166 VM update

Winter 2001

In this issue

VM Update

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

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

Editorial Panel
Chuck Meyer, Chuck Meyer Systems, Inc., USA.

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

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

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

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

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

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

Page headers for PIPES

PIPES has been a boon, not only for the serious CMS programmer, but also for the casual CMS user who is capable of typing simple CMS commands which use PIPES. The URO, MCTOASA, and ASATOMC stages provide easy access to create "punched" or "printed" output. And the SPEC stage can be used to preface each record with a carriage-control character. But unfortunately, none of these stages has the capability to format printed pages to include page-headers.

Suppose, for example, you have a tape-manager command that displays a list of tapes to be pulled for off-site storage. That list could contain any number of records (from none to hundreds), and the tape librarian needs a printed copy of it. It would be helpful if the printed list had page headers with useful information such as date, time, and page number.

PGHDR is a PIPE stage that will insert a user-defined (or default) page header into appropriate points of an input stream. The user may define a page header containing any number of lines, and may request that a page number be inserted anywhere within that page-header.

The stage's primary input stream contains the records to be "printed," and carriage-control characters must be provided by either:

1. Already existing CCCs in column-1 of every input record, or

2. Requesting PGHDR to insert a specified CCC into each record.

This user-defined page header may be communicated to PGHDR by either a secondary input stream, or by data passed directly on the PGHDR command. Each line of the page header must begin with a CCC.

A simple default page header is also available. It is similar to that given with CMS's PRINT command, plus a time stamp.

Page size is the total number of printable lines available on each page. This number can be communicated on the PGHDR command, or will

default to either that specifed on the `SPOOL E LPP nnn` command, or to 60. Page saturation is then calculated from the CCCs of the header lines and data lines. PGHDR will process both ANSI and MC CCCs, but in any one invocation of PGHDR, only one type should be used. If you intermix ANSI and MC CCCs, no error will be noted, but the output will appear strange, with unexpected overprinting and/or blank lines.

Example code

So you may now provide the tape librarian with the desired list of tapes to be pulled tomorrow by using the following code:

```
h. = ''
h.1 = '1Tapes to be pulled for' DATE('N',DATE('B')+1,'B') 'Page ~'
h.2 = ' Listing created on' DATE() 'at' TIME('N') 'by' USERID()
h.0 = 3

'PIPE (END ?)'   ,
   '| COMMAND' hey_tape_manager , /* TapMgr dsplys tapes list */
   '| SPEC 1-* 2'               , /* create a blank CCC */
   '|F:PGHDR LPP=50'            , /* 50 lines-per-page */
   '| URO 00E'                  , /* '| > TAPEPULL LISTING A' */
  '? STEM H.'                   , /* Page-headers to be fed */
   '|F:'                          /* into PGHDR (stream#1) */

Call DIAG 8,'CLOSE 000E CLASS L NAME TAPEPULL LISTING'
```

Note the tilde character, "~", (hex-A0) in the header line. It may appear anywhere in the header, and it will be replaced by the correct page number, left-justified. Using two or more adjacent tildes will result in a right-justified page number.

It is also possible to specify "end-of-report" line(s), which can include record counts—similar to report writers that end reports by adding a line like "nnnn Records Selected." This is specified like another header line, but prefixed with a dot. In the preceding example, we could have added/updated the following two lines:

```
h.4 = '.- ~ Tapes should be pulled'
h.0 = 4
```

The dot, ".", indicates an end-of-report line. The hyphen, "-", is an ANSI CCC to write-after-spacing-2. The tilde, "~", will be replaced by the record count. The H.0 value must be increased to cover the new H.4.

As an added bonus, a copy of SPACER.REXX has been included. This stage adds a blank record whenever it detects a change of value in a specified field. For example, suppose our tape manager output includes the VolSer in columns 1-6, and we would like to insert a blank line whenever the first three characters of VolSer change. We could add

```
'| SPACER 1.3'  , /* add a rec at ctl-break */
```

immediately following the COMMAND stage.

Advanced capabilities

- Add sequence numbers and/or LRECLs to each line.
- Start output with page numbers and/or sequence numbers other than 1.
- Save the next page number and next sequence number from one PIPE so that a subsequent PIPE can continue output with ever-increasing numbers.
- Dynamically add one line to the default page header. This line will include file information about a single CMS file.

This package includes four files

- PGHDR.REXX — the stage to insert page headers
- PGHDR.HELPPIPE — help
- PGHDR.EXEC — a sample program
- SPACER.REXX — the stage to insert a blank line at control-break

Installation instructions

1. Besides VM/ESA and PIPES, the only pre-requisite is VMARC, for unpacking the compressed files distributed by *VM Update*.
2. Upload one file to VM/CMS — PGHDR VMARC. It is in VMARC format, and must be uploaded as binary (not ASCII).

3. Issue the command

   ```
   PIPE < PGHDR VMARC|FBLOCK 80 00|>PGHDR2 VMARC A F 80
   ```

   Depending on your terminal emulator and version of VMARC, this step probably is not necessary, but it can't hurt.

4. Issue the command

   ```
   VMARC UNPACK PGHDR2 VMARC A * * T
   ```

   to extract all six files: the four run-time files noted above, plus PACKAGE and INFO files.

5. Copy the four run-time files to appropriate mdisks.

Download Caution

The following characters are used in both the .REXX and .HELPPIPE files, and are occasionally mistranslated by 3270 emulators. Please verify the correct translation of these characters.

¢¢¢¢	4A	cent sign	REXX label	HELP highlight
| | | |	4F	solid vertical bar	REXX or	PIPE stage delimiter
! ! ! !	5A	exclamation point	REXX label	PIPE or
¬¬¬¬	5F	not sign	REXX not	PIPE (not used)
????	6F	question mark	REXX label	
~~	A1	tilde		
©©©	B4	copyright symbol		
\ \ \ \	E0	reverse slash	REXX not	

PGHDR REXX

```
/*********************************************************************
*** 253-line source for PGHDR REXX X6 (2000-02-21 9:03:10) follows ...
*********************************************************************/

/*********************************************************************\
   Insert PageHeader(s) at appropriate points in a data-stream which
    contains otherwise print-ready lines.
   The desired HeaderLine(s) may be defined either -
      + as input to a secondary stage
      + as operands of the PGHDR stage command
      + or a default is created.

   Use "HELP PIPE PGHDR" a detailed description of this program.

 1997-07-07 CHM Written
 1997-08-05 CHM comments updated
 1997-09-05 CHM Correct the use of ".BL" from caller
 1999-11-05 CHM Replace header-stem-in-callers-program (which frequently
                did not work), with headers-via-secondary-input-stream
                Rename from RPT to PGHDR
 2000-01-13 CHM LPP now total LinesPerPage (not just Detail lines);
                ReportFooter now set via HdrLines starting w/ ".";
                INFILE=fn.ft.fm may be used to enhance default pg-hdrs.
 2000-02-21 CHM Minimize use of special-chars to ease upload/download
\*********************************************************************/
copyright = '© Copyright: Chuck Meyer Systems, Inc.;  1997, 2000 '  ,
            '(cmsi@attglobal.net)'
version   = '2000.02.21'

Trace  OFF
Parse  Upper Source . how_called myfn myft myfm myalias myenv .
Call   INITXX ARG(1)              /* Process the parms             */
Signal ON  NOVALUE

Do page = pgstart  Until ?fini    /* Build a page at a time        */
  ltp = hlc                       /* Number of lines on this page  */
  Do i=1  Until (ltp >= lpp)      /* Read enuf records for one page */
    'READTO R'
    ?fini = \(rc=0)
    If ?fini  Then Leave i
    r     = ccc || r              /* preface possible CCC          */
    Parse Var r  c9 +1 r9         /*   separate ccc and data       */
    If (i=1)  Then ,              /* first detail-line on page     */
      If (POS(c9,'F060F14E'x)>0)  Then r = ' 'r9   /* Space1-B4-Prt */
    len   = LENGTH(r9)            /* length of data without CCC    */
    ltp   = ACCC(ltp,r)           /* Update LinesThisPage          */
    If ?debug Then r = INSERT(RIGHT(ltp ,    2)' ',r,1)
    If ?sqnm  Then r = INSERT(RIGHT(sqnm,sqlen)' ',r,1)
    If ?recl  Then r = INSERT(RIGHT(len ,recln)' ',r,1)
    l.i   = STRIP(r,'T')
    l.0   = i
    sqnm  = sqnm + 1
    End  /* Do i=1 ... */
  sqnm1 = sqnm - 1
  If \(SYMBOL("L.0")='VAR')  Then l.0 = 0
  If \DATATYPE(l.0,  'W')    Then l.0 = 0
```

```
  If ?fini  &   (eor.0>0)          Then /* insert a BottomLine    */
    'CALLPIPE STEM   EOR.'          ,    /* from user's definition */
            '| CHANGE ;~;'sqnm1';'  ,    /* w/ possible RecCount   */
            '| STEM   L. APPEND'         /* append to output stem  */
  If (l.0>0)  Then              /* Write an output page w/ hdrs   */
    'CALLPIPE STEM   H.'                       , /* Header array   */
            '| LOCATE 1'                       , /* no NULL recs   */
            '| CHANGE ;~~~~~;'RIGHT(page,5)';' ,
            '| CHANGE  ;~~~~;'RIGHT(page,4)';' ,
            '| CHANGE   ;~~~;'RIGHT(page,3)';' ,
            '| CHANGE    ;~~;'RIGHT(page,2)';' ,
            '| CHANGE     ;~;'    ||page|| ';' ,
            '| APPEND STEM L.'                 , /* Data Lines     */
            '| XLATE  1-* 00 40'              , /* Possibly save  */
            '| STRIP  TRAILING'               , /*   spool space  */
            '| PAD    1'                      , /* no NULL recs   */
            '| *:'
  Drop l.

  End  /* Do page=pgstart ... */

Parse Value (-3) (page+1-(i=1)) (sqnm+0)  With rc p s .
pb = SPACE('Pgstart='p 'SQStart='s 'LPp='lpp  , /* PassBack value */
           WORD('SQLn='sqlen  ,2-?sqnm )      ,
           WORD('LN='recln    ,2-?recl )      ,
           WORD('Ccc='C2X(ccc),2-LENGTH(ccc)) ,
           WORD('Debug'       ,2-?debug)       )
If ?pb     Then Do
  'SELECT OUTPUT' ost
  If (rc=0)  Then 'OUTPUT ' pb
  End
If (rc\=0) & ?say  Then  ,
  'MESSAGE' myfn'.'myft 'PassBackValues : "'pb'"'

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

INITXX:
  Parse Arg a1
  'CALLPIPE (END ¢)'                          ,
    ' COMMAND IDENTIFY |SPEC W3|VAR CMNODE'   ,  /* CMS  NodeID       */
    '¢CP QUERY USERID  |SPEC W3|VAR CPNODE'   ,  /* CP's NodeID       */
    '¢CP QUERY T|TAKE 1|SPEC W4|VAR TZ'       ,  /* CP's TimeZone ID  */
    '¢CP Q V PRT|APPEND LITERAL LPP X|SPLIT' ,
       '|INSIDE /LPP/ 1|TAKE 1|VAR CPLPP'      /* CP's LinesPerPage */
  node    = cmnode || WORD('('cpnode')',1+(cmnode=cpnode))
  ?debug  = 0                        /* Test-Mode???                */
  ?say    = 0                        /* Should we SAY some numbers?? */
  h.      = 0                        /* PageHeader array            */
  eor.    = 0                        /* End-Of-Report array         */
  ctl.    = 0                        /* ControlStatement array      */
  ctldef  = 'pgstart=1 recln=0 sqlen=0 sqstart=1 sqnm=1' ,
            'lpp='WORD('60' cplpp,1+DATATYPE(cplpp,'W'))
  cmi     =           STRIP(STORAGE(220,64))          /* "PRINT" HDR */
  cmi     = TRANSLATE(SPACE(STORAGE(200,32)),'_',' ')  /* CMS IPL-ID  */
  h1      = '1'USERID()'@'node'    'LEFT(DATE('W'),3)','          ,
            TRANSLATE('CcYy/Mm/Dd','-'DATE('S'),'/CcYyMmDd')    ,
            '('TRANSLATE(  'Yy/123'  ,'/'DATE('J'),'/Yy123'   )')' ,
            TIME('N') tz'    'cmi'    Page ~'
```

```
h2      = ''   /* may be overriden by caller's "INFILE=xxx" param   */
ansiccc = '+ 0-123456789ABC'             /* Valid ANSI CCC's        */
ost     = STRMNO('OUTPUT') /* highest connected output stream numbr */
ist     = STRMNO( 'INPUT') /* highest connected  input stream numbr */
?pb     = (ost>0)          /* should we PassBack some numbers??     */
If (ist>0) , /* If Stream#1 is present, use it to populate 2 arrays */
  Then 'CALLPIPE (END c) *.INPUT.'ist':' ,
       '|A:FIND *|SPEC 2-*|STEM CTL.' , /* opt. Parameter   values  */
    'cA:|                 STEM H.'     /* opt. PageHeader  line(s) */
'CALLPIPE STEM CTL.|APPEND VAR CTLDEF|SPLIT|BUFFER|STEM CTL.'
Do i=1  For ctl.0                              /* analyze all CTLs  */
  Parse Upper Var ctl.i  nam '=' val .   /* VariableName and Value  */
  Select
    When (SYMBOL(nam)='BAD')  Then Nop   /* NG VarName,  so ignore  */
    When (SYMBOL(nam)='VAR')  Then Nop   /* Already set, so ignore  */
    When  \DATATYPE(val,'W')  Then Nop   /* Not numeric, so ignore  */
    Otherwise Interpret nam '=' val      /* Gopher it               */
    End  /*  Select  */
  End  /*  Do i=1  */
If (LENGTH(ccc)\=1)  |  (POS(ccc,ansiccc)=0)  Then ccc = ''
Do While \(a1='')              /* Check params entered w/ StageName */
  Parse Value STRIP(a1)  With  sep +1 a1sav  1 w1 a1
  Parse Upper Var w1   w1l '=' w1r
  ?i = DATATYPE(w1r,'W')
  ?x = DATATYPE(w1r,'X')     &  (LENGTH(w1r)=2)
  ?c = (POS(w1r,ansiccc)>0)  &  (LENGTH(w1r)=1)
  Select
    When ABBREV(  'CCC'     , w1l,1) & ?x  Then  ccc     = X2C(w1r)
    When ABBREV(  'CCC'     , w1l,1) & ?c  Then  ccc     =     w1r
    When ABBREV(  'DEBUG'   , w1l,1)       Then  ?debug  =     1
    When ABBREV(  'INFILE'  , w1l,1)       Then  h2      = FDX(w1r)
    When ABBREV(  'LENGTH'  , w1l,2) & ?i  Then  recln   =     w1r
    When ABBREV(  'LN'      , w1l,2) & ?i  Then  recln   =     w1r
    When ABBREV(  'LPP'     , w1l,2) & ?i  Then  lpp     =     w1r
    When ABBREV(  'PAGE'    , w1l,1) & ?i  Then  pgstart =     w1r
    When ABBREV(  'PGSTART' , w1l,1) & ?i  Then  pgstart =     w1r
    When ABBREV(  'SAY'     , w1l,2)       Then  ?say    =     1
    When ABBREV('NOSAY'     , w1l,4)       Then  ?say    =     0
    When ABBREV(  'SQLENGTH', w1l,3) & ?i  Then  sqlen   =     w1r
    When ABBREV(  'SQLN'    , w1l,3) & ?i  Then  sqlen   =     w1r
    When ABBREV(  'SQSTART' , w1l,3) & ?i  Then  sqnm    =     w1r
    When \DATATYPE(sep,'A')     & (h.0=0)  Then  'CALLPIPE' ,
      '|VAR A1SAV|SPLIT X'C2X(sep)'|STEM H.|HOLE|LITERAL|VAR A1'
    When ?say Then 'MESSAGE' myfn'.'myft 'operand "'w1'" is not' ,
                   'understood;  it is ignored.'
    Otherwise Nop   /* Invalid, but no message */
    End  /* Select */
  End  /* Do While ... */

If ?debug & (sqlen=0)  Then sqlen=3
If ?debug & (recln=0)  Then recln=3
?sqnm   = (sqlen>0)   /* Should we insert a SequenceNumber?? */
?recl   = (recln>0)   /* Should we insert a RecordLength??   */
If \DATATYPE(sqnm,'W')     ,
  Then sqnm = WORD('0' sqstart , 1+DATATYPE(sqstart,'W'))
   /* If no HdrLines yet found, we'll build a default set */
If (h.0<1)  Then  'CALLPIPE VAR H1' ,
  '|APPEND VAR H2|APPEND LITERAL 0|STRIP TRAILING|LOCATE 1|STEM H.'
'CALLPIPE (END ¢) STEM H.|A:FIND .|SPEC 2-*|STEM EOR.' ,
```

```
                    '¢A:|BUFFER|STEM H.'
  Parse Value '0 0' With hlc flc .
  Do i=1  For h.0          /* Get Num-of-Lines  */
    hlc = ACCC(hlc,h.i)    /*  occupied by      */
    End                    /*   HeaderLines     */
  Do i=1  For eor.0        /* Get Num-of-Lines  */
    flc = ACCC(flc,eor.i) /*  occupied by      */
    End                    /*   EndOfReportLines*/
  Drop a1 a1sav i j ?i ?x ?c rc.
  Return

STRMNO: Procedure /* Determine highest connected IN|OUT stream num */
  Parse Upper Arg io .
  io = WORD('INPUT OUTPUT' , 1+ABBREV('OUTPUT',io,1))
  'MAXSTREAM' io          /* Find the MAX in/out stream AVAILABLE   */
  Do str=rc  By -1  For rc  Until ((rc<9) & (rc>-1))
    'STREAMSTATE' io str /* Search hi-2-lo for first CONN'D stream */
    End
  Return str

ACCC:  Procedure  Expose  ?np ?an     /* Add a CCC to a line-count */
  Parse Arg n . , lin      /* cur-line-number , data-line w/ CCC   */
  cc  = LEFT(lin,1)        /* isolate the CarriageControlCharacter */
  ?np = 0                  /* NewPage-indicator is OFF             */
  ?an = (POS(cc,'+ 0-123456789ABC')>0)       /* is it antsy        */
  Select
    When (POS(cc,'03014E'x)>0) Then x = n+0  /* advance 0 lines    */
    When (POS(cc,'0B0940'x)>0) Then x = n+1  /* advance 1 line     */
    When (POS(cc,'1311F0'x)>0) Then x = n+2  /* advance 2 lines    */
    When (POS(cc,'1B1960'x)>0) Then x = n+3  /* advance 3 lines    */
    When (POS(cc,'8B81F1'x)>0) Then x =   1  /* skip to channel 01 */
    When (POS(cc,'9399F2'x)>0) Then x =   6  /* skip to channel 02 */
    When (POS(cc,'9B91F3'x)>0) Then x =  11  /* skip to channel 03 */
    When (POS(cc,'A3A9F4'x)>0) Then x =  16  /* skip to channel 04 */
    When (POS(cc,'ABA1F5'x)>0) Then x =  21  /* skip to channel 05 */
    When (POS(cc,'B3B9F6'x)>0) Then x =  26  /* skip to channel 06 */
    When (POS(cc,'BBB1F7'x)>0) Then x =  31  /* skip to channel 07 */
    When (POS(cc,'C3C9F8'x)>0) Then x =  36  /* skip to channel 08 */
    When (POS(cc,'CBC1F9'x)>0) Then x =  45  /* skip to channel 09 */
    When (POS(cc,'D3D9C1'x)>0) Then x =  50  /* skip to channel 10 */
    When (POS(cc,'DBD1C2'x)>0) Then x =  55  /* skip to channel 11 */
    When (POS(cc,'E3E9C3'x)>0) Then x =  60  /* skip to channel 12 */
    Otherwise                  x = n+0  /* Go figure!!        */
    End
  ?np = (x<n)   /* Set NewPage flag */
  Return x

FDX:  Procedure      /* get all kindsa good info about a CMS file */
    /* The following defines all characters which are NOT valid   */
    /* as part of a CMS FileName, but typically ARE 3270-keyable. */
  charx = '00    15       1C 1D    FF'x ,   /* Dup FldMk */
                    '4A 4B 4C 4D    4F'x ,
          '50       5A    5C 5D 5E 5F'x ,
             '61       6B 6C    6E 6F'x ,
                '79          7D 7E 7F'x ,
             'A1                FF'x ,
          'B0       BA BB          FF'x ,
          'E0                      FF'x
  Parse Upper Value TRANSLATE(ARG(1),,charx)  With fdxn /* Full FnFtFm */
```

```
  'CALLPIPE (END ¢)'                                    ,
       '|COMMAND IDENTIFY|SPEC W3|VAR NODEID'           , /* get NODEID      */
      '¢ VAR     FDXN'                                  ,
       '|SPEC    W1-3'                                  , /* Fn Ft Fm        */
       '|STATE   ISO  |SPLIT|STEM F.|JOIN * X40'        ,
       '|SPEC    /QUERY  DISK/ 1  W3 NW.1'              ,
       '|COMMAND'                                       , /* Q DISK x        */
       '|DROP    1'                                     ,
       '|XLATE   (1.6) 40 _'                            , /* Fix bozo VolSers*/
       '|SPEC    PAD 0  W2 1.4 R  W1 NW  W6-7 NW',
       '|VAR     VADDR'                                 ,
       '|SPEC    /QUERY MDISK/ 1  W1 NW /LOC/ NW',
       '|CP'                                            , /* Q MDISK xxxx    */
       '|DROP    1'                                     ,
       '|SPEC    W3 1 ;\; N W4 N  W6-7 NW'              ,
       '|VAR     OWNER'
  Parse Var vaddr   vaddr vvol dt   blksz .
  Parse Var owner   owner rvol radr .
  Return ' 'nodeid'\'owner'\'f.1'.'f.2'.'f.3 , /* Full File Name */
           f.4||f.5                          , /* RecFm || LRecL */
           f.8'@'RIGHT(f.9,8,'0')            , /* TimeStamp      */
           'Recs='f.6'('blksz/1024*f.7'K)'   , /* #Recs DiskSpace*/
           'Dev='dt'/'radr'/'rvol              /* RealDev Info   */

/*----------------------end of PGHDR.REXX-----------------------------*/
`
```

PGHDR HELPPIPE

```
*************************************************************************
**274-line source for PGHDR HELPPIPE X6 (2000-02-14 10:44:15) follows ..
*************************************************************************

.CS 2 ON
.cm © Copyright Chuck Meyer Systems, Inc. ;  2000   (cmsi@attglobal.net)
.cm 2000-02-14 CHM Written

¢|PGHDR¢%

>>--PGHDR-----.--------------.----------.----------------------.-------><
              | <------------< |            | <------------------< |
              |---Ccc=xx-------|            '---sep---page_header---'
              |---Infile=xxx---|
              |---LEngth=nnn---|
              |---LPp=nn-------|
              |---Pgstart=nnn--|
              |---SQLn=nn------|
              |---SQStart=nn---|
              '---DEBUG--------'


.CS 2 OFF
.CS 1 ON
¢|Purpose¢%

 The PGHDR stage will read "print-ready" records from its primary input
 stream (stream 0), and insert a user-provided page-header (of any number
 of lines) at appropriate points into the stream. The primary output
 stream then contains all of the records from the primary INPUT stream,
 plus header-line(s).

 To find the header-lines, PGHDR searches for the first-found of:
  1. Possible record(s) on the secondary input stream;
  2. Possible operand(s) of the PGHDR stage;
  3. A default header.
 Header-lines may contain a page-number anywhere in them, and it will be
 updated on successive output pages.

.CS 1 OFF
.CS 3 ON
¢|Operands¢%

 Operands have an assembler-like syntax, with an key-word (which may be
 abbreviated), an equal-sign, and  the desired operand value.  If
 duplicate key-words are found, the LAST occurance is used.

 Ccc=xx
  indicates a CarraigeControlCharacter to be prefaced to each record
  read from the primary input stream. Exactly the same effect would
  occur by preceding the PGHDR stage with a "|SPEC Xxx 1 1-* N" stage.
  For example, to create double-spaced output from input which does not
  contain a CCC, specify either "Ccc=F0" or precede the PGHDR stage with
  a "|SPEC XF0 1 1-* N" stage.

 Infile=fn.ft.fm
```

will modify the DEFAULT page-header by adding a one-line file descriptor. This operand causes a LISTFILE to be done, but NO FILE-IO is performed. Syntax requires that no blanks may be used, so the 2 or 3 words may be separated by any keyboard character(s) which is NOT valid as part of a CMS FileName. Examples: INF=PROFILE.EXEC or INFILE=MYFILE/DATA/J. This operand merely allows for quick-and-dirty file-print programs whose output is self-documenting.

LEngth=nn
indicates the number of columns (starting with column 1) to be reserved to contain the Record-Length of that record. The data currently in column-1 will be shifted right by this amount. The default is "0", indicating that record-lengths will NOT be inserted. The record-length will be right-justifed within this area, so inappropriate values could cause truncation (but no error).

Lpp=nn
indicates the maximum number of printable lines-per-page. The default is either the current LPP value for OOE, or "58". This is the TOTAL number of header AND detail lines printed per page, NOT just the number of detail lines. Page-saturation is then calculated, not by merely counting input and header lines, but by taking into account the CCCs of all header lines and of preceding detail lines. Note that in CP's "SPOOL E LPT nnn" command, "nnn" must be 30 thru 255, or "OFF". This program, however, will accept ANY numeric value for LPP.

Pgstart=nn
indicates the page-number which should appear on the first output page. The default is "1". This operand is significant only if a "~" (X'A1') occurs somewhere within the header-records.

SQLn=nn
indicates the number of columns (starting with column 1, or following the LENGTH-field if specified) to be reserved to contain the sequence-number of that record. The data currently in column-1 will be shifted right by this amount. The default is "0", indicating that sequence-numbers will NOT be inserted. The sequence-number will be right-justifed within this area, so inappropriate values could cause truncation (but no error).

SQStart=nnn
indicates the sequence-number to appear on column-1 of the first output record. The default value is "1".

sep header_line_n
After all keyword operands, any data which begins with a special character is assumed to be the start of one or more page-headers, and all subsequent data is considered part of the page-headers. The separator may be any non-alphanumeric which does not otherwise appear in any header-line. Each subsequent occurance of this separator defines the start of another header line, and every occurance must be IMMEDIATELY followed by a CCC.

DEBUG
is a debugging tool which will prefix 3 numbers to each record read from the primary input stream --- the record's LRECL, the record's sequence-number, and the line-number on which this line will be printed on the logical page (taking into account the CCCs of preceding detail-lines and header-lines). This allows for verification of the

```
  page layout, without physically printing the data.

¢|Streams Used¢%

 PRIMARY INPUT STREAM: PGHDR reads records from its primary input stream.
  Each record must begin with a CCC (or "Ccc=xx" must be specified).

 SECONDARY INPUT STREAM: If defined, PGHDR will treat these as page-header
  lines. Any number of records may be defined, and each record must
  begin with a CCC. See later info on the contents of header-lines.
  Data from this stream overrides possible header-lines specified on the
  PGHDR command.

 PRIMARY OUTPUT STREAM: PGHDR constructs pages of output, where each page
  contains the specified header-lines, followed by the appropriate
  number of records read from the primary input stream.  If an
  EndOfReport line has been defined, then it is the last record writen
  to this stream, and any "~" in it is replaced by the record-count (the
  number of records read from the primary input stream).  This output
  stream is typically passed to a URO stage, or a FILEFAST stage to
  create a LISTING file.

 SECONDARY OUTPUT STREAM: If this optional output stream is connected, a
  single record will be written to it.  It will contain "PGSTART=nnn
  SQLSTART=nnn", a value which may be used on a PGHDR stage of a
  sebsequent PIPE, to allow output of that PIPE to continue with the
  next higher page and sequence numbers.

.CS 3 OFF
.CS 5 ON
¢|Usage Notes¢%

   1.   PGHDR delays records to the extent needed to fill a page.

   2.   If the PGHDR stage discovers that its primary output stream is
        not connected, the PGHDR stage ends.

   3.   Page-header lines (whether defined via a secondary input stream,
        or as operands on the PGHDR stage) have these characteristics:
         + Each line must begin with a CCC.  The first record usually
           begins with a "1" (an ANSI "Write-After-Skip-To-1"),
           but this is not required.
         + One or more occurances of "~" (hex-A1) may be used to indicate
           where the updated page-number is to be placed --
           * a SINGLE "~" is replaced by the full, left-justified,
               variable-length, page-number;
           * MULTIPLE "~" are EXACTLY replaced by the right-justified
               (with blanks, not zeroes) page-number.
             ("Page ~~~" creates a pg-number always 3 characters long;
             ("Page ~"   creates a pg-number whose length varies)
         + If no headers are specified, then a default 3|4-line header
           is created (ref. "INFILE=xxx" parameter).
         + End-of-Report line (lines printed only after all detail
           lines have been printed) may be defined by HdrLines, but
           with the first character being a "." (period), followed by
           the desired CCC and line-data.  Any "~" will be replaced by
           the record-count (number of records from primary input
           stream).  This allows (sort of) emulation of some report-
           writers which append a "NNN records selected" output line.
```

4. CCC refers to printer CarriageControlCharacters. These are 1-byte values to control data placement on the page. There are two general types: ANSI and CCW-OP-CODE. Altho PGHDR handles both types, mixing types will cause garbled output. (Refer to the ASATOMC and MCTOASA stages to homogenize CCCs.) Since the demise of the 1403 with its awesome carriage-tape, the most common CCC values are those that cause spacing by number of lines (rather than to specific channel-numbers).

#	ANSI (print AFTER move)	CCW (print BEFORE move)
0	x'4E' "+"	x'01' (x'03' for move-only)
1	x'40' " "	x'09' (x'0B' for move-only)
2	x'F0' "0"	x'11' (x'13' for move-only)
3	x'60' "-"	x'19' (x'1B' for move-only)
Ch01	x'F1' "1"	x'89' (x'8B' for move-only)
Ch08	x'F8' "8"	x'C1' (x'C3' for move-only)
Ch10	x'C1' "A"	x'D1' (x'D3' for move-only)
Ch12	x'C3' "C"	x'E1' (x'E3' for move-only)

Except for the x'C1' ("A") and x'C3' ("C") characters, all characters are unique between the two sets.

```
.CS 5 OFF
.CS 5 ON
¢|Examples¢%

¢|Example 1¢%

     'PIPE  <      TEST FILE'                                ,
        '|  SPEC   PAD 40 1-* 2'                            , /* CCC X40 */
        '|  PGHDR  LPP=20  ;1Header 1 Page ~;-Header 2;     ' ,
        '|  >      TEST LISTING A'
```

The preceding example will read a file and create a LISTING file with 20 lines per page. Note that the header contains 3 lines delimited with semi-colons and using ANSI CCC's. These 3 lines will occupy lines 1 thru 5 on the printed output; lines 6 thru 20 will contain 15 single-spaced detail lines.

```
¢|Example 2¢%

     h.1 = '1Header 1 Page ~' /* Skip-to-1-and-Print (Line-1) */
     h.2 = '-Header 2'        /* Space-3-and-Print   (Line-4) */
     h.3 = '40'x              /* Space-1-and-Print   (Line-5) */
     h.0 = 3                  /* number of header-records     */
     'PIPE        (ENDCHAR ?)'     ,
          '<      TEST FILE'       ,
        '|A:PGHDR  LPP=20 CCC=40'  ,                      /* CCC X40 */
        '|  >      TEST LISTING A' ,
       '?   STEM   H.'             , /* Hdr-recs go to       */
        '|A:'                      /* PGHDR SecInputStream */
```

This example will produce the same results as Example 1, but two different coding techniques are demonstrated --

1. Header lines are defined via stem H.
2. A hex-40 CCC (ANSI write-after-space-1) is inserted via a "CCC=40",

rather then a preceding "SPEC" stage.

¢|Example 3¢%

```
file1  = 'TEST     FILE     A'
file1x = 'INFILE='TRANSLATE(SPACE(file1),,'.')
'PIPE  <'     file1                              ,
   '|  PGHDR  LPP=20 CCC=40 LRECL=4 SEQL=3' file1x ,
   '|  CHOP   133'                                ,
   '|  URO    000E'
Call DIAG  8,'CLOSE 000E NAME' SUBWORD(file1,1,2)
```

The preceding example is a simple file-print program (which was called an "80-80 LIST" in a previous life) which includes a 4-character LRECL and a 3-character SEQUENCE-number; and the default 3-line header will include all info about the file.

¢|Example 4¢%

```
file1  = 'TEST     FILE     A'
file2  = 'TEST     FILE  J'
'PIPE        (ENDCHAR ?)'     ,
    '<'     file1             ,
  '|A:PGHDR  LPP=20 CCC=40'  ,               /* CCC X40 */
  '|  >      TEST LISTING A' ,
  '?   STEM   H.'             , /* Hdr-recs go to       */
  '|A:'                       , /* PGHDR InputStream#1  */
'?A:|  VAR    PBVAR1'           /* PGHDR OutputStream#1 */

 ...

'PIPE        (ENDCHAR ?)'     ,
     '<'     file2            ,
   '|A:PGHDR' pbvar1          ,               /* CCC X40 */
   '|  >>     TEST LISTING A' ,
   '|A:VAR    PBVAR2'         , /* PGHDR OutputStream#2 */
   '?   STEM   H.'            , /* Hdr-recs go to       */
   '|A:'                        /* PGHDR InputStream#1  */
```

The preceding example will run two separate PIPEs (maybe because some other code must be executed between them, or because the second PIPE might not always be executed), but create a single output with ever-increasing page numbers.

.CS 5 OFF
`

PGHDR EXEC

```
/*********************************************************************
*** 124-line source for PGHDR EXEC X6 (2000-02-28 13:09:41) follows ...
*********************************************************************/

/*********************************************************************/
/* Sample program to demonstrate/test PGHDR.REXX                     */
/* It reads 2 files with various PGHDR incantations, and creates     */
/* a single LISTING file, then browses that file.                    */
/* A LISTING file is also created in the user's RDR queue.           */
/*********************************************************************/
Address COMMAND
Parse Upper Source . . myfn myft myfm .

filei1  =  myfn     myft     myfm  /* Input file 1*/
filei2  = 'PROFILE  EXEC     *'    /* Input file 2*/
filei1x = TRANSLATE(SPACE(filei1),'.',' ')
filei2x = TRANSLATE(SPACE(filei2),'.',' ')
fileo   = myfn 'SAMPOUT A'         /* Output File */
lpp     = 'LPP=50'                 /* Lines per Page */
h.      = '40'x
h.1     = '1Test Header 1'  ,
           LEFT(DATE('W'),3)',' ,
           TRANSLATE('CcYy/Mm/Dd','-'DATE('S'),'/CcYyMmDd')    ,
           '('TRANSLATE(  'Yy/123'  ,'/'DATE('J'),'/Yy123'   )')' ,
           'Page ~~~~'
h.2     = '0Test Header 2 (on line#3)'
h.3     = ' '  /* One blank-line between last header and first detail */
h.4     = '.-Report Footer *** Processed ~ records'
h.5     = '. This is line-2 of Report-Footer-Records'
h.0     = 5
pghdr1  = 'PGHDR IN='filei1x lpp
pghdr2  = 'PGHDR IN='filei2x lpp
out     = '>>' fileo
in1     = '<'  filei1
in2     = '<'  filei2
Say 'This is a sample program to demonstrate PGHDR PIPE stage.'
Say 'A file named' fileo 'will be created and then browsed.'
'ERASE' fileo

/*************************************************************\
**   1. Example of simple file-print with default header      **
\*************************************************************/
'PIPE' in2          ,
   '|  PGHDR CC=40' ,
   '|' out

/*************************************************************\
**   2. Same as #1, but default-hdr now contains file-info    **
\*************************************************************/
'PIPE' in2                  ,
   '|  PGHDR  C=40 I='filei2x ,
   '|' out
```

```
/*************************************************************\
**   3. Example of multiple PGHDRs with successive page-nums, **
**      and headers being supplied via stem "H."               **
\*************************************************************/
'PIPE  (END ¢)'               , /* MultiStream PIPE              */
       in1                    , /* read input                    */
   '|A:'pghdr1 'CCC=40'       , /* PGHDR                         */
   '|' out                    , /* write output                  */
'¢     STEM    H.'            , /* Header-records go into        */
   '|A:'                      , /*   PGHDR's secondary input     */
'¢A:|  VAR     PBVARS'          /* SecOutput has PassBack values */

'PIPE' in2                                ,
   '|' myfn   'I=PROFILE/EXEC' pbvars     ,
   '|' out

/*************************************************************\
**   4. Example of insertion of LineNum and Lrecl             **
\*************************************************************/
'PIPE  (END ¢)'               , /* MultiStream PIPE              */
       in1                    , /* read input                    */
   '|A:'pghdr1 'CCC=40 SQLN=4 LEN=2' , /* 4-byte seq#; 2-byte len*/
   '|' out                    , /* write output                  */
'¢     STEM    H.'            , /* Header-records go into        */
   '|A:'                      , /*   PGHDR's secondary input     */
'¢A:|  VAR     PBVARS'          /* SecOutput has PassBack values */

/*************************************************************\
**   5. Example of insertion of debugging data                **
\*************************************************************/
'PIPE  (END ¢)'               , /* MultiStream PIPE              */
       in1                    , /* read input                    */
   '|A:'pghdr1 'debug' pbvars , /* PGHDR                         */
   '|' out                    , /* write output                  */
'¢     STEM    H.'            , /* Header-records go into        */
   '|A:'                      , /*   PGHDR's secondary input     */
'¢A:|  VAR     PBVARS'          /* SecOutput has PassBack values */

/*************************************************************\
**   6. Example of direct printing w/o disturbing OOE         **
**      We do that because either -                           **
**        we don't want to save/restore attributes of OOE, or **
**        virtual OOE doesn't support wider print-lines.      **
\*************************************************************/
prtr   = FREE_CUU('60E')                  /* Gimme an address          */
Call DIAG 8,'DEFINE 4248 AS' prtr'15'x , /*   and a new printer there */
          'SPOOL' prtr 'TO * CLASS Q NAME PGHDR LISTINGq'
'PIPE' in1                , /* read the file      */
    '|'pghdr1 'CC=40'     , /* headerize it       */
    '| ASATOMC'           , /* cvt ASA to CCW     */
    '| CHOP 169'          , /* make it fit a 4248 */
    '| STRIP'             , /* save spool-space   */
    '| URO' prtr           /* to printer         */
Parse Value DIAG(8,'DETACH' prtr)  With . . rdrnum . . . . . recs .
Say 'Created RDR file' rdrnum/1 'with' recs/1 'records.'
Say 'Press ENTER to continue'
Parse Upper Pull rep .
If \ABBREV('NO',rep,1)  Then 'XEDIT' fileo
```

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

FREE_CUU:          Procedure  /* REXXTIP  #98, p 54  &  #277, p 174 ` */
    /* Returns first free address, after requested address             */
    /*    FREE_CUU(190)      ===>  193  <if 190,191,&192 already used */
    /*    FREE_CUU(-190)     ===>  18F  <BACKWARDS search>             */
    /*    FREE_CUU()         ===>  440  <or first avail. FROM 440>     */
  Parse Upper Arg cuu . . . . . . sign
  cuu = WORD(cuu 440,1)
  If \DATATYPE(cuu,'X')  Then Parse Var cuu 1 sign +1 cuu
  Do z = X2D(cuu)   By sign||1   For 255   ,
    While SUBSTR(DIAG(24,D2X(z)),13,1) <> '3'
    End
  Return D2X(z)

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

SPACER REXX

```
/*********************************************************************
**** 89-line source for SPACER REXX X6 (1996-11-21 18:57:44) follows ...
*********************************************************************/

/*****************************SPACER.REXX*****************************/
/* Add a "spacer" record when a specified field changes value         */
/*                                                                    */
/*      SPACER  field-spec  <fill-character>                          */
/*                                                                    */
/* For example --                                                     */
/*      'PIPE COMMAND LISTFILE * * A (DATE NOH'  ,                    */
/*         '| SORT    10.8 1.19'                 ,                   */
/*         '| SPACER  10.3'                      ,                   */
/*         '| CONSOLE'                                                */
/*                                                                    */
/*   will sort LISTFILE output on FileType (10.8), and then add       */
/*   a blank record between lines where there is a change in the      */
/*   first 3 characters of FileType.                                  */
/* The added record will be as long as the right-most column          */
/*   being compared (13, in the preceding example).                   */
/* If the stage had read  "SPACER 10.3 -"  or  "SPACER 10.3 60"       */
/*   then each inserted record has dashes (X'60') in the field.       */
/* "SPACER W2" (in this case) creates same results as "SPACER 10.8".  */
/*                                                                    */
/* 1996/11/15 CMSi/CHM Written by Chuck Meyer                         */
/* 1996/11/21 CMSi/CHM Allow "Wnnn" as operand (word number)          */
/*                                                                    */
/*********************************************************************/
copyright = 'Copyright: Chuck Meyer Systems, Inc.;  1996'
version   = '1996.11.21'

Signal ON ERROR

Parse Upper Source . . fn1 ft1 fm1 fn2 . 1 _source_
Parse Upper Arg c01     fil  .  /* possibly 2 words of input      */
Parse Var c01   c11 '-' c12     /* maybe it's FROM-TO             */
Parse Var c01   c21 '.' c22     /* maybe it's FROM.LENGTH         */
Parse Var c01   'W'     c31     /* maybe it's WORD number         */

?wdnum = DATATYPE(c31,'W')

Select  /* determine the column-range to be compared */
  When DATATYPE(c11,'W') & DATATYPE(c12,'W')  Then ss = c11  (c12 +  1)
  When DATATYPE(c21,'W') & DATATYPE(c22,'W')  Then ss = c21  (c22 +c21)
  When DATATYPE(c01,'W')                      Then ss = c01  (c01 +  9)
  When ?wdnum                                 Then ss = c31  '0'
  Otherwise                                        ss = 1    (10  +  1)
  End
Parse Var ss  c1 c2 .

Select  /* determine what the fill-record should look like */
  When (LENGTH(fil)=2)   & DATATYPE(fil,'X')  Then fx = X2C( fil)
  Otherwise                                        fx = LEFT(fil,1,' ')
  End
num_added  = 0
num_input  = 0
```

```
'PEEKTO RECORD'

Do num_input=1
  If ?wdnum
    Then field = WORD(  record,c1)
    Else field = SUBSTR(record,c1,c2-c1)
  If (num_input=1)  Then Do
    field_sav  = field
    x          = COPIES(fx,LENGTH(field))
    If ?wdnum
      Then spacer_rec = COPIES(' ',WORDINDEX(record,c1)-1) || x
      Else spacer_rec = COPIES(' ',c1-1)                   || x
    Drop x
    End
  If \(field_sav = field)  Then Do
       field_sav = field
       num_added = num_added + 1
       'OUTPUT' spacer_rec      /* write out the SPACER record */
       End
  'OUTPUT' record   /* write out the record                    */
  'READTO'          /* consume the previously-PEEKTO-ed record */
  'PEEKTO  RECORD'  /* and then expose the next record         */
  End

ERROR:
  rcx = rc*(rc\=12)
        /* If called from a REXX prog, pass-back some numbers */
  Signal OFF ERROR
  Address COMMAND 'PIPE (SEP   ? )'  ,
         'COMMAND  PIPE (NOMSG 15)'  ,
         'LITERAL' num_input num_added '| VAR' fn1 '1 | HOLE'
  If \(rc=0)  Then Say 'PIPE stage' fn1 'read' num_input 'records,' ,
              'inserted' num_added 'records ('num_input+num_added rc')'
  Exit rcx

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

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

Contributing Articles and Code to *VM Update*

We welcome your inquiries and manuscripts. Please send them to vu-ed@sdsusa.com

VM Update magazine has changed hands. Its new publisher, Software Diversified Services, strives to continue *VM Update*'s service to the VM community. To that end, SDS invites, welcomes, and encourages your contributions to *VM Update*.

The scope of the magazine has not changed. *VM Update* welcomes your description and code of use to technicians, programmers, and managers running VM systems. Your articles might share modifications, fixes, utilities, short-cuts, hints, guidelines, evaluations, and advice.

Authoring needn't be difficult. The typical article copies code the author wrote for some other purpose and adds a few pages of explanatory text that *VM Update*'s editor will happily polish for you.

In return for the license to publish your article at its website, *VM Update* will pay you $125 per article. We understand that is less than Xephon used to pay. But remember we now distribute *VM Update* free of charge.

Along with that payment you also get worldwide attention via the Web, you get a contribution to your reputation and your employer's reputation, and you get to help your colleagues in the business of running VM. Last year, *VM Update* printed on paper went to about 150 VM professionals each month. By publishing free of charge on the Web, we expect to bring you more readers than that.

So please join SDS in the care and feeding of *VM Update*, a valuable resource in the worldwide community of VM professionals.

Please send your ideas and manuscripts to vu-ed@sdsusa.com. Further details are available at www.sdsusa.com/vmupdate/vutoauthors.htm. Thank you.