.MM - Time of repetition
*       DD = MO/TU/WE/TH/FR/SA/SO - every selected weekday
*       DD = 3X - the last day of the month
*
*        MM/DD.HH:MM - Date and time of last execution
*
*          DD.HH:MM # - (optional) subsequent repetition time and count
*
*                          =... - Command
*
*92*  01/01.08:15 00.01:00 12/31.24:00 =EXEC AUTOUIMP
*177* 01/01.08:00 01.00:00 12/31.24:00 00.10:00 1 =CP ACNT ALL CLOSE
¬178* 01/01.23:30 01.00:00 12/31.24:00 =EXEC UPDVMACC
*228* 01/01.09:30 SA.00:00 12/31.24:00 =CP START 0F2 CLASS E
*248* 01/01.09:15 01.00:00 12/31.24:00 00.03:00 5=EXEC REACCESS
```

```
*280* 01/01.00:00 00.00:05 12/31.24:00 =EXEC CHECKUID
*298* 01/01.13:00 MO.00:00 12/31.24:00 01.00:00 3=EXEC DAYTIME ATA LUNCH
*299* 01/01.12:30 FR.00:00 12/31.24:00  =EXEC DAYTIME ATA LUNCH
*310* 01/01.03:00 00.06:00 12/31.24:00  =EXEC CHECKPRI
*404* 01/01.05:00 01.00:00 12/31.24:00  =CP XAUTOLOG ORACLE2 #
*507* 01/01.08:00 MO.00:00 12/31.24:00  00.00:05 120 =EXEC QPAGING
*511* 01/01.08:00 FR.00:00 12/31.24:00  00.00:05 84 =EXEC QPAGING
*544* 01/01.00:00 3X.00:00 12/31.24:00  =XAUTOLOG ATA #EXEC PROPACCT
```

PROPREPT EXEC

```
/********************************************************************/
/* PROPREPT - Main EXEC for handling of repeatable commands         */
/*                                                                  */
/*   Functions:                                                     */
/* PROPREPT SEND  fn ft fm                                          */
/*          - sends a copy of the specified file to the requestor   */
/* PROPREPT FILE  fn1 ft1 vaddr fn2 ft2 fm2  <(linkpw)>             */
/*          - places or replaces the specified file2 on the PROP    */
/*            mini-disk by copying file1 from the requestors disk   */
/*                                                                  */
/********************************************************************/
arg req_id req_node lop_id lop_node msg_type prop_id prop_node net_id
➤ rtable_fn
if queued() = 2 then pull action current_file
if queued() = 1 then pull rtable_parameter
select
   when action = "SEND" then do
      if words(current_file) = 2 then current_file = current_file "A"
      if words(current_file) < 3 then do
         if current_file = "?" then do
           'tell' req_id 'at' req_node
           ➤ "Function  SEND  directs the PROP machine to send a copy"
           'tell' req_id 'at' req_node "of the specified file to you."
           end
         else 'tell' req_id 'at' req_node "Invalid format of command"
         'tell' req_id 'at' req_node "use:"
         'tell' req_id 'at' req_node "
          ➤  MSG" prop_id "SEND filename filetype <filemode>"
         exit; end
      'set cmstype ht'
      'state' current_file
      if rc = 0 then 'sendfile' current_file 'to' req_id 'at' req_node
      else 'tell' req_id 'at' req_node "File:" current_file "not found"
      'set cmstype rt'
      end
   when action = "FILE" then do
      New_Cmd = 0
      if words(current_file) < 6 then do
         if current_file = "?" then do
            'tell' req_id 'at' req_node "Function FILE invoke the
            ➤  installation of the specified"
            'tell' req_id 'at' req_node "new or updated file from your
            ➤ mini-disk into the PROP machine."
            end
         else 'tell' req_id 'at' req_node "Invalid format of command"
         'tell' req_id 'at' req_node "use:"
         'tell' req_id 'at' req_node "    MSG" prop_id
          ➤ "FILE  fn1 ft1 vaddr fn2 ft2 fm2  <(link-pw)>"
         exit; end
      parse var current_file f_n1 f_t1 v_addr f_n2 f_t2 f_m2.
      ➤ "(" link_pw ")"
      if substr(diag(ØØ),1,5) = 'VM/SP' then if link_pw =
      ➤ "" then link_pw = "ALL"
      'cp .det 2ff'
      'cp .link' req_id v_addr '2ff rr' link_pw
      if rc ¬= Ø then do; 'tell' req_id 'at' req_node
      ➤ "Can't link to your disk:" v_addr "rc = " rc; exit; end
      'desbuf'
      'set cmstype ht'
      'access 2ff j'
      file_n2 = f_n2
      if f_n2 = "=" then file_n2 = f_n1
      file_t2 = f_t2
      if f_t2 = "=" then file_t2 = f_t1
      if f_m2 = "D" then do
         stacked = queued();
         'query disk d (lifo'
         if rc = Ø & queued()-stacked = 2 then do;
            pull . D_Addr .; pull; end
         else do;
            do queued()-stacked; pull; end;
            'tell' req_id 'at' req_node
            ➤ "No disk with mode D accessed"; exit rc; end
         'access' D_Addr 'd'
         end
      'erase' file_n2 substr("OLD"file_t2,1,8) f_m2
      what_done = "replaced"
      'state' file_n2 file_t2 f_m2
      if rc ¬= Ø then what_done = "copied"
      else 'rename' file_n2 file_t2 f_m2 file_n2
      ➤  substr("OLD"file_t2,1,8) f_m2
      'copyfile' f_n1 f_t1 "J" f_n2 f_t2 f_m2
      if rc ¬= Ø then 'tell' req_id 'at' req_node
      ➤ "Copyfile failed, rc =" rc;
      else 'tell' req_id 'at' req_node "File:
      ➤ "file_n2 file_t2 f_m2 "  " what_done
      if f_m2 = "D" then 'access' D_Addr 'd/d'
      'release 2ff'
      'cp .det 2ff'
```

```
      'set cmstype rt'
      end
   otherwise do
      'tell' req_id 'at' req_node "No such command exist:" action
      'tell' req_id 'at' req_node "use:"
      'tell' req_id 'at' req_node "
       ➤ MSG" prop_id "SEND  filename filetype <filemode>"
      'tell' req_id 'at' req_node "or:"
      'tell' req_id 'at' req_node "
       ➤ MSG" prop_id "FILE  fn1 ft1 vaddr fn2 ft2 fm2  <(linkpw)>"
      end
   end
exit
```

Anton Altbauer (Germany) © Xephon 1999

Displaying ‘pseudo-graphics’ revisited

In the article *Displaying ‘pseudo-graphics’*, published in *VM Update*, Issue 148, December 1998, some periods (full stops) were omitted from the text. The following amendments should be noted:

On page 37 it should read:

- X. – the data for the x-axis. ‘X.’ has to be set to ' ' as the initial value.
- Y1. – the first data area.

On page 38 it should read:

- Y2. - the second data area.

The period (full stop) after X, Y1, and Y2 is important, and alters the meaning. For example:

- ‘X = ' '’ means to set the variable ‘X’ to ' '.
- ‘X. = ' '’ means to set the root of stem ‘X.’ to ' '.

© Xephon 1999

IBM's VM Download Library

Continuing our series of VM Web site reviews, we visit the VM Download Library. The site can be accessed at http://www.vm.ibm.com/download/. If you have comments on the Web sites reviewed in this series, or suggestions for relevant sites to review, please feel free to contact the author at gabe@acm.org or Xephon at any of the addresses shown on page 2.

In the old days – not too long by geological standards, but several computing generations ago – VM users were frustrated by knowing about VM-related tools IBM used internally that were unavailable, except under the most unusual, constrained, and mostly confidential circumstances. And at the same time, various libraries were maintained for software distributed at no cost (though often copyrighted) by VM consultants, sites, and software vendors. Fortunately, IBM has recognized that its customers are also its partners, and works to strengthen the partnership by providing software and facilities which strengthen the VM community by adding value to VM. This site, IBM's VM Download Library, begins:

"One of the experiments we're trying with this Web site is that of using this site as a download library. In general terms, we want to offer this site as a clearinghouse or repository for tools, documentation, and other nifty gadgets of interest specifically to VMers. We have set up the library so that both IBMers and non-IBMers can submit content and so that anyone can take content."

The good news is that the download site operates amazingly well with a minimum of formality; it's a tribute to several VM community members – IBMers and customers – who worked long and hard to initiate it. Most gratifying to contributors is that it's consistently among the most-often visited segments of the VM Web site. The unintrusive license agreement, mandatory to read before download, boils down to (but this summary is not legal advice!):

"You may download, use, execute, reproduce, display, and distribute this software. It's supplied as-is and may be withdrawn at any time. Respect any copyright notices, don't reverse engineer it, don't charge for it unless you own it. You may modify it but not distribute derivative

works based on other peoples' library contributions. Use it at your own risk, there are no warranties or support assurances. You are responsible for making selections from the Library, appropriateness and reliability of Library contents, and the manner and effectiveness of the downloading process."

Library packages consist of one or more downloadable files, in formats appropriate to the target platform, with instructions provided for handling each:

- VM content in a VMARC archive.
- PC content in a ZIP archive.
- UNIX content in a TAR archive.

IBM notes that downloads are available from three different types of IBM Web pages:

- No matter where else they might also reside, the packages page contains a summary of all downloads on the site.
- Product pages themselves sometimes contain downloads. Items on product pages relate, topically speaking, to the pages on which they reside. For example, you might find a performance-related download on a performance page.
- Developer pages sometimes contain items submitted by developers.

The linked packages page opens with pointers to other library pages (Neale Ferguson's OpenEdition tools page, CMS Pipelines Runtime Library Page, 1995 through 1998 VM Workshop tapes) followed by choices for viewing the library in three formats: all entries, monthly favourites, or all-time favourites. No matter how entries are viewed, download and documentation instructions are simple, as shown in Figure 1. This is clearer when viewing a sample entry, such as shown in Figure 2. This item comes in two versions, ZIP and TAR, which download when their links are selected. The product's description, which includes an e-mail link to the author, begins:

"This package allows you to send your VM user ID, password, and accounting information to the VM/ESA NFS server separately from a MOUNT request. The password can be specified as part of the mount

To grab the packages...	Click on...
VMARC archive	The v- link
ZIP archive	The z- link
TAR archive	The t- link
Description	The (+)

Figure 1: Download instructions

argument string, but that makes it visible to anyone who can issue the query form of the mount command (usually non-privileged), or ask the local mount service what has been mounted."

Since there's no telling which packages will strike one's fancy, it's worth perusing the entire list. But it's instructive initially to view the all-time favourites for indications of value and utility. Download counts (at the time of writing) varied from 2,356 for B2H (which converts Bookmaster, GML, Script/VS, and 'flat' files to HTML) to 357 for Getfile (which remotely initiates uploading PC files to VM users). B2H, the champion product, contributed by Gary L Richtmeyer of IBM Global Services, has a large top-quality documentation file rivalling what one sees with commercial products. The background section quickly establishes the software's context:

"If your organization is like mine, establishing a presence on the Internet by using a World Wide Web server and defining a Home Page is the 'thing to do'. You can either use someone else's Web server or install your own."

```
              z-52K
MOUNTPW    1998-03-17 Send userID/password/acct info to NFS server (+)
              t-120K
```

Figure 2: Sample entry

"We had plenty of mainframe-type documents to place on the server. Some were written using BookMaster, GML, and Script/VS markup, while others existed as normal 'flat' files. Even though the Web's HTML language is conceptually simple, the thought of manually converting the desired files from their source format to HTML was daunting."

"Although there are a few HTML converters around, none met all our major requirements. So we ended up writing our own, called it B2H, and are making it available for other organizations."

That's followed by a *What's new and different?* section giving the software's extensive maintenance and enhancement history – this is Version 4.2, after all. Under *What does B2H do?* is the summary sentence *"B2H reads files written in Bookmaster, Generalized Markup Language (GML), Script/VS, and even 'flat' files, and converts them into HTML format suitable for use in an Internet World Wide Web environment"*, followed by a lengthy highlights list too long to quote, the first few of which are:

- Converts most commonly-used tags, control words, macros and symbols of BookMaster, GML, and Script/VS (DCF).
- Supports conversion of 'flat' files.
- Output conforms to either HTML Release 2, 3, or 4, as specified at execution time.
- Can generate full and partial tables-of-contents; and if generated, options are available for specifying the table-of-content's format.
- Can generate an index; and if generated, options are available for specifying the index's format.
- Can generate figure and table lists.
- Can automatically generate hypertext links within the file itself.

The final demonstration of B2H's comprehensive nature is the list of platforms on which it operates: VM/ESA, AIX, Linux, TSO/E, OS/2, and Windows 95/98/NT.

Charlotte, the fifth-most popular download package, can enhance VM participation in Internet activities, and also simplify accessing other download resources. A very powerful and well-regarded Web browser, Charlotte will let you surf the Web from your 3270-style terminal, and allows fetching files directly from the IBM download library to VM without the two steps of downloading to a PC and uploading to VM. Charlotte installation is straightforward; it essentially works as installed/configured, possibly requiring a few simple commands as described in file WW2 README. (Presently available on the download page is Charlotte Version 2. Since Version 1 files began 'WWW', Version 2 files are denoted 'WW2'.) Charlotte highlights from the description file include:

- Provides full-screen text-only Web access from a VM/CMS terminal.
- Uses extended 3270 highlighting and colour where available.
- Based primarily on HTML 3.2 and HTTP 1.0 specifications.
- Where necessary, follows common practice rather than standards.
- Handles HTTP, FTP, GOPHER, NNTP (news), and local file protocols.
- Supports SOCKS or PROXY firewall servers.
- Supports binary download (either via link or from submit).
- Formats complex documents much faster than older REXX-only browsers.
- Wide tables are automatically adjusted to fit the screen if possible.
- Copes well with common errors such as missing or mismatched end tags.
- Interrogates the terminal character set to select the correct translate table.

Installation required renaming EAGALPRC MODULE to EAGRTPRC MODULE, and EAGALUME TXTAMENG to EAGUME TXTAMENG. After invoking Charlotte with the command WW2, switch to URL PF keys by pressing PF2, then press PF4 to allow a Web page to visit to be specified. You can also specify the initial URL to load as the operand of the WW2 command. After entering the address of the IBM download page, navigate with PF7/8 (for up/down scrolling) and the Tab key (for selecting links within a page). When the cursor is on the desired file link, press PF3 to return to the main PF key selection, then press PF4 to select File PF keys. Then press PF4 to invoke binary receive of the file, and you'll receive the specified file directly to your CMS A disk. You can create a simple EXEC to automate processing of VMARC files, as specified on the download page:

```
/**/
TRACE ERRORS
ARG FN
'PIPE <' FN 'VMARC A | FBLOCK 80 00 | >' FN 'VMARC A F 80'
'VMARC UNPK' FN 'VMARC A'
```

Another entry in the download hit-parade is LP3820, which converts AFP (LIST3820) printer files to files that can be printed on PC printers. Many mainframe-produced or related print files are in LIST3820 format, such as a number of files on Melinda Varian's home page (reviewed in *VM Update,* Issue 141, May 1998). This simple but powerful function needs only a minimal description:

- lp3820 Version 2.6.
- lp3820 converts AFP files (such as LIST3820) to personal printers including PostScript, HP LaserJet, DeskJet, and IBM LaserPrinter.
- For PostScript, HP LaserJet III and above, and Lexmark 4029 you need only this package. For DeskJet, HP LaserJet II, and IBM 4019 printers you need the additional font package lp3820f.

This is a port to VM/CMS of a PC program. As such, the command syntax and most of the documentation are for OS/2 and DOS. The documentation is included as an HTML file.

IBM Belgium employees contributed the CPQUERY command, a CMS GUI application written in REXX. The CMS GUI is described by http://www.vm.ibm.com/gui/ as:

"... a no-charge feature (which) provides an efficient means for host-resident applications to be displayed on a workstation using a graphical user interface and provides the capability to modernize the view to VM from the end-user perspective. The VM/ESA GUI Facility employs the concept of distributed presentation to allow you to produce VM host-resident GUI applications. The VM/ESA GUI Facility comprises an application programming interface and the CMS Desktop."

CPQUERY issues more than 20 CP commands to query the real VM system and presents the results on the desktop in an attractive manner. The program can display results for querying real and expanded storage use, DASD usage, and Saved Segments.

Another high-ranking item is LOOKALL, modestly entitled *Probably the best XEDIT-based search engine you will find*. It allows using XEDIT LOCATE syntax to search many files, presents a list of files containing the target, and then XEDITs each file in turn. It includes a global change macro, and recently added multiple target capability.

Returning to the main download library page, the other link for resources leads to VM developer home pages. These pages, revealing a bit about the people who bring us VM, are described:

"There are many folk in VM development, and they have all kinds of talents... for example, some are programmers and testers, some are information developers, others do business planning, forecasts, or marketing activities, and some are in management. Some of these people are interested in sharing their work with you via the Web, so we've made this area available on our site so that those who'd like 'a little corner of the Web' can easily have one."

Readers who attend user groups such as SHARE and WAVV, or the IBM VM/VSE Technical Conference, are likely to know several IBM VMers whose pages are shown. Bill Bitner, for example, is a well-known VM performance expert. His page, in addition to graphically illustrating the process by which he answers questions, and highlighting some of his contributions on the download page, begins:

"I joined IBM in 1985 and have worked in VM performance my entire career, and I love it. If you love performance (or hate it) you might want to check out the other VM performance info. My understanding of what VM performance means in the real world grew greatly when I became a development rep to the VM Performance and Capacity Planning Project of the VM Cluster in the SHARE organization."

That's followed by a section *More of my tidbits...* which links to a page described *If...then you're probably not on VM*. The page, entitled *How can you tell if you run on VM?*, includes gems like:

- *If you've never installed an operating system, had no idea what* *you were doing, and had it come up perfectly the first time, then you're probably not on VM.*

- *If after sending someone a note, you phone them the next day just to check if they got the note; then you're probably not on VM.*
- *If the keys with the most wear and dirt on your key board are Ctrl, Alt, and Delete, then you're probably not on VM.*

And Bill solicits submission of other sure-fire ways to tell you're using VM. A link invokes a Migration Sizing Tool to help when migrating between VM releases. Bill suggests, *"Test drive it and let me know what you think"*. The Sizing Tool page begins:

"Hello, do you want to get a quick sizing for a VM/ESA software migration? Fill in the following fields and you will get a quick sizing along with notes about things to watch out for. Much of the data is extrapolated. The data is for a CMS intensive environment, not necessarily OV/VM, DB, or guest environments. Please note the disclaimer which follows."

Other links provide various presentations Bill has done, and his Performance Almanac, which includes truths like these from Virg Meredith (IBM performance great):

- *"The right answer to the wrong question, is still the wrong answer."*
- *If someone tells me they ran a program once and it ran fine and then they ran it again without changing anything and it ran much slower, I ask them "If you didn't change anything, why did you run it again?"*

Another VMer familiar to user group and conference attendees is Brian Wade, who has done impressive research and development into service machine architecture and implementation. His Web page links to the Reusable Server Kernel (formerly known as the VM Server Superstructure) which is now generally available. The RSK page describes it:

"This package lets vendors and ambitious application programmers write multi-threaded server programs that are heavily exploitive of VM/ESA's best server-related technologies. These servers can be constructed without knowledge of data transport mechanisms (eg

TCP/IP), multi-threading APIs (CMS's ThreadCreate) or I/O performance boosters (eg VM Data Spaces) and without re-inventing API suites necessary in one server after another (eg, authorization primitives). APAR VM61878 for VM/ESA 2.2.0 and 2.3.0 delivers the RSK. The PTF numbers are UM29139 and UM29140 respectively. The PTF provides, with the caveat that experience with Assembler language or PL/X is required:

- *Line drivers for commonly-used data transport methods.*
- *A callable DASD I/O engine that hides volume boundaries, presenting a flat, block-oriented, persistent storage model.*
- *An authorization API relating users, objects, and actions.*
- *An enrollment API exploitive of VM Data Spaces.*
- *A file caching API exploitive of VM Data Spaces, including code page translation support.*
- *An anchor API.*
- *A storage management facility, including the ability to allocate and release storage in a VM Data Space.*
- *An API set to manage subordinate (worker) virtual machines.*
- *A run-time environment manager.*
- *Administrative command sets of various kinds.*
- *Language bindings for Assembler and PL/X programmers."*

More than three dozen VMer home pages are posted; visit those of people you know, or meet some new people on-line. If you have IBM VMer friends who don't have Web pages, encourage them to create them! The final area on the download page concerns submissions to the library – with separate submission agreements and procedures for IBM and non-IBM employees. Neither agreement/procedure is overly burdensome; IBM is clearly working for the benefit of the VM community as much as possible by sharing its resources and supporting the distribution process.

Gabe Goldberg
Computers and Publishing (USA) © Xephon 1999

Mouse-clickable XEDIT enhancements

Continuing the Mouse on the mainframe *series of articles on the manipulation of System/390 applications with a PC or workstation mouse, the author discusses writing mouse-clickable XEDIT enhancements.*

INTRODUCTION

Previous articles in this series have discussed the concept of mouse-clickable 3270 applications as well as specific programming techniques that can be used to create such applications. 'Pointer Enabled Tools' or PETs were seen as easier to use than traditional 3270 applications, especially for novice or casual VM/CMS users. Programs can be enabled for mouse clicks by exploiting standard programming tools readily available on all VM/CMS systems.

This article outlines one way in which XEDIT macros can be written to be manipulated with a workstation mouse. Mouse-clickable XEDIT tools can be written using REXX, XEDIT subcommands, virtual screens, and CMS windows. These techniques are standard on all VM/CMS systems, and documentation is readily available in Help files and in IBM manuals.

This article attempts to demonstrate how programming techniques can be combined to create new PETs. Topics to be covered include:

- Changing the look and feel of the XEDIT screen.
- Using PROFILE XEDIT to customize XEDIT.
- Using XEDIT reserved lines to add PF key help text to the screen.
- Assigning alternative functions to the PF keys.
- Extracting cursor location and other information into program variables.
- Invoking PF key functions with mouse clicks.
- Adding clickable pop-up command menus to the XEDIT screen.

- Mapping XEDIT profiles to filetypes.

XEDIT AND XEDIT PROFILES

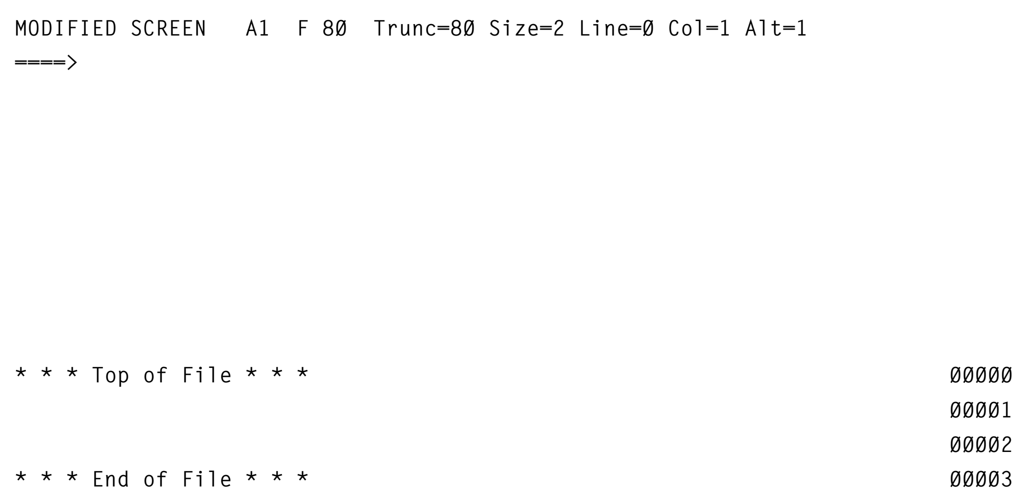
XEDIT is a powerful and versatile text editor, long available with VM/CMS. While development of XEDIT itself may be finished, any number of new functions can be written to enhance the editor, including functions which respond to appropriate ‘mouse clicks’. Let’s begin by reviewing the XEDIT screen. By default, XEDIT presents a screen similar to that shown in Figure 1.

```
DEFAULT  SCREEN   A1  F 80  Trunc=80 Size=0 Line=0 Col=1 Alt=0






===== * * * Top of File * * *
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
=====
=====
===== * * * End of File * * *




====>
                                                            X E D I T  1 File
```

Figure 1: The default XEDIT screen

Novice users of XEDIT often don’t realize that the look and feel of the XEDIT screen can be modified by issuing appropriate XEDIT subcommands. For example, an XEDIT screen might be modified to look like Figure 2 by entering the following XEDIT subcommands on the XEDIT command line:

```
SET CMD TOP
SET NUM ON
```



```
MODIFIED SCREEN   A1  F 80  Trunc=80 Size=2 Line=0 Col=1 Alt=1
====>









* * * Top of File * * *                                            00000
                                                                   00001
                                                                   00002
* * * End of File * * *                                            00003
```

Figure 2: Modified XEDIT screen

```
SET PREFIX ON RIGHT
SET SCALE OFF
```

These subcommands set the command line on line 2 of the 3270 display, turn on line numbering, reposition the prefix area to the right of the text area, and hide the scale line. XEDIT provides a number of controls which modify how the screen looks.

Most XEDIT users know that a special file called ‘PROFILE XEDIT’ can be used to initialize every XEDIT session in a similar way. The PROFILE XEDIT file is an XEDIT macro or program that can contain XEDIT subcommands. The PROFILE XEDIT macro is executed whenever XEDIT opens a file. To initialize every XEDIT session so that the screen looks like Figure 2, one could create the following PROFILE XEDIT macro.

```
/* Sample PROFILE XEDIT */
'SET CMD TOP'
'SET NUM ON'
'SET PREFIX ON RIGHT'
'SET SCALE OFF'
Exit
```

PF KEYS

Most XEDIT users also know that XEDIT assigns certain XEDIT subcommands to the PF keys. The default XEDIT screen, however, provides no assistance in learning or remembering those PF key assignments. One can issue the XEDIT subcommand QUERY PF which returns information similar to that in Figure 3, but repeatedly typing this subcommand is rather inefficient.

```
PF1      BEFORE  HELP MENU
PF2      BEFORE  SOS LINEADD
PF3      BEFORE  QUIT
PF4      BEFORE  TABKEY
PF5      BEFORE  SCHANGE 6
PF6      ONLY    ?
PF7      BEFORE  BACKWARD
PF8      BEFORE  FORWARD
PF9      ONLY    =
PF1Ø     BEFORE  RGTLEFT
PF11     BEFORE  SPLTJOIN
PF12     BEFORE  CURSOR HOME
PF13     BEFORE  HELP MENU
PF14     BEFORE  SOS LINEADD
PF15     BEFORE  QUIT
PF16     BEFORE  TABKEY
PF17     BEFORE  SCHANGE 18
PF18     ONLY    ?
PF19     BEFORE  BACKWARD
PF2Ø     BEFORE  FORWARD
PF21     ONLY    =
PF22     BEFORE  RGTLEFT
PF23     BEFORE  SPLTJOIN
PF24     BEFORE  CURSOR HOME
```

Figure 3: Default XEDIT PF assignments

It is helpful if some ‘on screen reminders’ or ‘help text’ about the PF key assignments is provided. This is usually accomplished by setting aside one or two XEDIT ‘reserved lines’ for the help text. Reserved lines are typically defined in a PROFILE XEDIT macro with the RESERVE XEDIT subcommand:

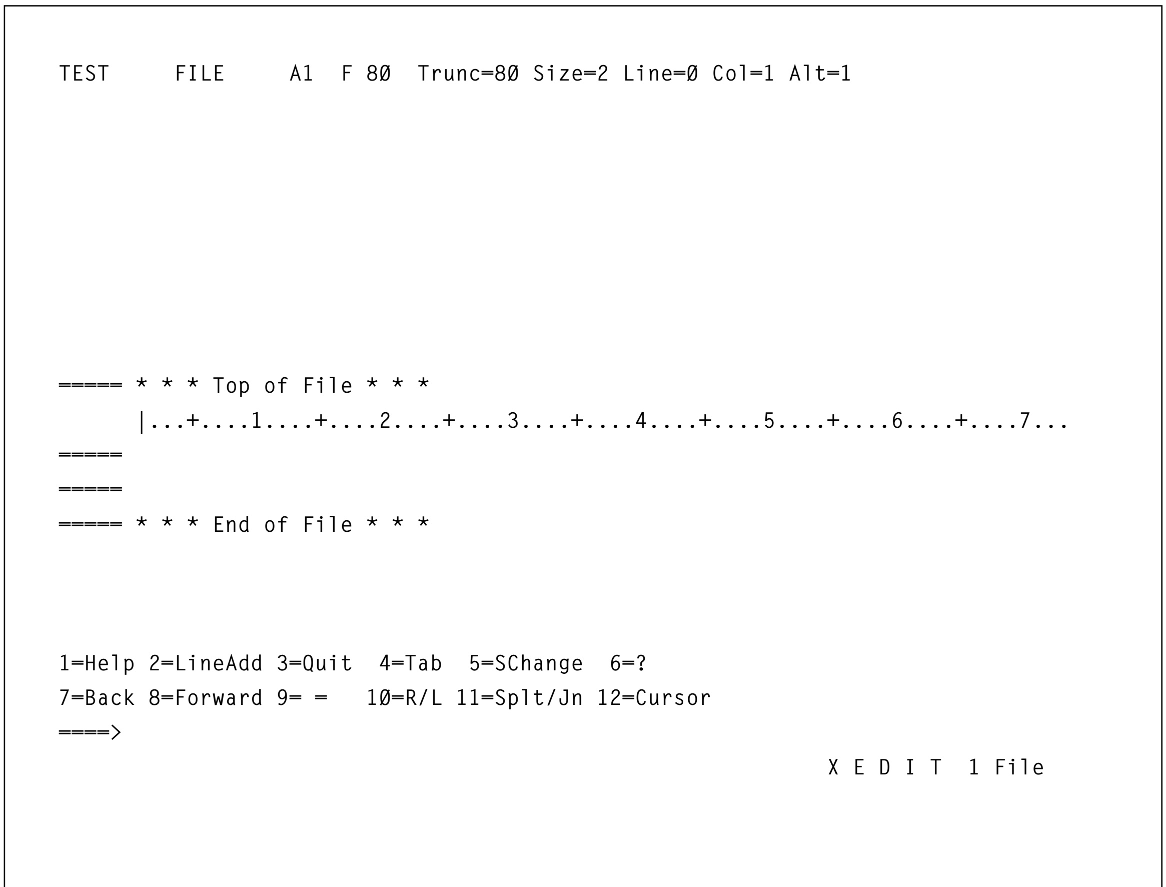
```
/* PROFILE XEDIT setting reserved lines */
.
.
```

```
.
'RESERVE -4 Y N 1=Help 2=LineAdd 3=Quit  4=Tab  5=SChange  6=?'
'RESERVE -3 Y N 7=Back 8=Forward 9= =   10=R/L 11=Splt/Jn 12=Cursor'
Exit
```

where:

- ‘RESERVE’ is the XEDIT subcommand.
- ‘-4’ designates the line which is fourth from the bottom.
- ‘Y’ designates the colour Yellow.
- ‘N’ indicates No highlighting.
- ‘1=Help...’ is the text that is to be displayed in yellow on line 4.

Now when XEDIT is started, the PROFILE XEDIT macro runs and produces an XEDIT screen that looks similar to Figure 4. A little more work is required to ‘stretch out’ the help text on the reserved lines to give them a more pleasing aspect.



```
TEST     FILE     A1  F 80  Trunc=80 Size=2 Line=0 Col=1 Alt=1









===== * * * Top of File * * *
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
=====
=====
===== * * * End of File * * *



1=Help 2=LineAdd 3=Quit  4=Tab  5=SChange  6=?
7=Back 8=Forward 9= =   10=R/L 11=Splt/Jn 12=Cursor
====>
                                                X E D I T  1 File
```

Figure 4: The XEDIT screen with reserved lines

ASSIGNING DIFFERENT FUNCTIONS TO THE PF KEYS

So far the PF key default assignments have not been altered, although now with reserved lines we can more readily remember what the defaults are. If the default PF key assignments are not suitable for a task or a user they can be easily changed within the PROFILE XEDIT (or any other macro, for that matter).

```
/* PROFILE XEDIT assigning alternate functions to PF keys */
.
.
.
'SET PF3 QQUIT'
'SET PF12 EXEC FILELIST'
.
.
.
'RESERVE -4 W N 1=Help 2=LineAdd 3=QQuit .... 12=Filelist'
Exit
```

Here, PF key 3 is assigned the XEDIT subcommand ‘QQUIT’ (unprotected Quit) while PF key 12 is assigned the CMS command ‘EXEC FILELIST’. The corresponding reserved line help text is altered.

THE EXTRACT SUBCOMMAND

XEDIT retains a good deal of information about the editing session and about the file which is being edited. Much of this information can be retrieved by an XEDIT macro with the EXTRACT subcommand. For example:

```
     'EXTRACT /CMDLINE/CURSOR/'
```

retrieves information about the XEDIT command line and about the current position of the XEDIT cursor. Command line information is stored in the CMDLINE. stem variable, and cursor information is stored in the CURSOR. stem variable. For details about the EXTRACT subcommand or about the specific values which are returned, please see the appropriate help file (‘HELP XEDIT EXTRACT’) or other documentation.

THE HOTKEYS MACRO

If the XEDIT reserved lines contain help text of the form ‘1=Help

2=Addline 3=Quit ...' then it is possible to view the reserved line help text as a series of 'hot spots' delimited by one or more blanks. Clicking on a hot spot initiates the action assigned to the corresponding PF key. In this example, mouse-clicking on the string '1=Help' could call up XEDIT help, and mouse-clicking on the string '3=Quit' could initiate an XEDIT protected Quit. The trick is to convert a mouse click into the proper PF key invocation.

PETs programs are designed to respond to appropriate mouse clicks. For this to occur, the 3270 terminal emulation software on a PC or workstation must provide for a mouse action (such as single-clicking the right mouse button), which emulates two 3270 actions – 'set cursor' and 'press ENTER'. Some emulators provide this function by default. Some emulators can be configured to work properly. Others cannot be configured to provide the desired keystroke emulation at all.

If a mouse action successfully emulates 'set cursor' and 'press ENTER', then information about the cursor location and which 'key' was pressed is passed to XEDIT, just as if the 3270 cursor had been repositioned with the arrow keys and the real ENTER key had been pressed.

Since a mouse click essentially translates into setting the 3270 cursor and pressing the ENTER key, XEDIT has to be provided with a way to determine whether a particular ENTER keystroke is intended to invoke a PF key attached function, or whether it is just a standard ENTER keystroke. If the 3270 cursor is on a reserved line when ENTER is 'pressed', then we intend a PF key to be invoked; otherwise we expect XEDIT to handle the ENTER normally. In effect, we have to provide an ENTER key 'filter' which makes the determination based on the location of the cursor, and we have to make sure that the filter will be invoked every time an ENTER keystroke is detected. The filtering function is provided by the HOTKEYS XEDIT macro, provided below. To ensure that HOTKEYS is invoked each time an ENTER keystroke is detected, we must redefine the meaning of the ENTER key, as processed by XEDIT.

In a manner similar to assigning alternate functions to PF keys, the XEDIT ENTER key can be redefined in a PROFILE XEDIT macro:

```
      'SET ENTER BEFORE MACRO HOTKEYS'
```

In this case we instruct XEDIT to run the macro HOTKEYS whenever the ENTER key is pressed and before processing any command line command. Every time the ENTER key is pressed (or the mouse is clicked), XEDIT processes HOTKEYS XEDIT. This might seem a bit wasteful of computing resources, but in practice there is no noticeable delay, since in most cases very few REXX instructions are interpreted.

The HOTKEYS macro relies on the fact that XEDIT places the current location of the cursor in the CURSOR. stem variable. HOTKEYS needs the cursor location to determine whether reserved line help text has been clicked and, if so, which PF key was selected.

The logic of HOTKEYS is as follows:

- HOTKEYS determines whether the cursor's last position was on the command line, and if so it places the cursor back on the command line and exits.
- HOTKEYS determines whether the cursor's last position was 'outside' the XEDIT screen text area; if so, HOTKEYS determines if the cursor was positioned on a reserved line containing help text of the form 'n=function'; if so, HOTKEYS determines which PF key was selected and executes that function; HOTKEYS then exits.
- Otherwise HOTKEYS sets the cursor on the command line and exits.

The HOTKEYS XEDIT macro follows:

```
/* HOTKEYS XEDIT - Making Reserved Lines Clickable                  */
'EXT/CMD/CURS/LS/RESERVE */'                    /* extract key values */

Select;
   When (cursor.1=cmdline.2) Then p='CMDLINE'   /* cursor on CMDLINE? */
   When (cursor.3<Ø)                            /* cursor not in file?*/
      Then
         Do
            p='CMDLINE P 255'                   /* set cursor location*/
            nrl=cursor.1-lscreen.1-1            /* negative res line  */
            prl=cursor.1                        /* positive res line  */
            Do ri=1 To reserved.Ø               /* examine res lines  */
               rl=Word(reserved.ri,1)           /* res line number?   */
If (rl=nrl | rl=prl)                            /* cursor on res line?*/
                  Then                          /* yes!               */
                     Do
```

```
                        f=GETKEY()              /* perform  function  */
                        Leave ri                /* leave              */
                        End
                  End
               End
   Otherwise p='CMDLINE P 3Ø'                   /* otherwise location */
   End
'CURSOR' p                                      /* set the cursor     */
Exit(Ø)

GETKEY:
If(Pos('=',reserved.ri)=Ø) Then Return('')      /* return if no "="   */
Parse Var reserved.ri . . . . . rline           /* get res line text  */

/* These next lines attempt to isolate which PF key help text of the  */
/* form n=function has been selected.  Variable 'sline' eventually    */
/* contains just the limited string 'n=function' which is then itself */
/* parsed to extract the number of the PF key.                        */

If (Substr(rline,cursor.2,1)=' ')
   Then sline=Strip(Substr(rline,1,cursor.2),'T')
   Else sline=Strip(Substr(rline' ',1,Pos(' ',rline' ',cursor.2)))
If (Pos('=',sline)=Ø)
   Then sline=' 'Substr(rline,1,Pos('=',rline))
sline=Substr(' 'sline,Lastpos(' ',' 'sline,Lastpos('=',sline)))

Parse Var sline n'='label                       /* get PF number      */
n=Strip(n)                                      /* just the number    */
If (Datatype(n)¬='NUM' | n<1 | n>24)            /* valid PF key?      */
   Then 'MSG PFKEY labels must be of the form', /* error message      */
        '1=label 2=label 3=label ... 24=label'
   Else 'SOS PF'n                               /* perform function   */
Return('')
```

In summary, to enable mouse clicks to activate PF key attached functions, one must do the following:

- Ensure that the 3270 emulation software supports a mouse action, which emulates ‘set the 3270 cursor’ and ‘press ENTER’.
- Ensure access to the HOTKEYS XEDIT macro.
- Assign appropriate functions to the XEDIT PF keys.
- Add reserved lines to the XEDIT screen, with help text in the appropriate form (‘1=Help 2=Addline 3=Quit ...’).
- Redefine the ‘meaning’ of the ENTER key in the PROFILE XEDIT macro with the command:

```
'SET ENTER BEFORE MACRO HOTKEYS'
```

- Try it out.

Users well-versed in 3270 emulation software will recognize that many terminal emulators provide for ‘hot spots’ of one sort or another. Some emulators ‘recognize’ when PF key help text (eg ‘1=Help’) is clicked and subsequently emulate pressing the corresponding PF key. But there are some subtle differences in this support as offered by the emulators and that provided by the REXX/XEDIT techniques described in this section.

For example, an emulator might emulate ‘press PF1’ if the ‘1’ in the reserved line help text is clicked upon, but not if the word ‘Help’ is clicked. More problematically, that same emulator might emulate ‘press PF1’ if *any* ‘1’ on the screen is clicked! With the PETs approach, any part of the string ‘1=Help’ can be clicked, but no action is taken if the string ‘1=Help’ appears in the text area (in the file).

More importantly, the coding techniques described in this section are generalizable; macros can be written so that *any* screen area or content can be enabled for mouse clicks. For example, a macro could be written to turn the XEDIT prefix area into a ‘scrolling’ area; clicking on the prefix area associated with any line positions that line as the *current* line; clicking on the top-most visible prefix area scrolls the file backward (ie issues the BACKWARD subcommand).

ADDING POP-UP MENUS TO XEDIT: THE KEYWIN MACRO

Adding reserved line help text to the XEDIT screen assists users in remembering PF key assignments. Reserved lines also serve as ‘targets’ for mouse clicks, as described in the previous section. But using reserved lines takes up precious 3270 screen real estate – lines reserved for help text cannot be used to display file text. A simple solution to this problem is to use a 32 or 43 line screen emulation, rather than the more common 24 line emulation. Most 3270 emulators offer these alternative screen sizes.

Another limitation has to do with the number of PF keys that can be assigned functions within XEDIT. At most, 24 keys can be assigned. And assuming that help text for six PF keys fits comfortably on an 80 column reserved line, then four reserved lines are required to show help text for all 24 PF keys. This can be done, but the screen begins to look crowded.

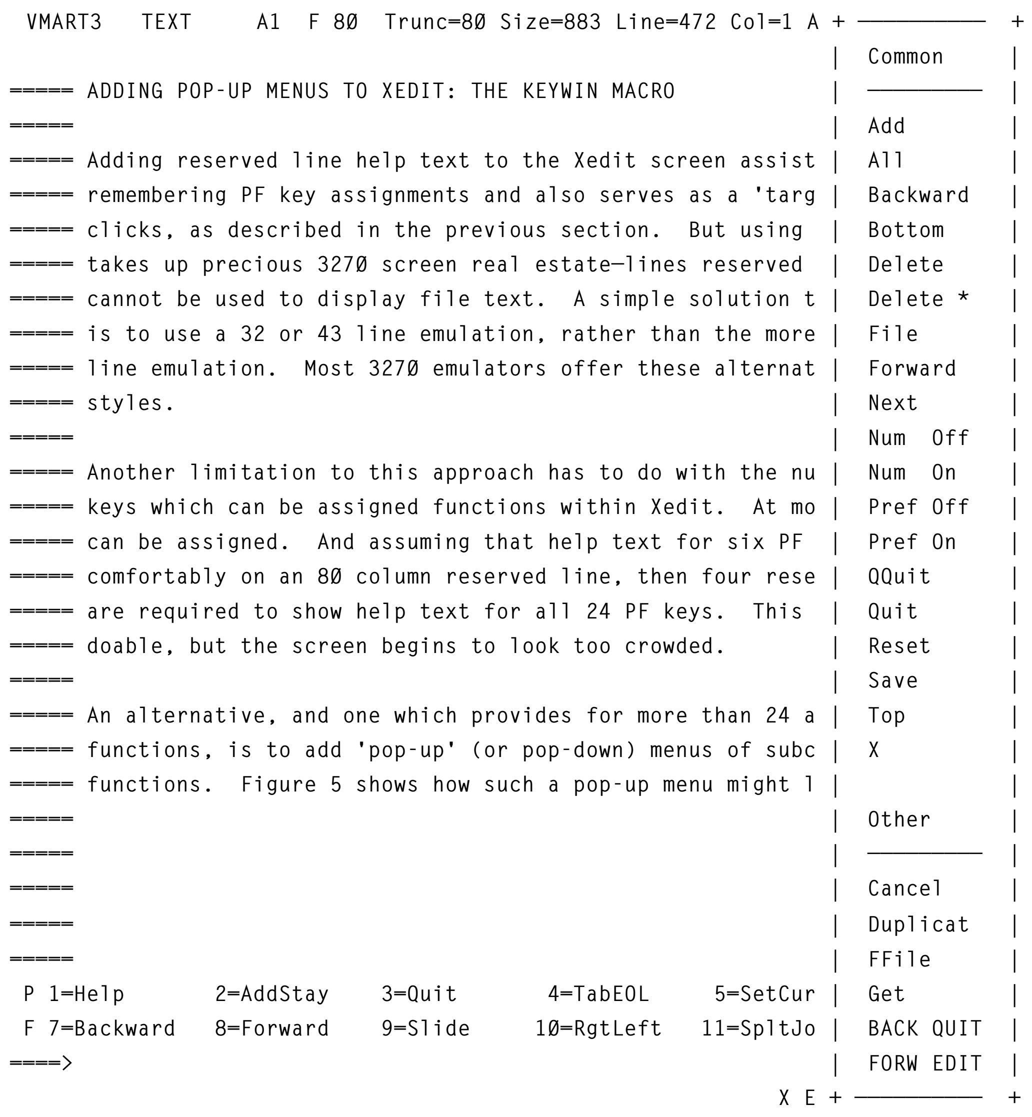
An alternative, and one which provides for more than 24 clickable functions, is to add 'pop-up' (or drop-down) menus of subcommands and functions. Figure 5 shows how such a pop-up menu might look.

The menu of subcommands which overlays the XEDIT screen in Figure 5 is contained in a virtual screen and presented through a CMS window. Any subcommand can be invoked by mouse-clicking on that subcommand (or by repositioning the 3270 with arrow keys and pressing the ENTER key). Certain subcommands, such as QUIT, cause the window to be closed before the subcommand is executed. Other subcommands are executed but the window remains open so that other actions can be selected.

The bottom two lines in the window contain the control functions, 'BACK QUIT' and 'FORW EDIT' which, on a colour display, are highlighted in yellow to distinguish them from the subcommands in the menu. When mouse-clicked, these controls scroll forward or backward through the list of subcommands, facilitate changing the menu on-the-fly (EDIT), or close the menu (QUIT). Clicking outside the window also closes the menu.

The specific menu items are listed in a file. In this case, the file is named XCMDS KEYWIN and contains the following lines:

```
                 'Common
                 '——————
add              'Add
all              'All
backward         'Backward
bottom           'Bottom
delete           'Delete
delete *         'Delete *
file             'File
forward          'Forward
next             'Next
num off          'Num  Off
num on           'Num  On
prefix off       'Pref Off
prefix on        'Pref On
qquit            'QQuit
quit             'Quit
```



```
 VMART3   TEXT     A1  F 80  Trunc=80 Size=883 Line=472 Col=1 A + ---------- +
                                                               | Common     |
===== ADDING POP-UP MENUS TO XEDIT: THE KEYWIN MACRO           | ---------  |
=====                                                          | Add        |
===== Adding reserved line help text to the Xedit screen assist | All       |
===== remembering PF key assignments and also serves as a 'targ | Backward  |
===== clicks, as described in the previous section.  But using  | Bottom    |
===== takes up precious 3270 screen real estate—lines reserved  | Delete    |
===== cannot be used to display file text.  A simple solution t | Delete *  |
===== is to use a 32 or 43 line emulation, rather than the more | File      |
===== line emulation.  Most 3270 emulators offer these alternat | Forward   |
===== styles.                                                   | Next      |
=====                                                           | Num  Off  |
===== Another limitation to this approach has to do with the nu | Num  On   |
===== keys which can be assigned functions within Xedit.  At mo | Pref Off  |
===== can be assigned.  And assuming that help text for six PF  | Pref On   |
===== comfortably on an 80 column reserved line, then four rese | QQuit     |
===== are required to show help text for all 24 PF keys.  This  | Quit      |
===== doable, but the screen begins to look too crowded.        | Reset     |
=====                                                           | Save      |
===== An alternative, and one which provides for more than 24 a | Top       |
===== functions, is to add 'pop-up' (or pop-down) menus of subc | X         |
===== functions.  Figure 5 shows how such a pop-up menu might l |           |
=====                                                           | Other     |
=====                                                           | --------- |
=====                                                           | Cancel    |
=====                                                           | Duplicat  |
=====                                                           | FFile     |
 P 1=Help      2=AddStay    3=Quit       4=TabEOL     5=SetCur | Get       |
 F 7=Backward  8=Forward    9=Slide     10=RgtLeft   11=SpltJo | BACK QUIT |
====>                                                           | FORW EDIT |
                                                          X E + ---------- +
```

Figure 5: XEDIT screen with pop-up menu of subcommands

```
reset           'Reset
save            'Save
top             'Top
x               'X

                'Other
                '-----

cancel          'Cancel
duplicat        'Duplicat
```

```
file              'FFile
get               'Get
help xedit menu   'Help
hextype 1         'HexType/1
input             'Input
left 10           'Left/10
lowercas          'LowerCas
powerinp          'PowerInp
put *             'Put/*
recover 1         'Recover/1
rgtleft           'RgtLeft
right 10          'Right/10
scale off         'Scale Off
scale on m        'Scale On
screen 1          'Screen/1
screen 2 h        'Screen/2H
screen 2 v        'Screen/2V
ssave             'SSave
up                'Up
uppercas          'UpperCas
verify 1 *        'Ver 1 *
verify h 1 *      'Ver h 1 *
```

The left column contains the exact text of the subcommands (or other CMS commands or EXECs) while the right column contains the descriptions that show in the menu window. A single quote (') separates a command from its description.

The KEYWIN XEDIT macro is invoked as follows:

```
     'KEYWIN 6 XCMDS'
```

where:

- '6' is the screen position for the window.
- 'XCMDS' is the filename of the menu ('KEYWIN' is the filetype).

KEYWIN offers six screen positions, with screen position '1' being leftmost and screen position '6' being rightmost. Screen positions 7-12 can also be specified, but '7' is equivalent to '1' and so forth.

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

Richard G Ellis
Director of Computing and Information Systems
University of Connecticut (USA)

VM news

IBM has announced Version 6 Release 1 of its DB2 Server for VM and VSE. Version 6.1 extends e-business capabilities with TCP/IP for VM, provides faster access to distributed data through the use of stored procedures, simplifies information access with QMF for Windows, shortens the back-up window with its incremental archive feature, and extends distributed database solutions. It also comes with Y2K readiness and euro support.

With the DRDA RUOW Online Application Requester, programs can execute SQL statements to access and manipulate data managed by any remote application server that implements RDA. DRDA RUOW can be used over a TCP/IP network to connect databases, and users can also choose to secure TCP/IP connections using any external security manager that supports the RACROUTE interface.

A Control Center management option lets VM/ESA users take advantage of shared file system support and CA-DYNAM/T interface support.

For further information contact your local IBM representative.

* * *

VM users can benefit from Qualex Consulting Services' Find2000 search tool for resolving Y2K compliance issues in SAS code and data. Its wildcard capability allows for the searching of numbers, characters, and various date formats and informats.

The point and click interface searches multiple catalog entries, including SOURCE, SCL, PROGRAM, LOG, OUTPUT, HELP, CBT, and CATAMS.

Find2000 includes the capability to search variable values in SAS datasets, the SCL associated with FSEDIT SCREEN entries, SAS format/informat range and label values, the format/informat names assigned to SAS dataset variables, SLIST entries, CLASS entries, FRAME entries, and EIS entries. It also includes a program that downloads all members of a mainframe-partitioned dataset to a pre-specified directory.

For further information contact:
Qualex Consulting Services, 382 Fox Chase Drive, Collinsville, VA 24078, USA.
Tel (919) 380 8284.
URL: http://www.qlx.com.

* * *

IBM has announced ADSTAR Distributed Storage Manager for VM/ESA Version 3. This provides ADSM Version 3 function, including enterprise management enhancements. These include new control features and the availability of ADSM Connect Agents to back up databases and applications on-line.

For further information contact your local IBM representative.

* * *