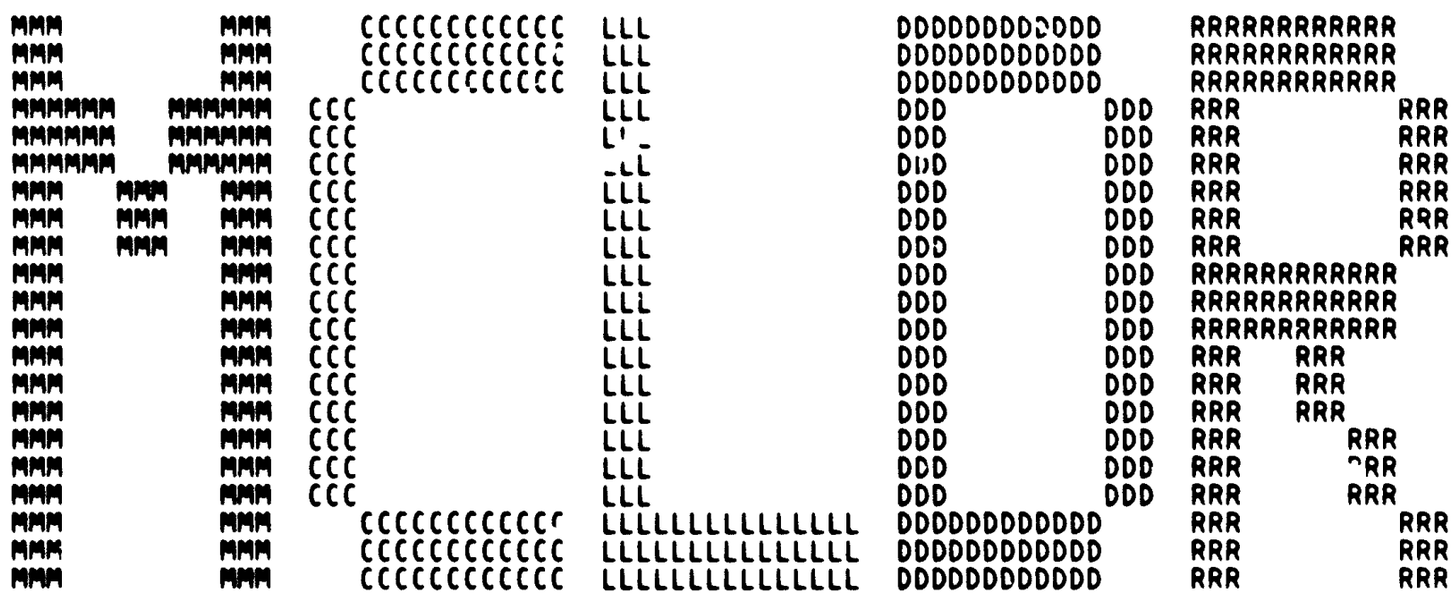

Val

000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF

FILEID**XFLOADER

```
XX      XX  FFFFFFFFFF  LL            000000      AAAAAA    DDDDDDDD    EEEEEEEEEE  RRRRRRRR
XX      XX  FFFFFFFFFF  LL            000000      AAAAAA    DDDDDDDD    EEEEEEEEEE  RRRRRRRR
XX      XX  FF          LL          00      00  AA      AA  DD      DD  EE          RR      RR
XX      XX  FF          LL          00      00  AA      AA  DD      DD  EE          RR      RR
  XX  XX    FF          LL          00      00  AA      AA  DD      DD  EE          RR      RR
  XX  XX    FF          LL          00      00  AA      AA  DD      DD  EE          RR      RR
    XX      FFFFFFFF    LL          00      00  AA      AA  DD      DD  EEEEEEEE    RRRRRRRR
    XX      FFFFFFFF    LL          00      00  AA      AA  DD      DD  EEEEEEEE    RRRRRRRR
  XX  XX    FF          LL          00      00  AAAAAAAAAA  DD      DD  EE          RR  RR
  XX  XX    FF          LL          00      00  AAAAAAAAAA  DD      DD  EE          RR  RR
XX      XX  FF          LL          00      00  AA      AA  DD      DD  EE          RR    RR        ....
XX      XX  FF          LL          00      00  AA      AA  DD      DD  EE          RR    RR        ....
XX      XX  FF          LLLLLLLLLL    000000    AA      AA  DDDDDDDD    EEEEEEEEEE  RR      RR      ....
XX      XX  FF          LLLLLLLLLL    000000    AA      AA  DDDDDDDD    EEEEEEEEEE  RR      RR      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          .TITLE  XFLOADER
0000     2          .IDENT  'V04-000'
0000     3
0000     4  ;
0000     5  ;*****************************************************************************
0000     6  ;*                                                                           *
0000     7  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9  ;*  ALL RIGHTS RESERVED.                                                     *
0000    10  ;*                                                                           *
0000    11  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    12  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    13  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    14  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    15  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    16  ;*  TRANSFERRED.                                                             *
0000    17  ;*                                                                           *
0000    18  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    19  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    20  ;*  CORPORATION.                                                             *
0000    21  ;*                                                                           *
0000    22  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    23  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24  ;*                                                                           *
0000    25  ;*                                                                           *
0000    26  ;*****************************************************************************
0000    27  ;
0000    28
0000    29  ;++
0000    30  ; FACILITY:     DR32 UTILITY PROGRAMS
0000    31  ;
0000    32  ; ABSTRACT:
0000    33  ;       THIS PROGRAM IS THE DR32 MICROCODE LOADER.
0000    34  ;
0000    35  ; ENVIRONMENT:  USER MODE
0000    36  ;
0000    37  ; AUTHOR:  STEVE BECKHARDT,     CREATION DATE:  6-APR-1979
0000    38  ;
0000    39  ; MODIFIED BY:
0000    40  ;
0000    41  ;       V03-001 KTA0102         Kerbey T. Altmann       18-Jun-1982
0000    42  ;               Add a register to save mask.
0000    43  ;
0000    44  ;       V02-003 KTA0014         Kerbey T. Altmann               15-Apr-1981
0000    45  ;               Changed some code to accommodate changed IOC$FFCHAN.
0000    46  ;
0000    47  ;       V02-003 SRB0006         Steve Beckhardt         23-Sep-1980
0000    48  ;               Modified the microcode loader to support the DR750.
0000    49  ;
0000    50  ;--
```

```
                     52          .SBTTL  DECLARATIONS
                     53 ;
                     54 ; INCLUDE FILES:
                     55 ;
                     56
                     57 ;
                     58 ; MACROS:
                     59 ;
                     60
                     61          $CCBDEF                     ; Define CCB offsets
                     62          $CLIDEF                     ; Define CLI values
                     63          $CLIMSGDEF                  ; Define CLI message values
                     64          $DCDEF                      ; Define device classes and types
                     65          $DIBDEF                     ; Define device info. block offsets
                     66          $UCBDEF                     ; Define UCB offsets
                     67
                     68 ;
                     69 ; EQUATED SYMBOLS:
                     70 ;
                     71
00000400             72 WCSSIZE = 1024                      ; NUMBER OF WCS WORDS
                     73
200003FF             74 WCS_PADL_780 = ^X200003FF           ; WCS WORD TO USE FOR PADDING ON DR780
00000000             75 WCS_PADH_780 = 00
                     76
00000000             77 WCS_PADL_750 = 00000000             ; WCS WORD TO USE FOR PADDING ON DR750
00000001             78 WCS_PADH_750 = 01
                     79
                     80 ;
                     81 ; OWN STORAGE:
                     82 ;
                     83
00000000             84          .PSECT  XFDATA,LONG
                     85
                     86
                     87 WCSFAB: $FAB     DNM = <.ULD>,-          ; FAB for reading in WCS
                     88                  FAC = GET,-
                     89                  FOP = SQO
                     90
                     91 WCSRAB: $RAB     FAB = WCSFAB,-          ; RAB for reading in WCS
                     92                  RAC = SEQ,-
                     93                  UBF = WCSLINE,-
                     94                  USZ = 133
                     95
                     96 ERRFAB: $FAB     FAC = PUT,-             ; FAB for writing error messages
                     97                  FNM = <SYS$ERROR>,-
                     98                  FOP = <CIF,SQO>,-
                     99                  MRS = 133,-
                    100                  ORG = SEQ,-
                    101                  RAT = CR,-
                    102                  RFM = VAR
                    103
                    104 ERRRAB: $RAB     FAB = ERRFAB,-         ; RAB for writing error messages
                    105                  RAC = SEQ,-
                    106                  ROP = EOF
                    107
                    108
```

```
00000130  0128   109 XFIOSB: .BLKQ   1                          ; I/O status block
          0130   110
000001A4  0130   111 DEVBUF: .BLKB   DIB$K_LENGTH              ; Buffer for device characteristics
00000074  01A4   112 DEVBUFSIZ = .-DEVBUF
          01A4   113
000001A6  01A4   114 XFCHAN: .BLKW   1                          ; Holds channel number
          01A6   115
          01A6   116 GETCMD: $CLIREQDESC     RQTYPE = CLI$K_GETCMD   ; Get command CLI request block
          01C2   117
          01C2   118 WCS_PADL:                                  ; WCS word to use in padding
000001C6  01C2   119         .BLKL   1
          01C6   120 WCS_PADH:
000001C7  01C6   121         .BLKB   1
          01C7   122
          01C7   123 HIBERFLAG:                                 ; Hibernated flag
       00 01C7   124         .BYTE   0
          01C8   125
          01C8   126 WCSLINE:                                   ; Buffer to hold WCS line.  Also used
0000024E  01C8   127         .BLKB   134                        ; to hold message returned by $GETMSG.
00000086  024E   128 WCSLINESIZ = .-WCSLINE
          024E   129
          024E   130 WCSBFR: .REPT   WCSSIZE                    ; Buffer to hold actual WCS image
          024E   131         .BLKB   5                          ; 1 WCS word = 5 bytes
00000253  024E   132         .ENDR
00001400  164E   133 WCSBFRSIZ = .-WCSBFR
          164E   134
    46 58 164E   135 XFNAME: .ASCII  'XF'                       ; Device name
          1650   136 XFCTRLR:
       20 1650   137         .ASCII  ' '                        ; Space for controller designator
       30 1651   138         .ASCII  '0'                        ; Unit number
00000004  1652   139 XFNAMESIZ = .-XFNAME
          1652   140
          1652   141 WCSFILNAM:
       46 58 1652   142         .ASCII  'XF'                   ; WCS file name
          1654   143 WCS_FNM_CTRLR:
       20 1654   144         .ASCII  ' '                        ; Space for controller designator
 53 43 57 24 1655   145         .ASCII  '$WCS'                 ; Rest of name
00000007  1659   146 WCSFILNAMSIZ = .-WCSFILNAM
          1659   147
          1659   148 WCSDFLTNAM:                               ; Default WCS file name
58 3A 4D 45 54 53 59 53 24 53 59 53
          46 1665   149         .ASCII  'SYS$SYSTEM:XF'
       30 38 37 1666   150 CPUNUM: .ASCII  '780'                ;  CPU number
 44 4C 55 2E 1669   151         .ASCII  '.ULD'
00000014  166D   152 WCSDFLTNAMSIZ = .-WCSDFLTNAM
0000166E  166D   153 NUMDRS: .BLKB   1                         ; Number of DR32s loaded
```

```
                                166E    155          .SBTTL  Read only data
                                166E    156
                             00000000   157          .PSECT  XFCODE,LONG,NOWRT
                                0000    158
                                0000    159 XFNAMEDSC:                                  ; XF Name descriptor
                    00000004    0000    160          .LONG   XFNAMESIZ
                    0000164E'   0004    161          .LONG   XFNAME
                                0008    162
                                0008    163 FMTERRDSC:                                  ; Format error descriptor
                    0000001D'   0008    164          .LONG   FMTERRSIZ
                    00000028'   000C    165          .LONG   FMTERR
                                0010    166
                                0010    167 NODRSERRDSC:                                ; No DR32s error descriptor
                    00000010'   0010    168          .LONG   NODRSERRSIZ
                    00000045'   0014    169          .LONG   NODRSERR
                                0018    170
                                0018    171 WCSLINEDSC:                                 ; WCS line descriptor
                    00000086    0018    172          .LONG   WCSLINESIZ
                    000001C8'   001C    173          .LONG   WCSLINE
                                0020    174
                                0020    175 DEVBUFDSC:                                  ; Device char. buffer descriptor
                    00000074    0020    176          .LONG   DEVBUFSIZ
                    00000130'   0024    177          .LONG   DEVBUF
                                0028    178
6F 66 20 74 63 65 72 72 6F 63 6E 49 0028    179 FMTERR: .ASCII  'Incorrect format in WCS file.'
20 53 43 57 20 6E 69 20 74 61 6D 72 0034
                       2E 65 6C 69 66 0040
                             0000001D  0045    180 FMTERRSIZ = .-FMTERR
                                0045    181
                                0045    182 NODRSERR:
61 6F 6C 20 73 32 33 52 44 20 6F 4E 0045    183          .ASCII  'No DR32s loaded.'
                             2E 64 65 64 0051
                                00000010  0055    184 NODRSERRSIZ = .-NODRSERR
                                0055    185
                                0055    186 HEXDIGITBL:                                 ; Hexadecimal digit table
34 35 36 37 38 39 41 42 43 44 45 46 0055    187          .ASCII  'FEDCBA9876543210'
                             30 31 32 33 0061
                                0065    188
                                0065    189
                                0065    190          .DEFAULT        DISPLACEMENT,WORD
```

```
                     0065   192          .SBTTL  XFLOADER - MAIN PROGRAM
                     0065   193 ;++
                     0065   194 ; FUNCTIONAL DESCRIPTION:
                     0065   195 ;
                     0065   196 ;       THIS IS THE MAIN ENTRY POINT FOR XFLOADER.  IT TRIES TO LOAD
                     0065   197 ;       MICROCODE INTO EACH POSSIBLE DR32 (UP TO 16).
                     0065   198 ;       AFTER LOADING ALL THE DR32s, THIS ROUTINE DETERMINES IF IT
                     0065   199 ;       WAS CALLED BY PROCSTRT.  IF SO, IT WAS RUN WITH A RUN/UIC=[uic]
                     0065   200 ;       COMMAND AS A SEPARATE PROCESS AND SHOULD HIBERNATE.  OTHERWISE,
                     0065   201 ;       IT WAS RUN WITH AN ORDINARY RUN COMMAND AND SHOULD RETURN.
                     0065   202 ;
                     0065   203 ; CALLING SEQUENCE:
                     0065   204 ;
                     0065   205 ;       CALL    XFLOADER
                     0065   206 ;
                     0065   207 ; INPUT PARAMETERS:
                     0065   208 ;
                     0065   209 ;       NONE
                     0065   210 ;
                     0065   211 ; IMPLICIT INPUTS:
                     0065   212 ;
                     0065   213 ;       NONE
                     0065   214 ;
                     0065   215 ; OUTPUT PARAMETERS:
                     0065   216 ;
                     0065   217 ;       NONE
                     0065   218 ;
                     0065   219 ; IMPLICIT OUTPUTS:
                     0065   220 ;
                     0065   221 ;       NONE
                     0065   222 ;
                     0065   223 ; COMPLETION CODES:
                     0065   224 ;
                     0065   225 ;       NONE
                     0065   226 ;
                     0065   227 ; SIDE EFFECTS:
                     0065   228 ;
                     0065   229 ;       NONE
                     0065   230 ;
                     0065   231 ;--
                     0065   232
                     0070   0065   233          .ENTRY  XFLOADER,^M<R4,R5,R6>
                            0067   234
                            0067   235          $SETPRA_S       XFLOADER        ; Specify power recovery AST
                            0073   236
   166D'CF    94            0073   237          CLRB    NUMDRS                  ; Clear number of DR32s loaded
                            0077   238
55   41 8F    9A            0077   239          MOVZBL  #^A/A/,R5               ; First controller letter
   56   10    D0            007B   240          MOVL    #16,R6                  ; Number of controllers
                            007E   241
                            007E   242 10$:     ; Try to load microcode into next DR32
                            007E   243
     4A    10               007E   244          BSBB    LOAD_WCS
   43 50    E9               0080   245          BLBC    R0,80$                  ; If LBC, exit
      55    D6               0083   246          INCL    R5                      ; Next controller letter
   F6 56    F5               0085   247          SOBGTR  R6,10$                  ; Do next one
                            0088   248
```

```
                166D'CF   95   0088   249          TSTB    NUMDRS                          ; Test number of DR32s loaded
                     0C   12   008C   250          BNEQ    40$                             ; There is at least one
                     50   D4   008E   251          CLRL    R0                              ; No DR32s - Give error message
           51    FF7C CF   7E   0090   252          MOVAQ   NODRSERRDSC,R1
                  0153   30   0095   253          BSBW    OUTPUT_ERRMSG
                     2C   11   0098   254          BRB     80$                             ; Exit
                          009A   255
                          009A   256 40$:    ; Determine if we were called from PROCSTRT (as a process
                          009A   257         ; without a CLI) or from a CLI or as a power recovery AST.
                          009A   258         ; If the first case, then hibernate waiting for a power recovery
                          009A   259         ; AST.  Otherwise, return.
                          009A   260
                01C7'CF   95   009A   261          TSTB    HIBERFLAG                       ; Have we hibernated before?
                     26   12   009E   262          BNEQ    80$                             ; Yes, this must be an AST - return
                06   6C   D1   00A0   263          CMPL    (AP),#6                         ; Were we called with 6 arguments?
                     21   12   00A3   264          BNEQ    80$                             ; No, return
                  08 AC   D5   00A5   265          TSTL    CLI$A_UTILSERV(AP)              ; Is there a CLI callback routine?
                     1C   13   00A8   266          BEQL    80$                             ; No, return
                01A6'CF   9F   00AA   267          PUSHAB  GETCMD                          ; Push address of CLI request block
             08 BC    01   FB   00AE   268          CALLS   #1,@CLI$A_UTILSERV(AP)          ; Call CLI callback routine
     00038822 8F    50   D1   00B2   269          CMPL    R0,#CLI$_INVREQTYP              ; Is it invalid request type?
                     0B   12   00B9   270          BNEQ    80$                             ; No, we were called from a CLI
                01C7'CF   96   00BB   271          INCB    HIBERFLAG                       ; Yes, we were called from PROCSTRT,
                          00BF   272          $HIBER_S                                    ; set flag and hibernate
                          00C6   273
                50    00'  3C   00C6   274 80$:     MOVZWL  S^#SS$_NORMAL,R0
                          04   00C9   275          RET                                     ; and exit
```

```
                        .SBTTL  LOAD_WCS - Load Microcode on a specified DR32
                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       This routine loads microcode into a specified DR32.  It assigns
                ;       a channel to the DR, opens the microcode file, and loads
                ;       the DR.
                ;
                ; CALLING SEQUENCE:
                ;
                ;       BSBW    LOAD_WCS
                ;
                ; INPUT PARAMETERS:
                ;
                ;       R5      Controller letter
                ;
                ; IMPLICIT INPUTS:
                ;
                ;       None
                ;
                ; OUTPUT PARAMETERS:
                ;
                ;       R0      0 = Exit
                ;               1 = Continue
                ;
                ; IMPLICIT OUTPUTS:
                ;
                ;       NUMDRS is incremented if microcode is loaded successfully
                ;
                ; COMPLETION CODES:
                ;
                ;       None
                ;
                ; SIDE EFFECTS:
                ;
                ;       Errors are written to the file SYS$ERROR
                ;--
                ;
                LOAD_WCS:
1654'CF    55   90  00CA        MOVB    R5,WCS_FNM_CTRLR        ; Store controller letter in WCS
                    00CF                                        ; file name
1650'CF    55   90  00CF        MOVB    R5,XFCTRLR              ; Store controller letter in DR name
                    00D4
                    00D4        ; Assign a channel to the DR.  This is done with an internal
                    00D4        ; procedure instead of $ASSIGN_S so that we can assign a channel
                    00D4        ; even if another process has a channel assigned or if the DR
                    00D4        ; is allocated.
                    00D4
    01A4'CF    3F   00D4        PUSHAW  XFCHAN                  ; Address of channel number
    FF24 CF    7F   00D8        PUSHAQ  XFNAMEDSC               ; Address of device name desc.
         02    DD   00DC        PUSHL   #2                      ; Number of arguments
    50   5E    D0   00DE        MOVL    SP,R0                   ; Pointer to argument list
                    00E1        $CMKRNL_S       ASSIGN,(R0)     ; Assign the channel
    5E   0C    C0   00EE        ADDL    #12,SP                  ; Adjust stack
      11 50    E8   00F1        BLBS    R0,10$                  ; Successful assignment
0000'8F  50    B1   00F4        CMPW    R0,#SS$_NOSUCHDEV       ; No such device error?
         04    12   00F9        BNEQ    5$                      ; No, error
```

```
                50   01    D0  00FB    334          MOVL    #1,R0                       ; Yes, continue
                           05  00FE    335          RSB
                               00FF    336
                  00E9     30  00FF    337 5$:      BSBW    OUTPUT_ERRMSG               ; Give error message
                    50     D4  0102    338          CLRL    R0                          ; Exit
                           05  0104    339          RSB
                               0105    340
                               0105    341 10$:     ; Open WCS file by opening a file whose logical name is
                               0105    342          ; XFc$WCS where 'c' is the controller letter.
                               0105    343
        002C'CF   1652'CF      9E  0105    344          MOVAB   WCSFILNAM,WCSFAB+FAB$L_FNA      ; Store file name
             0034'CF    07     90  010C    345          MOVB    #WCSFILNAMSIZ,WCSFAB+FAB$B_FNS  ; Store file name size
                               0111    346          $OPEN   FAB = WCSFAB
                  61 50    E8  011C    347          BLBS    R0,20$                      ; Success
                               011F    348
                               011F    349          ; That didn't work so open a default WCS file.
                               011F    350          ; However, the default file name (as well as the WCS padding word)
                               011F    351          ; is dependent on what type of DR32 we have.  Currently, the
                               011F    352          ; DR780 and DR750 are supported.
                               011F    353
             1667'CF    38     90  011F    354          MOVB    #^A'8',CPUNUM+1             ; Assume DR780 - set number in default
        01C2'CF   200003FF 8F     D0  0124    355          MOVL    #WCS_PADL_780,WCS_PADL  ; file name and also set WCS padding word
             01C6'CF    00     90  012D    356          MOVB    #WCS_PADH_780,WCS_PADH
                               0132    357          $GETCHN_S       PRIBUF = DEVBUFDSC,-    ; Get device info
                               0132    358                          CHAN   = XFCHAN
                  30 50    E9  0148    359          BLBC    R0,15$                      ; Error
                     02    91  014B    360          CMPB    #DT$_DR780,-                ; Check device type to see if we have
                  0135'CF        014D    361                  DIB$B_DEVTYPE+DEVBUF        ; A DR780
                     0F    13  0150    362          BEQL    12$                         ; We have a DR780
             1667'CF    35     90  0152    363          MOVB    #^A'5',CPUNUM+1             ; We have a DR750 - set number in default
             01C2'CF    00     D0  0157    364          MOVL    #WCS_PADL_750,WCS_PADL  ; file name and also set WCS padding word
             01C6'CF    01     90  015C    365          MOVB    #WCS_PADH_750,WCS_PADH
                               0161    366 12$:
        002C'CF   1659'CF      9E  0161    367          MOVAB   WCSDFLTNAM,WCSFAB+FAB$L_FNA     ; Store default file name
             0034'CF    14     90  0168    368          MOVB    #WCSDFLTNAMSIZ,WCSFAB+FAB$B_FNS ; Store default file name size
                               016D    369          $OPEN   FAB = WCSFAB
                  05 50    E8  0178    370          BLBS    R0,20$                      ; Opened successfully
                               017B    371 15$:
                   006D    30  017B    372          BSBW    OUTPUT_ERRMSG               ; Error - output error message
                     5B    11  017E    373          BRB     70$                         ; Deassign channel
                               0180    374
                               0180    375 20$:     $CONNECT        RAB = WCSRAB
                  40 50    E9  018B    376          BLBC    R0,50$                      ; Error
                               018E    377
                               018E    378          ; Now read in (and format) WCS
                               018E    379
             0287'CF    00     FB  018E    380          CALLS   #0,READ_WCS
                  38 50    E9  0193    381          BLBC    R0,50$                      ; Error
                               0196    382
                               0196    383          ; Issue QIO to actually load the microcode
                               0196    384
                               0196    385          $QIOW_S         FUNC = #IO$_LOADMCODE,-
                               0196    386                          CHAN = XFCHAN,-
                               0196    387                          IOSB = XFIOSB,-
                               0196    388                          P1 = WCSBFR,-
                               0196    389                          P2 = #WCSBFRSIZ
                  0E 50    E9  01BD    390          BLBC    R0,50$                      ; Error
```

```
                50   0128'CF   3C   01C0   391          MOVZWL  XFIOSB,R0              ; Get I/O status block
                       06 50   E9   01C5   392          BLBC    R0,50$                 ; Error
                   166D'CF     96   01C8   393          INCB    NUMDRS                 ; Increment # of DRs loaded
                         02    11   01CC   394          BRB     60$
                                    01CE   395
                         1B    10   01CE   396 50$:     BSBB    OUTPUT_ERRMSG          ; Output error message
                                    01D0   397
                                    01D0   398 60$:     $CLOSE  FAB = WCSFAB           ; Close file
                                    01DB   399 70$:     $DASSGN_S       XFCHAN         ; Deassign channel
                                    01E7   400
                50   01    D0   01E7   401          MOVL    #1,R0                  ; Continue
                                05  01EA   402          RSB
```

```
                        01EB   404          .SBTTL  OUTPUT_ERRMSG - Output error message
                        01EB   405 ;++
                        01EB   406 ; FUNCTIONAL DESCRIPTION:
                        01EB   407 ;
                        01EB   408 ;       This routine outputs error messages to SYS$ERROR.
                        01EB   409 ;
                        01EB   410 ; CALLING SEQUENCE:
                        01EB   411 ;
                        01EB   412 ;       BSBW    OUTPUT_ERRMSG
                        01EB   413 ;
                        01EB   414 ; INPUT PARAMETERS:
                        01EB   415 ;
                        01EB   416 ;       R0      0 or VMS completion code
                        01EB   417 ;
                        01EB   418 ;       R1      If R0 = 0, then R1 contains address of error message descriptor
                        01EB   419 ;
                        01EB   420 ; IMPLICIT INPUTS:
                        01EB   421 ;
                        01EB   422 ;       None
                        01EB   423 ;
                        01EB   424 ; OUTPUT PARAMETERS:
                        01EB   425 ;
                        01EB   426 ;       None
                        01EB   427 ;
                        01EB   428 ; IMPLICIT OUTPUTS:
                        01EB   429 ;
                        01EB   430 ;       None
                        01EB   431 ;
                        01EB   432 ; COMPLETION CODES:
                        01EB   433 ;
                        01EB   434 ;       None
                        01EB   435 ;
                        01EB   436 ; SIDE EFFECTS:
                        01EB   437 ;
                        01EB   438 ;       R2 is not preserved
                        01EB   439 ;--
                        01EB   440
                        01EB   441 OUTPUT_ERRMSG:
     52   00E4'CF    9E  01EB   442          MOVAB   ERRRAB,R2               ; Put address of RAB in R2
                50    D5  01F0   443          TSTL    R0                      ; Have VMS completion status?
                  0B  12  01F2   444          BNEQ    10$                     ; Yes
        22 A2    61    B0  01F4   445          MOVW    (R1),RAB$W_RSZ(R2)      ; No, move size of message into RAB
  28 A2    04 A1    D0  01F8   446          MOVL    4(R1),RAB$L_RBF(R2)     ; Move address of message into RAB
                  1A  11  01FD   447          BRB     20$
                      01FF   448
                      01FF   449 10$:     ; Have VMS completion status in R0.  Get corresponding message.
                      01FF   450
                      01FF   451          $GETMSG_S       MSGID = R0,-
                      01FF   452                          MSGLEN = RAB$W_RSZ(R2),-
                      01FF   453                          BUFADR = WCSLINEDSC
28 A2   01C8'CF    9E  0213   454          MOVAB   WCSLINE,RAB$L_RBF(R2)   ; Store address of buffer in RAB
                      0219   455
                      0219   456 20$:     ; Create output file or just open it if it already exists.  Then
                      0219   457          ; output message and close file.
                      0219   458
                      0219   459          $CREATE         FAB = ERRFAB
        1D 50    E9  0224   460          BLBC    R0,60$                  ; Error
```

```
                0227    461
                0227    462             $CONNECT        RAB = (R2)
                0230    463
                0230    464             $PUT            RAB = (R2)
                0239    465
                0239    466             $CLOSE          FAB = ERRFAB
                0244    467
05              0244    468 60$:        RSB
```

```
                     0245   470          .SBTTL  ASSIGN - Assign a channel
                     0245   471 ;++
                     0245   472 ; FUNCTIONAL DESCRIPTION:
                     0245   473 ;
                     0245   474 ;       This routine assigns a channel to a device.  It is functionally
                     0245   475 ;       a subset of the $ASSIGN system service.  It is used here
                     0245   476 ;       instead of the $ASSIGN system service for the following
                     0245   477 ;       reason:  The DR32 is a non-shareable device.  Therefore, if
                     0245   478 ;       a process is using a DR32 and a power failure occurs, a normal
                     0245   479 ;       $ASSIGN would fail since another process would have a channel
                     0245   480 ;       assigned.  In other words, we would be unable to reload microcode
                     0245   481 ;       on DR32s that were in use at the time of a power failure.
                     0245   482 ;       This ASSIGN, on the other hand, will work.  Note that if we
                     0245   483 ;       try to load microcode on a DR32 in the middle of a data
                     0245   484 ;       transfer, the load microcode will fail and the data transfer
                     0245   485 ;       will continue.
                     0245   486 ;
                     0245   487 ; CALLING SEQUENCE:
                     0245   488 ;
                     0245   489 ;       CALLS/G ASSIGN  (in KERNEL mode)
                     0245   490 ;
                     0245   491 ; INPUT PARAMETERS:
                     0245   492 ;
                     0245   493 ;       4(AP)   Address of device name string descriptor
                     0245   494 ;       8(AP)   Address to store assigned channel number
                     0245   495 ;
                     0245   496 ; IMPLICIT INPUTS:
                     0245   497 ;
                     0245   498 ;       None
                     0245   499 ;
                     0245   500 ; OUTPUT PARAMETERS:
                     0245   501 ;
                     0245   502 ;       R0      Completion code
                     0245   503 ;
                     0245   504 ; IMPLICIT OUTPUTS:
                     0245   505 ;
                     0245   506 ;       None
                     0245   507 ;
                     0245   508 ; COMPLETION CODES:
                     0245   509 ;
                     0245   510 ;       SS$_IVDEVNAM    Invalid device name
                     0245   511 ;       SS$_NOIOCHAN    No I/O channel is available for assignment
                     0245   512 ;       SS$_NOSUCHDEV   No such device on this system
                     0245   513 ;
                     0245   514 ; SIDE EFFECTS:
                     0245   515 ;
                     0245   516 ;       None
                     0245   517 ;--
                     0245   518
                0198 0245   519          .ENTRY  ASSIGN,^M<R3,R4,R7,R8>
                     0247   520
54   00000000'GF   D0 0247   521          MOVL    G^SCH$GL_CURPCB,R4      ; Get current PCB
                     024E   522
     00000000'GF   16 024E   523          JSB     G^IOC$FFCHAN            ; Find free I/O channel
           01 50   E8 0254   524          BLBS    R0,10$                  ; Have one
                   04 0257   525          RET                             ; No free I/O channel
                      0258   526
```

```
                57    51    7D   0258    527 10$:    MOVQ    R1,R7                       ; Save channel index and CCB address
00000000'GF     16    025B    528         JSB     G^$CH$IOLOCKW               ; Lock I/O database for write access
 51    04 AC    D0    0261    529         MOVL    4(AP),R1                    ; Get address of device name desc.
00000000'GF     16    0265    530         JSB     G^IOC$SEARCHDEV             ; Search for device
       13 50    E9    026B    531         BLBC    R0,20$                      ; Didn't find it
                      026F    532
                      026E    533         ; Do the assignment
                      026E    534
                      026E    535
    68    51    D0    026E    535         MOVL    R1,CCB$L_UCB(R8)            ; Store UCB address in CCB
      5C A1    B6    0271    536         INCW    UCB$W_REFC(R1)              ; Incr. UCB reference count
 09 A8    04    90    0274    537         MOVB    #4,CCB$B_AMOD(R8)           ; Store access mode
 08 BC    57    B0    0278    538         MOVW    R7,@8(AP)                   ; Store assigned channel number
50    0000'8F    3C    027C    539         MOVZWL  #SS$_NORMAL,R0              ; Success status
                      0281    540
00000000'GF     17    0281    541 20$:    JMP     G^IOC$UNLOCK                ; Unlock I/O data base and return
```

```
                     0287   543          .SBTTL  READ_WCS - Read in and Format WCS
                     0287   544 ;++
                     0287   545 ; FUNCTIONAL DESCRIPTION:
                     0287   546 ;
                     0287   547 ;       This routine reads in and formats the WCS.  The input line
                     0287   548 ;       is in the format produced by MICRO2:
                     0287   549 ;
                     0287   550 ;               [addr]=value
                     0287   551 ;
                     0287   552 ;       E.g.    [21D]=30FF12AB00        (Note 40 bit WCS word)
                     0287   553 ;
                     0287   554 ; CALLING SEQUENCE:
                     0287   555 ;
                     0287   556 ;       CALLS/G READ_WCS
                     0287   557 ;
                     0287   558 ; INPUT PARAMETERS:
                     0287   559 ;
                     0287   560 ;       None
                     0287   561 ;
                     0287   562 ; IMPLICIT INPUTS:
                     0287   563 ;
                     0287   564 ;       The WCS is stored in WCSBFR
                     0287   565 ;
                     0287   566 ; OUTPUT PARAMETERS:
                     0287   567 ;
                     0287   568 ;       R0      Completion code or 0.  If 0 then R1 contains descriptor
                     0287   569 ;               to error message
                     0287   570 ;
                     0287   571 ; IMPLICIT OUTPUTS:
                     0287   572 ;
                     0287   573 ;       None
                     0287   574 ;
                     0287   575 ; COMPLETION CODES:
                     0287   576 ;
                     0287   577 ;       The ones returned by $GET
                     0287   578 ;
                     0287   579 ; SIDE EFFECTS:
                     0287   580 ;
                     0287   581 ;       None
                     0287   582 ;--
                     0287   583
                     0287   584 READ_WCS:
                00FC 0287   585          .WORD   ^M<R2,R3,R4,R5,R6,R7>
                     0289   586
    52  024E'CF   9E 0289   587          MOVAB   WCSBFR,R2               ; Address of WCS buffer
          53    D4 028E   588          CLRL    R3                      ; WCS word #
                     0290   589
                     0290   590 10$:     ; Get next line from WCS file
                     0290   591
                     0290   592          $GET    RAB = WCSRAB
00000000'8F    50   D1 029B   593          CMPL    R0,#RMS$_EOF            ; End of file?
          3D   13 02A2   594          BEQL    70$                     ; Yes
        4D 50  E9 02A4   595          BLBC    R0,80$                  ; No, other error
                     02A7   596
                     02A7   597          ; Get size of line read, zero byte after end of line.
                     02A7   598
    55  0072'CF   3C 02A7   599          MOVZWL  WCSRAB+RAB$W_RSZ,R5     ; Get size of line read
```

```
                                02B4    602
                                02B4    603             ; If first character on line is ';' then ignore this line.
                                02B4    604
                        3B  64  91      02B4    605             CMPB    (R4),#^A/;/             ; Is it ';'?
                            D7  13      02B7    606             BEQL    10$                     ; Yes, get next line
                                02B9    607
                                02B9    608             ; If first character on line is not '[' then there is no more
                                02B9    609             ; WCS in file.
                                02B9    610
                    5B 8F   84  91      02B9    611             CMPB    (R4)+,#^A/[/            ; Compare it
                            22  12      02BD    612             BNEQ    70$                     ; It's not '['
                            3C  10      02BF    613             BSBB    CVTADDR                 ; It is.  Convert address (in R5)
                        53  55  D1      02C1    614             CMPL    R5,R3                   ; Compare this address with last+1
                            04  13      02C4    615             BEQL    30$                     ; Have sequential addresses
                            2D  19      02C6    616             BLSS    ERROR                   ; Addresses went backwards
                            72  10      02C8    617             BSBB    FILL                    ; Have a hole - fill it
                                02CA    618
                                02CA    619 30$:        ; Convert the data in two parts:  first 1 byte and then 4 bytes.
                                02CA    620
                        57  02  D0      02CA    621             MOVL    #2,R7                   ; Number of digits to convert
                            56  10      02CD    622             BSBB    CVTDATA                 ; Convert data - returns in R5
                        56  55  D0      02CF    623             MOVL    R5,R6                   ; Save
                        57  08  D0      02D2    624             MOVL    #8,R7                   ; Number of digits to convert
                            4E  10      02D5    625             BSBB    CVTDATA                 ; Returns data in R5
                        82  55  D0      02D7    626             MOVL    R5,(R2)+                ; Store data in buffer
                        82  56  90      02DA    627             MOVB    R6,(R2)+                ; Rest of WCS word
                            53  D6      02DD    628             INCL    R3                      ; Inc. WCS word counter
                            AF  11      02DF    629             BRB     10$                     ; Repeat
                                02E1    630
                                02E1    631 70$:        ; No more data in file
                            53  D5      02E1    632             TSTL    R3                      ; Make sure we got at least 1 word
                            10  13      02E3    633             BEQL    ERROR                   ; We didn't
                55  0400  8F  3C      02E5    634             MOVZWL  #WCSSIZE,R5             ; Number of words of WCS required
                    53  55  D1      02EA    635             CMPL    R5,R3                   ; Have enough?
                            02  13      02ED    636             BEQL    75$                     ; Yes
                            4B  10      02EF    637             BSBB    FILL                    ; No, fill buffer to end
                                02F1    638
                        50  00' D0      02F1    639 75$:        MOVL    S^#SS$_NORMAL,R0        ; Success
                                02F4    640
                            04  02F4    641 80$:        RET
                                02F5    642
                                02F5    643
                                02F5    644
                                02F5    645 ERPOR:      ; Come here for errors involving format of WCS file.
                                02F5    646
                51  FDOF CF  7E      02F5    647             MOVAQ   FMTERRDSC,R1            ; Get address of error msg. descriptor
                            50  D4      02FA    648             CLRL    R0                      ; Indicates descriptor is in R1
                            04  02FC    649             RET
```

```
                     02FD   651          .SBTTL  CVTADDR - Convert WCS address
                     02FD   652 ;++
                     02FD   653 ; FUNCTIONAL DESCRIPTION:
                     02FD   654 ;
                     02FD   655 ;       This routine converts the WCS address to binary.  This routine
                     02FD   656 ;       converts until a non-hex digit is found.  At that point,
                     02FD   657 ;       the next two characters must be ']='.  If they're not, then it's
                     02FD   658 ;       a WCS format error.
                     02FD   659 ;
                     02FD   660 ; CALLING SEQUENCE:
                     02FD   661 ;
                     02FD   662 ;       BSBW    CVTADDRE
                     02FD   663 ;
                     02FD   664 ; INPUT PARAMETERS:
                     02FD   665 ;
                     02FD   666 ;       R4      Address of first byte of WCS address to convert
                     02FD   667 ;
                     02FD   668 ; IMPLICIT INPUTS:
                     02FD   669 ;
                     02FD   670 ;       None
                     02FD   671 ;
                     02FD   672 ; OUTPUT PARAMETERS:
                     02FD   673 ;
                     02FD   674 ;       R5      WCS Address
                     02FD   675 ;
                     02FD   676 ; IMPLICIT OUTPUTS:
                     02FD   677 ;
                     02FD   678 ;       None
                     02FD   679 ;
                     02FD   680 ; COMPLETION CODES:
                     02FD   681 ;
                     02FD   682 ;       None
                     02FD   683 ;
                     02FD   684 ; SIDE EFFECTS:
                     02FD   685 ;
                     02FD   686 ;       Errors are handled by branching to ERROR
                     02FD   687 ;--
                     02FD   688
                     02FD   689 CVTADDR:
                55   D4   02FD   690          CLRL    R5                      ; Will contain address
FD50 CF    10   84   3A   02FF   691 10$:     LOCC    (R4)+,#16,HEXDIGITBL    ; Locate char. in hex digit table
                0B   13   0305   692          BEQL    20$                     ; Not a hex digit
                50   D7   0307   693          DECL    R0                      ; Subtract one to get true value
          55    55   04   78   0309   694          ASHL    #4,R5,R5                ; Multiply address so far by 16
                55   50   C0   030D   695          ADDL    R0,R5                   ; Add next digit
                ED   11   0310   696          BRB     10$                     ; Repeat
                          0312   697
                          0312   698 20$:     ; Have a non-hex digit.  Make sure it and  next char. are ']='.
                          0312   699
                54   D7   0312   700          DECL    R4                      ; Back up 1 byte
          3D5D  8F   84   B1   0314   701          CMPW    (R4)+,#^A/]=/           ; Right characters?
                DA   12   0319   702          BNEQ    ERROR                   ; No
                          031B   703
                          031B   704          ; Make sure address is within size of WCS.
                          031B   705
 00000400 8F    55   D1   031B   706          CMPL    R5,#WCSSIZE
                D1   18   0322   707          BGEQ    ERROR                   ; Too large
```

CVTADDR - Convert WCS address

05 0324 708 RSB

```
                                0325   710          .SBTTL  CVTDATA - Convert WCS Data
                                0325   711 ;++
                                0325   712 ; FUNCTIONAL DESCRIPTION:
                                0325   713 ;
                                0325   714 ;       This routine converts a specified number of hexadecimal bytes
                                0325   715 ;       to binary.
                                0325   716 ;
                                0325   717 ; CALLING SEQUENCE:
                                0325   718 ;
                                0325   719 ;       BSBW    CVTDATA
                                0325   720 ;
                                0325   721 ; INPUT PARAMETERS:
                                0325   722 ;
                                0325   723 ;       R4      Address of string to convert
                                0325   724 ;       R7      Number of bytes to convert
                                0325   725 ;
                                0325   726 ; IMPLICIT INPUTS:
                                0325   727 ;
                                0325   728 ;       None
                                0325   729 ;
                                0325   730 ; OUTPUT PARAMETERS:
                                0325   731 ;
                                0325   732 ;       R5      Converted WCS data
                                0325   733 ;
                                0325   734 ; IMPLICIT OUTPUTS:
                                0325   735 ;
                                0325   736 ;       None
                                0325   737 ;
                                0325   738 ; COMPLETION CODES:
                                0325   739 ;
                                0325   740 ;       None
                                0325   741 ;
                                0325   742 ; SIDE EFFECTS:
                                0325   743 ;
                                0325   744 ;       Errors are handled by branching to ERROR.
                                0325   745 ;--
                                0325   746
                                0325   747 CVTDATA:
                          55    D4    0325   748          CLRL    R5                      ; Will hold result
FD28 CF    10    84    3A    0327   749 10$:     LOCC    (R4)+,#16,HEXDIGITBL    ; Locate char. in hex digit table
                    C6    13    032D   750          BEQL    ERROR                   ; Not a hex digit.
                    50    D7    032F   751          DECL    R0                      ; Subtract one to get true value
      55    55    04    78    0331   752          ASHL    #4,R5,R5                ; Multiply number so far by 16
            55    50    C0    0335   753          ADDL    R0,R5                   ; Add in nect character
              EC  57    F5    0338   754          SOBGTR  R7,10$                  ; Repeat
                    05    033B   755          RSB
```

```
                        .SBTTL  FILL - Fill Holes in Buffer
                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       This routine is called to fill holes in the WCS image with
                ;       a default WCS word.  This is necessary because the addresses
                ;       in the WCS file are not necessarily sequential.
                ;
                ; CALLING SEQUENCE:
                ;
                ;       BSBW    FILL
                ;
                ; INPUT ARGUMENTS:
                ;
                ;       R2      Address of next location in WCS buffer
                ;       R3      Current WCS address
                ;       R5      WCS address to fill to
                ;
                ; IMPLICIT INPUTS:
                ;
                ;       None
                ;
                ; OUTPUT ARGUMENTS:
                ;
                ;       None
                ;
                ; IMPLICIT OUTPUTS:
                ;
                ;       None
                ;
                ; COMPLETION CODES:
                ;
                ;       None
                ;
                ; SIDE EFFECTS:
                ;
                ;       None
                ;--
                ;
82   01C2'CF    D0   033C    796 FILL.   MOVL    WCS_PADL,(R2)+          ; Store low 4 bytes
82   01C6'CF    90   0341    797         MOVB    WCS_PADH,(R2)+          ; Store high byte
  F2 53    55   F2   0346    798         AOBLSS  R5,R3,FILL              ; Repeat
                  05   034A    799         RSB
                       034B    800
                       034B    801
                       034B    802
                       034B    803         .END    XFLOADER
```

```
$$.TAB                  =  000000E4 R      02
$$.TABEND               =  00000128 R      02
$$.TMP                  =  00000100
$$.TMP1                 =  00000001
$$.TMP2                 =  000000CF
$$.TMPX                 =  00000004 R      03
$$.TMPX1                =  00000009
$$T1                    =  00000001
$CLI.                   =  000001A6 R      02
$CLI..                  =  000001C2 R      02
ASSIGN                     00000245 RG     04
CCB$B_AMOD              =  00000009
CCB$L_UCB               =  00000000
CLI$A_UTILSERV          =  00000008
CLI$B_RQTYPE            =  00000000
CLI$C_REQDESC           =  0000001C
CLI$K_GETCMD            =  00000001
CLI$_INVREQTYP          =  00038822
CPUNUM                     00001666 R      02
CVTADDR                    000002FD R      04
CVTDATA                    00000325 R      04
DEVBUF                     00000130 R      02
DEVBUFDSC                  00000020 R      04
DEVBUFSIZ               =  00000074
DIB$B_DEVTYPE           =  00000005
DIB$K_LENGTH            =  00000074
DT$_DR780               =  00000002
ERRFAB                     00000094 R      02
ERROR                      000002F5 R      04
ERRRAB                     000000E4 R      02
FAB$B_DNS               =  00000035
FAB$B_FNS               =  00000034
FAB$C_BID               =  00000003
FAB$C_BLN               =  00000050
FAB$C_SEQ               =  00000000
FAB$C_VAR               =  00000002
FAB$L_ALQ               =  00000010
FAB$L_DNA               =  00000030
FAB$L_FNA               =  0000002C
FAB$L_FOP               =  00000004
FAB$V_CHAN_MODE         =  00000002
FAB$V_CIF               =  00000019
FAB$V_CR                =  00000001
FAB$V_FILE_MODE         =  00000004
FAB$V_GET               =  00000001
FAB$V_LNM_MODE          =  00000000
FAB$V_PUT               =  00000000
FAB$V_SQO               =  00000006
FAB$W_GBC               =  00000048
FILL                       0000033C R      04
FMTERR                     00000028 R      04
FMTERRDSC                  00000008 R      04
FMTERRSIZ               =  0000001D
GETCMD                     000001A6 R      02
HEXDIGITBL                 00000055 R      04
HIBERFLAG                  000001C7 R      02
IO$_LOADMCODE              ******** X      04
IOC$FFCHAN                 ******** X      04
IOC$SEARCHDEV              ******** X      04
IOC$UNLOCK                 ******** X      04
LOAD_WCS                   000000CA R      04
NODRSERR                   00000045 R      04
NODRSERRDSC                00000010 R      04
NODRSERRSIZ             =  00000010
NUMDRS                     0000166D R      02
OUTPUT_ERRMSG              000001EB R      04
RAB$B_RAC               =  0000001E
RAB$C_BID               =  00000001
RAB$C_BLN               =  00000044
RAB$C_SEQ               =  00000000
RAB$L_CTX               =  00000018
RAB$L_RBF               =  00000028
RAB$L_ROP               =  00000004
RAB$V_EOF               =  00000008
RAB$W_RSZ               =  00000022
READ_WCS                   00000287 R      04
RMS$_EOF                   ******** X      04
SCH$GL_CURPCB              ******** X      04
SCH$IOLOCKW                ******** X      04
SS$_NORMAL                 ******** X      04
SS$_NOSUCHDEV              ******** X      04
SYS$CLOSE                  ******** GX     04
SYS$CMKRNL                 ******** GX     04
SYS$CONNECT                ******** GX     04
SYS$CREATE                 ******** GX     04
SYS$DASSGN                 ******** GX     04
SYS$GET                    ******** GX     04
SYS$GETCHN                 ******** GX     04
SYS$GETMSG                 ******** GX     04
SYS$HIBER                  ******** GX     04
SYS$OPEN                   ******** GX     04
SYS$PUT                    ******** GX     04
SYS$QIOW                   ******** GX     04
SYS$SETPRA                 ******** GX     04
UCB$W_REFC              =  0000005C
WCSBFR                     0000024E R      02
WCSBFRSIZ               =  00001400
WCSDFLTNAM                 00001659 R      02
WCSDFLTNAMSIZ           =  00000014
WCSFAB                     00000000 R      02
WCSFILNAM                  00001652 R      02
WCSFILNAMSIZ            =  00000007
WCSLINE                    000001C8 R      02
WCSLINEDSC                 00000018 R      04
WCSLINESIZ              =  00000086
WCSRAB                     00000050 R      02
WCSSIZE                 =  00000400
WCS_FNM_CTRLR              00001654 R      02
WCS_PADR                   000001C6 R      02
WCS_PADH_750            =  00000001
WCS_PADH_780            =  00000000
WCS_PADL                   000001C2 R      02
WCS_PADL_750            =  00000000
WCS_PADL_780            =  200003FF
```

```
XFCHAN                          000001A4 R      02
XFCTRLR                         00001650 R      02
XFIOSB                          00000128 R      02
XFLOADER                        00000065 RG     04
XFNAME                          0000164E R      02
XFNAMEDSC                       00000000 R      04
XFNAMESIZ                     = 00000004
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
XFDATA	0000166E (5742.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
$RMSNAM	0000000D (13.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
XFCODE	0000034B (843.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	16	00:00:00.12	00:00:00.99
Command processing	153	00:00:00.70	00:00:03.30
Pass 1	408	00:00:16.96	00:00:35.40
Symbol table sort	0	00:00:01.91	00:00:03.60
Pass 2	216	00:00:03.96	00:00:08.68
Symbol table output	18	00:00:00.11	00:00:00.31
Psect synopsis output	5	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	816	00:00:23.80	00:00:52.32

```
The working set limit was 900 pages.
112861 bytes (221 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1296 non-local and 24 local symbols
803 source lines were read in Pass 1, producing 35 object records in Pass 2.
46 pages of virtual memory were used to define 39 macros.
```

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYS.OBJ]LIB.MLB;1	2
_255DUA28:[SYSLIB]STARLET.MLB;2	34
TOTALS (all libraries)	36

```
1648 GETS were required to define 36 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:XFLOADER/OBJ=OBJ$:XFLOADER MSRC$:XFLOADER/UPDATE=(ENH$:XFLOADER)+EXECML$/LIB
```

0233 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

MELDR

MDL32

LALOAD
MAP

MDL32
MAP

XFLOADER
MAP

LALOADER
LIS

XFLOADER
LIS

LADAMCODE
LIS

LALOAD
LIS

MARBLI

MARBLI
MAP

LALOADER
MAP

LAAMRMCODE
LIS

LAADMCODE
LIS

MARBLI
LIS

LADMDT
LIS