Table of contents

```
                        .title  XXDPP - XXDP+ Operating System Monitor
                        .nlist  ttm

                        ;       XXDPP.MAC reconstructs the XXDP+ operating system source
                        ;
                        ;       Build procedure (RUST/RT-11):
                        ;
                        ;       %build
                        ;       macro xds:xxdpp/object:xdb:
                        ;       macro xds:xxdpp/object:xdb:/noobject/list:xdb:
                        ;       r detab
                        ;       xdb:xxdpp.lst xdb:xxdppw.lst
                        ;       ^C
                        ;       link xdb:xxdpp/exe:xdb:xxdpp/map:xdb:/cross/nobitmap
                        ;       %end
                        ;
                        ;       Edit History:
                        ;
                        ;    01 01-Jan-2000 IJH Disassemble and study XXDP+
                        ;    02 0n-Mar-2021 IJH Complete initial source code recovery
                        ;
                        ;       Source introduction:
                        ;
                        ;       XXDP was/is the diagnostic operating system for PDP-11 computers.
                        ;       This source file was created by reverse assembling the binary
                        ;       image of the XXDP+ HMDLD0 monitor found on XXDP23 distribution.
                        ;
                        ;       XXDP constitutes the de factor definition of the PDP-11, as
                        ;       anyone who writes an emulator soon finds out. It's the toughest
                        ;       the PDP-11 systems, case-hardened by its use on only partially
                        ;       functioning systems.
                        ;
                        ;       XXDP's architecture is based on a simple, near-boolean level
                        ;       state machine. There is rarely any analytic vagueness regarding
                        ;       system state.
                        ;
                        ;       XXDP has a remarkably flat structure. Registers rarely need
                        ;       to be saved/restored across routines (less than 10 instances).
                        ;       All parameters and results are passed in registers, obeying
                        ;       a strict usage protocol.
                        ;
                        ;       Because of the fixed space restrictions placed on the monitor,
                        ;       code compression was always required to find space for new
                        ;       functionality. XXDP uses many software techniques to achieve
                        ;       that goal. The code is heavily compressed.
                        ;
                        ;       One overarching simplification is the almost complete absence of
                        ;       sanity testing. It will accept any disk volume as an XXDP volume,
                        ;       no matter how crazy the directory structure might appear. It tests
                        ;       only for conditions that make it impossible to continue.
                        ;
                        ;       The original XXDP+ would of course have had separate source modules
                        ;       for the monitor and the various drivers. For this stage of the
                        ;       recovery process I've thought it best to have everything in a single
                        ;       source module with no external dependencies.
                        ;
                        ;       The monitor source code translation is complete with this release.
```

```
                                        ;       However, the documentation requires a programmer's guide and a system
                                        ;       logic manual, at some distant point in time. For clarity I have
                                        ;       not used macro definitions for system EMT calls in this, preferring
                                        ;       to see all the binary code instructions. A later release should
                                        ;       employ macros for system calls. The comments can also be improved.
                                        ;
                                        ;       The XXDP monitor itself is restricted to read-only support for
                                        ;       system media. The other half of the system, that creates and writes
                                        ;       files, is buried in a DRVCOM package that are embedded in the UPD1,
                                        ;       UPD2, PATCH and XTECO utilities and the stand-alone drivers.
                                        ;
                                        ;       I began looking at XXDP around year 2000 when I used it to test
                                        ;       a PDP-11 emulator I'd written. I got curious and wrote a simple
                                        ;       disassembler and began annotating it (which I came back to in
                                        ;       2010 and 2015). At first I approached the project as an act of
                                        ;       diligence: I thought the source was important for the history of
                                        ;       the PDP-11. However, it turned out to be a fascinating task and
                                        ;       it was a real joy to see the operating system as a whole slowly
                                        ;       emerge. There were so many subtleties to be discovered. There's
                                        ;       some horrible HELLO WORLD coding here and there, but most of it
                                        ;       is tight and the state machine design is highly disciplined.
                                        ;
                                        ;       Some grateful acknowledgements:
                                        ;
                                        ;       I spent so much time in Al Kossow's amazing bitsavers.org that I
                                        ;       thought I should start paying rent. I crawled endlessly through
                                        ;       diagnostics, looking for tiny clues. Joerg Hoppe's extensive XXDP
                                        ;       microfiche contributions to bitsavers included some critical sources.
                                        ;       A sometime DEC diagnostic programmer, Michael Morony, who visited
                                        ;       alt.sys.pdp11 some years ago, was kind enough to dig up and send
                                        ;       me a copy of MACROM.MAC, the XXDP+ system macro module, which was
                                        ;       truly invaluable: I had "names" for the system services. There have
                                        ;       been quite a few valuable websites that have dedicated time and space
                                        ;       to XXDP over the years from which I have gleaned information.
```

```
                                .sbttl  XXDP API                                              (data)

                                ;       I've renamed the system services to reflect their functional
                                ;       role within the monitor. The original service names are listed
                                ;       in the MACROM column
                                ;
                                ;  EMT  XXDPP   MACROM  Function
                                ;  ---  ------  ------  --------
                                ;  000  GetLin  GCmdSt  Get terminal/batch command line
                                ;  001  ParFld  GToken  Parse next command line field
                                ;  002  TypMon  PutLin  Type (relocated) monitor message
                                ;  003  TypMsg  TypMsg  Type (unrelocated) message
                                ;  004  PutChk  PutChr  Display character, check for Ctrl/C
                                ;  005  GetAvl  CKybd   Check keyboard character available
                                ;  006  GetChk  GetChr  Get character and check for Ctrl-C
                                ;  007  Newlin  CrLf    Display newline
                                ;  010  PutTab  Tabs    Display tab
                                ;  011  ParOct  GetNum  Parse octal number
                                ;  012  OpnFil  Open    Open file
                                ;  013  CloFil  Close   Close file
                                ;  014  LoaFil  Load    Load LDA-format program
                                ;  015  ReaWrd  GetWrd  Input word from file
                                ;  016  ReaByt  GetByt  Input byte from file
                                ;  017  PutCha  OneChr  Display character
                                ;  020  ReaNxt  NxtBlk  Read next sequential block
                                ;  021  ReaBlk  BkRead  Read any block
                                ;  022  SetAbt  SetErr  Set abort location
                                ;  023  JmpAbt  Error   Jump to abort location
                                ;  024  CmpSpc  CmpNam  Compare filespecs
                                ;  025  SpcAsc  UPkNam  Convert Rad50 to ascii
                                ;  026  SetLin  KSwitch Set terminal buffer address/length
                                ;  027  GetDat  Date    Get the system date
                                ;  030  OctAsc  IToA    Convert octal to ascii
                                ;  031  GetDev  Default Get system device information
                                ;  032  RptFld  RToken  Repeat the current field
                                ;  033  LptMod  LinePtr Write output to printer
                                ;  034  TerMod  NoPrtr  Restore terminal output
                                ;  035  LoaSup  AutoLoad Load supervisor program
                                ;  036  ParDec  GetDec  Parse decimal number
                                ;  037  PadTer  Fill    Write nulls to the terminal
                                ;  040  PshBat  PutScp  Set batch mode
                                ;  041  PopBat  CChain  Set terminal mode
                                ;  042  GetCom  Comm    Get monitor information common
                                ;  043  GetDrv  Rdrive  Copy the system driver
                                ;  044  TypBrk  FrcTyp  Type breakthrough message

                                        .MACRO  API NAM COD
                                         .macro nam
                                           emt  cod
                                         .endm
                                        .ENDM

   000000                               API GetLin 0    ;
   000000                               API ParFld 1    ;
   000000                               API TypMon 2    ;
   000000                               API TypMsg 3    ;
   000000                               API PutChk 4    ;
```

```
000000                                  API GetAvl 5    ;
000000                                  API GetChk 6    ;
000000                                  API NewLin 7    ;
000000                                  API PutTab 10   ;
000000                                  API ParOct 11   ;
000000                                  API OpnFil 12   ;
000000                                  API CloFil 13   ;
000000                                  API LoaFil 14   ;
000000                                  API ReaWrd 15   ;
000000                                  API ReaByt 16   ;
000000                                  API PutCha 17   ;
000000                                  API ReaNxt 20   ;
000000                                  API ReaBlk 21   ;
000000                                  API SetAbt 22   ;
000000                                  API JmpAbt 23   ;
000000                                  API CmpSpc 24   ;
000000                                  API SpcAsc 25   ;
000000                                  API SetLin 26   ;
000000                                  API GetDat 27   ;
000000                                  API OctAsc 30   ;
000000                                  API GetDev 31   ;
000000                                  API RptFld 32   ;
000000                                  API LptMod 33   ;
000000                                  API TerMod 34   ;
000000                                  API LoaSup 35   ;
000000                                  API ParDec 36   ;
000000                                  API PadTer 37   ;
000000                                  API PshBat 40   ;
000000                                  API PopBat 41   ;
000000                                  API GetCom 42   ;
000000                                  API GetDrv 43   ;
000000                                  API TypBrk 44   ;

                                        .MACRO  ASSUME EX1 CND EX2 COM
                                         .iif cnd <ex1>-<ex2>,.mexit
                                         .error <ex1>-<ex2> ;;;assume ex1 cnd ex2: com
                                        .ENDM

                                        .MACRO  FALL C
                                         .iif eq c-.,.mexit
                                         .error c-. ;fall c
                                        .ENDM

                                        .MACRO  STACK C D E F G H I J
                                         maval.=0
                                         .irp manam.,<c d e f g h i j>
                                          sp.'manam.=maval.
                                          maval.=maval.+2
                                         .endr
                                        .ENDM
```

```
                                .sbttl  constants                                              (data)

                                ;       CPU and device vectors

        000004                          v$ebus  = 4             ; bus error
        000010                          v$ecpu  = 10            ; cpu error
        000030                          v$eemt  = 30            ; emt
        000100                          v$eltc  = 100           ; line clock
        000104                          v$ekwp  = 104           ; kw11p clock

                                ;       Hardware registers

        177560                          TKS     = 177560        ; keyboard CSR
        177562                          TKB     = 177562        ; keyboard buffer
        177564                          TPS     = 177564        ; terminal output CSR
        177566                          TPB     = 177566        ; terminal output buffer
        177776                          PSW     = 177776        ; PSW
        000001                           cbit   = 1             ; PSW carry bit
                                                                ;
        177546                          LTC     = 177546        ; line clock
        172540                          KWP     = 172540        ; KW11P clock
        177514                          LPT     = 177514        ; line printer

                                ;       S$YCFG - hardware config flags

        000001                          syLTC$  = 1             ; line clock present
        000002                          syKWP$  = 2             ; programmable clock present
        000004                          syLPT$  = 4             ; line printer present
        000010                          syNUB$  = 10            ; NoUniBus
        000020                          sy50H$  = 20            ; 50 Hertz clock

                                ;       Terminal constants

        000011                          ht      = 11            ; tab
        000012                          lf      = 12            ; line feed
        000015                          cr      = 15            ; carriage return
        000040                          space   = 40            ; space
        000177                          del     = 177           ; delete/rubout
                                                                ;
                                ;       Control keys

        000003                          ctrlc   = 3             ; ^C - cancel activity or pause DRS batch file
        000021                          ctrlq   = 21            ; ^Q - resume terminal output
        000023                          ctrls   = 23            ; ^S - pause terminal output
        000025                          ctrlu   = 25            ; ^U - cancel line
        000030                          ctrlx   = 30            ; ^X - resume after batch WAIT
        000032                          ctrlz   = 32            ; ^Z - terminate DRS batch file
```

```
                                .sbttl  Monitor structure                                       (data)

                                ;       Memory layout
                                ;
                                ;       kw memory  kw image disk pointer label
                                ;       ---------  -------- ---- ------- -----
                                ;       28 160000   4 20000      s$ytop: x$xtop:  I/O page
                                ;          157777     17777              x$xlim:  Address limit
                                ;          157xxx     17xxx              x$xdrv:  Driver
                                ;       27 154000   3 14000              x$xsta:  Static
                                ;          152000     12000  12  s$yper: x$xper:  Permanent
                                ;       26 150000   2 10000  10  s$ytra: x$xtra:  Transient
                                ;          146000     06000  06          x$xbat:  Batch
                                ;       25 144000   1 04000              x$xhgh:  Init high
                                ;          141000     01000  01          x$xini:  Init and MFD
                                ;       24 140000   0 00000  00  s$yrel: x$xbot:  Boot
                                ;          137000                s$ysup: x$xsup:  Supervisor
                                ;
                                ;       Monitor region block numbers

                                ;       moBOO.  = 0             ; boot
                                ;       moMFD.  = 1             ; MFD block
        000006                          moBAT.  = 6             ; batch area block
        000010                          moTRA.  = 8.            ; transient area block
        000012                          moCLI.  = 10.           ; cli area block
        001414                          moOVL.  = 1414          ; monitor overlay length

                                ;       MFD block

        000000                          mf.ufd  = 0     ;1002   ;
        000024                          mf.mon  = 24    ;1026   ;

                                ;       Init information block

                                ;       in.50h  = 0     ;1000   ; 50 hertz flag
                                ;       in.aut  = 2     ;1002   ; automated startup flag

                                ;       Command line structure

                                ;       cl.ptr  = 0             ; command line base pointer
                                ;       cl.len  = 2             ; command line length
                                ;        cllen. = 44.   ;54     ; default command line length
                                ;        clavl. = 42.   ;52     ; available characters

                                ;       Manual control (CMI/SMI)

        000001                          scMAN$  = 1

                                ;       Device information block

        000000                          dv.nam  = 0     ;"DL"   ; driver name
        000002                          dv.uni  = 2     ;"0"    ; device unit
        000003                          dv.med  = 3     ;dlMED. ; media code
        000002                           dvRK5. = 2     ;2      ; DK: disk (RT-11 RK:)
        000014                           dvRL1. = 14    ;12.    ; DL: disk

                                ;       Driver interface
```

```
     58
     59         177752                          dr.buf  = -22.  ; 752   ; buffer pointer
     60         177754                          dr.ent  = -20.  ; 754   ; directory entry number in segment
     61         177756                          dr.fnm  = -18.  ; 756   ; rad50 filename
     62         177764                          dr.sbl  = -12.  ; 764   ; first file block
     63         177766                          dr.opn  = -10.  ; 766   ; open file function
     64         177770                          dr.rst  = -8.   ; 770   ; read monitor function
     65         177772                          dr.tra  = -6    ; 772   ; transfer function
     66         177774                          dr.dev  = -4    ; 774   ; get device info function
     67         177776                          dr.uni  = -2    ; 776   ; device unit
     68         177777                          dr.sts  = -1    ; 777   ; operation status
     69         000000                          dr.csr  =  0    ; 000   ; CSR address
     70         000002                          io.wct  =  2    ; 002   ; word count
     71         000004                          io.buf  =  4    ; 004   ; buffer address
     72         000006                          io.blk  =  6    ; 006   ; block number
     73         000010                          io.ufd  =  8.   ; 012   ; (user file) directory start block
     74         000012                          io.spc  =  10.  ; 014   ; ascii filespec
     75
     76                                 ;       dr.sts - driver status
     77
     78                                 ;       drSUC.  = 0             ; an absence of errors
     79                                 ;       drTRA.  = -1            ; dr.tra - transfer error
     80                                 ;       drFNF.  = 1             ; dr.opn - file not found
```

```
                                .sbttl  RL01/02 device structure                                (data)

                                ;       registers

        174400                          dlcsr.  = 174400        ; csr
        000000                          dl.csr  = 0             ; csr
        000002                          dl.buf  = 2             ; 02    ; buffer
        000004                          dl.adr  = 4             ; track/sector
        000006                          dl.wct  = 6             ; word count (write)
        000006                          dl.dat  = 6             ; data value (read)

                                ;       geometry
                                                                ;
        000400                          dlcyl.  = 256.          ; cylinders
        000002                          dlrl2.  = 2             ; RL02 cylinder factor
        000002                          dlhds.  = 2             ; heads
        000024                          dlbpt.  = 20.           ; blocks per track
        000100                          dltrk.  = 64.           ; track
        024000                          dl1sz.  = 10240.        ; blocks per RL01
        050000                          dl2sz.  = 20480.        ; blocks per RL02
        024000                          dlsiz.  = dl1sz.        ;

                                ;       CSR definitions.

        000001                          dlrdy$  = 1             ; operation complete ("drive ready")
        000016                          dlfun$  = 7*2           ; function
        000200                          dlgo$   = 200           ; clear to start operation
                                                                ; actually "controller ready"
        001400                          dluni$  = 1400          ; unit number (0..3)
        176377                          dlun$m  = 176377                ; unit mask
        100000                          dlerr$  = 100000                ; error seen

        000000                          dlNOP.  = 0*2   ; 00    ; nop
        000004                          dlSTA.  = 2*2   ; 04    ; get status
        000006                          dlSEE.  = 3*2   ; 06    ; seek
        000010                          dlRHD.  = 4*2   ; 10    ; read header
        000014                          dlREA.  = 6*2   ; 14    ; read data
        000016                          dlRDX.  = 7*2   ; 16    ; read with no header check

                                ;       Seek

        000001                          dlsee$  = 1     ;dl.adr ; seek activate
        000004                          dldir$  = 4             ; seek direction
        000020                          dlhea$  = 20            ; head select

        177600                          dltrk$  = 177600        ;dl.dat ; track mask (RL02)

                                ;       Get device status/size

        000001                          dlmrk$  = 1     ;dl.adr ; marker
        000002                          dlsts$  = 2             ; get status
        000010                          dlrst$  = 10            ; reset errors
        000013                          dlrep$  = 13            ; get device size (rst,sts,mrk)

        000200                          dlrl2$  = 200   ;dl.dat ; RL02
```

```
                                .sbttl  XXDP disk structure                                    (data)

                                ;       HOMBLK - XXDP MFD block

        000001                          hbblk.  = 1             ; block 1
        001000                          hbbas.  = 1000          ; disk byte address

        000000                          hb.nxt  = 0     ;0      ; next block    (always zero)
        000002                          hb.ufd  = 2     ;       ; first UFD directory block
                                ;       hb.dbc  = 4     ;       ; directory block count
                                ;       hb.map  = 6     ;       ; first map block
                                ;       hb.mbc  = 10    ;       ; map block count
                                ;       hb.mfd  = 12    ;       ; MFD block (self-reference to block 1)
                                ;       hb.ver  = 14    ;       ; XXDP version (never seen used)
                                ;        dbxv2. =  1002 ;0      ; XXDP V2 version code
                                ;       hb.tot  = 16    ;       ; total blocks
                                ;       hb.res  = 20    ;       ; reserved blocks
                                ;       hb.int  = 22    ;       ; interleave factor
                                ;       hb.boo  = 24    ;0      ; boot block
        000026                          hb.mon  = 26    ;30     ; monitor block
                                ;       hb.ref  = 30    ;       ; MFD refreshed flag (not-0 = yes)
                                ;       hbbbs.  = 30            ; block size

                                ;       Directory entry

        000000                          en.fil  = 0     ;fil    ; rad50 filename (0=>deleted)
        000002                          en.nam  = 2     ;nam    ;
        000004                          en.typ  = 4     ;typ    ;
        000006                          en.dat  = 6     ;       ; file date and contiguous flag
        000012                          en.ffb  = 12    ;       ; first-free byte
        000014                          en.sta  = 14    ;       ; start block
        000016                          en.len  = 16    ;       ; length in blocks
        000020                          en.lst  = 20    ;       ; last block in use
        000022                          en.flg  = 22    ;       ; flags
        000024                          enbbs   = 24    ;18.    ;

                                ;       en.dat - XXDP/DOSbatch date
                                ;
                                ;       date = (year*1000.) + day-in-year;
                                ;
                                ;       endat$  = 077777        ; date field mask
                                ;       enctg$  = 100000        ; contiguous file flag
```

```
                                  .sbttl  boot engine                                           (boot)
000000                            .asect
000000  000000                    x$xlow: . = 0

                                  ;       XXDP boot enters at location zero, like everyone else

                                                                          ; boot communication area
000000  000240                    bo$pri: nop             ;0000   ; boot primary entry point
000002  000407                            br      bo$con  ;0002   ; continuation
000004  000006                            .word   .+2     ;0004   ; bus trap vector
000006  000000                            .word   0       ;0006   ;
000010  000012                            .word   .+2     ;0010   ; cpu trap vector
000012  000000                            .word   0       ;0012   ;
000014  000000                            .word   0       ;0014   ; bpt vector skipped
000016  000000                            .word   0       ;0016   ;
000020  174400                    b$ocsr: .word   dlcsr.  ;0020   ; CSR address      (patch point)(note)
000022  000240                    bo$con: nop             ;0022   ; boot continuation
000024  000407                            br      bo$eng  ;0024   ; boot mainline

000026  000000                    b$otrk: .word   0       ;0026   ; track - cylinder mask + initial sector
000030  000400                    b$owct: .word   256.    ;0030   ; word count
000032  000000                    b$oblk: .word   0       ;0032   ; block
000034  000000                    b$ocyl: .word   0       ;0034   ; cylinder number
000036     000                    b$osec: .byte   0       ;0036   ; sector: 0..39.
000037     000                    b$ohea: .byte   0       ;0037   ;
000040  000000                            .word   0       ;0040   ;
000042  000000                            .word   0       ;0042   ;

                                  ;       BO$ENG - Boot engine and start

                                          .enabl  lsb
000044  012706  040000            bo$eng: mov     #40000,sp       ; some random stack
000050  016701  177744                    mov     b$ocsr,r1       ; r1 -> RL01 csr
000054  012767  001000  000362            mov     #1000,b$obuf    ; read buffer address
000062  062767  000002  177736            add     #2,b$otrk       ; sector needs to +2 for MFD and monitor

                                  ;       MFD/monitor loop

000070  016746  177734            10$:    mov     b$owct,-(sp)    ; (sp) = remaining word count

                                  ;       Block loop

000074  162716  000400            20$:    sub     #256.,(sp)      ; shave off 256. words
000100  101404                            blos    30$             ; too much
000102  012767  000400  000336            mov     #256.,b$otwc    ; transfer a full block
000110  000405                            br      40$             ;
000112  011667  000330            30$:    mov     (sp),b$otwc     ; less than a block - count is negative
000116  062767  000400  000322            add     #256.,b$otwc    ; add to get partial block word count
                                                                  ;
000124  004767  000430            40$:    call    bo$see          ; seek
000130  004767  000240                    call    bo$adr          ; setup address/wordcount/buffer
000134  042711  000016                    bic     #dlfun$,(r1)    ; clear the function
000140  052711  000014                    bis     #dlREA.,(r1)    ; set read function
000144  004767  000164                    call    bo$opr          ; go go and wait
000150  000402                            br      60$             ; continue just below

                                  ;       I/O error halt and retry
```

```

000152  000000                  bo$hlt: halt                    ; stop the world
000154  000733                          br      bo$eng          ; restart the world

                                ;       Continue boot

000156  005716                  60$:    tst     (sp)            ; read completed?
000160  003403                          ble     70$             ; yes - MFD or monitor read done
000162  004767  000312                  call    bo$nxt          ; no - advance our cause
000166  000742                          br      20$             ; and read the next block's worth

                                ;       MFD or monitor read done?

000170  005726                  70$:    tst     (sp)+           ; pop temporary transfer word count
                                                                ;
000172  005727                          tst     (pc)+           ; MFD or monitor read done?
000174  000000                  80$:    .word   0               ; 0=MFD, -1=Monitor
000176  001031                          bne     90$             ; monitor read - finish up
000200  005167  177770                  com     80$             ; MFD - monitor next time

                                ;       MFD read done
                                ;
                                ;       Setup for monitor read

        001002                          b$omfd = 1002           ; 1000 - boot MFD buffer
        001026                          b$omon = b$omfd+mf.mon  ; 1026 - MFD monitor block
        007400                          bowct. = 4096.-256.     ; 7400 - monitor wordcount - boot block

000204  013767  001026  177620          mov     @#b$omon,b$oblk ; monitor block from the MFD
000212  012767  007400  177610          mov     #bowct.,b$owct  ; monitor word count
000220  062767  000001  177604          add     #1,b$oblk       ; skipping the boot block
000226  004767  000444                  call    bo$geo          ; cylinder/sector/head geometry
000232  004767  000052                  call    bo$cyl          ; get the cylinder
000236  012761  000001  000004          mov     #dlsee$,dl.adr(r1) ; seek flag
000244  052761  000004  000004          bis     #dldir$,dl.adr(r1) ; we know its forward
000252  056761  177550  000004          bis     b$otrk,dl.adr(r1)  ; cylinder (and sector for MFD read)
000260  000703                          br      10$             ; do the monitor read loop

                                ;       Monitor read done
                                ;
                                ;       Setup monitor CSR/unit and launch init

000262  011167  014624'         90$:    mov     (r1),d$runi     ; pass the unit number to the driver (boot)
000266  042767  176377  014624'         bic     #dlun$m,d$runi  ; mask the unit
000274  000367  014624'                 swab    d$runi          ; into low byte
000300  010167  014626'                 mov     r1,d$rcsr       ; IOB csr
                                                                ;
                                ;       Launch the init engine  ;
                                                                ;
000304  000167  002340'                 jmp     in$eng          ; galacto city
                                        .dsabl  lsb

                                ;       BO$CYL - Cylinder calculation

000310  016746  177520          bo$cyl: mov     b$ocyl,-(sp)
000314  012703  000007                  mov     #7,r3           ; shift count
000320  006316                  10$:    asl     (sp)            ; shift
```

```
000322  005303                          dec     r3              ;
000324  001375                          bne     10$             ;
000326  012667  177474                  mov     (sp)+,b$otrk    ; cylinder
000332  000207                          return

                                ;       BO$OPR - Boot operation

                                        .enabl  lsb
000334  042711  000200          bo$opr: bic     #dlgo$,(r1)     ; clear to activate
000340  032711  100200          bo$wai: bit     #dlerr$!dlgo$,(r1) ; wazzup DL?
000344  001775                          beq     bo$wai          ; we are waiting
000346  100401                          bmi     30$             ; error
000350  000207                  20$:    return                  ; fine
000352  000167  177574          30$:    jmp     bo$hlt          ; fail - halt

                                ;       BO$CHK - Wait for seek to complete

000356  032711  100001          bo$chk: bit     #dlerr$!dlrdy$,(r1) ; ready/error
000362  001775                          beq     bo$chk          ; neither
000364  100371                          bpl     20$             ; fine - return
000366  004767  000056                  call    bo$res          ; bummer - reset (which calls bo$wai above)
000372  000767                          br      30$             ; and go halt via 30$
                                        .dsabl  lsb

                                ;       BO$ADR - Combine all the addressing bits

000374  016746  177426          bo$adr: mov     b$otrk,-(sp)    ; cylinder (and MFD sector)
000400  156716  177432                  bisb    b$osec,(sp)     ; sector
000404  105767  177427                  tstb    b$ohea          ;
000410  001402                          beq     10$             ;
000412  052716  000100                  bis     #100,(sp)       ; head
000416  012661  000004          10$:    mov     (sp)+,dl.adr(r1); combined track
000422  016746  000020                  mov     b$otwc,-(sp)    ; transfer word count
000426  005416                          neg     (sp)            ;
000430  012661  000006                  mov     (sp)+,dl.wct(r1); negated word count
000434  016761  000004  000002          mov     b$obuf,dl.buf(r1); buffer address
000442  000207                          return

000444  000000                  b$obuf: .word   0               ; buffer address
000446  000000                  b$otwc: .word   0               ; transfer word count

                                ;       BO$RES - Reset
                                ;
                                ;       Called after an error and before a restart

000450  011146                  bo$res: mov     (r1),-(sp)
000452  042716  176377                  bic     #dlun$m,(sp)    ; clear all but unit
000456  052716  000004                  bis     #dlSTA.,(sp)    ; get status
000462  052761  000013  000004          bis     #dlrep$,dl.adr(r1) ; get device status
000470  012611                          mov     (sp)+,(r1)      ; take that CSR!
000472  004767  177642                  call    bo$wai          ; wait and check
000476  000207                          return

                                ;       BO$NXT - Next block/track/sector/buffer address

000500  062767  000002  177330  bo$nxt: add     #2,b$osec       ; two sectors per block
000506  126727  177324  000050          cmpb    b$osec,#40.     ; end of track?
```

```
000514  002415                          blt     10$             ; nope
000516  105067  177314                  clrb    b$osec          ; sector = 0
000522  105267  177311                  incb    b$ohea          ;
000526  142767  000376  177303          bicb    #376,b$ohea     ;
000534  001005                          bne     10$             ;
000536  062767  000200  177262          add     #200,b$otrk     ;
000544  005267  177264                  inc     b$ocyl          ;
000550  062767  001000  177666  10$:    add     #^o1000,b$obuf  ; next transfer buffer address
000556  000207                          return

                                ;       BO$SEE - Seek

000560  042711  000016          bo$see: bic     #dlfun$,(r1)    ; clear function
000564  052711  000010                  bis     #dlRHD.,(r1)    ; read header
000570  004767  177540                  call    bo$opr          ;
                                                                ;
000574  016146  000006                  mov     dl.wct(r1),-(sp); wct holds current track address
000600  012761  000001  000004          mov     #dlsee$,dl.adr(r1) ; we will be seeking
000606  105767  177225                  tstb    b$ohea          ;
000612  001403                          beq     10$             ;
000614  052761  000020  000004          bis     #dlhea$,dl.adr(r1) ; set the head
000622  042716  000177          10$:    bic     #177,(sp)       ; determine cylinder
000626  166716  177174                  sub     b$otrk,(sp)     ; subtract what we want
000632  103006                          bcc     20$             ; we are going forward
000634  005416                          neg     (sp)            ; we are going backwardsd
000636  042716  000177                  bic     #177,(sp)       ;
000642  052761  000004  000004          bis     #dldir$,dl.adr(r1) ; set negative direction
000650  052661  000004          20$:    bis     (sp)+,dl.adr(r1); tell the RL01
000654  042711  000016                  bic     #dlfun$,(r1)    ; clear function
000660  052711  000006                  bis     #dlSEE.,(r1)    ; function = read data
000664  042711  000200                  bic     #dlgo$,(r1)     ; go
000670  004767  177462                  call    bo$chk          ; wait and check errors
000674  000207                          return

                                ;       BO$GEO - Cylinder/sector/head geometry

000676  005067  177132          bo$geo: clr     b$ocyl          ; clear proto cylinder
000702  016703  177124                  mov     b$oblk,r3       ; r3 = target block
000706  012702  000050                  mov     #40.,r2         ; r2 = sectors per cylinder
                                                                ;
000712  160203                  10$:    sub     r2,r3           ; get cylinder
000714  103403                          bcs     20$             ; no more sectors here
000716  005267  177112                  inc     b$ocyl          ; up cylinder
000722  000773                          br      10$             ;
000724  060203                  20$:    add     r2,r3           ; backout the subtraction above
                                                                ;
000726  005067  177104                  clr     b$osec          ;
000732  012702  000024                  mov     #20.,r2         ; r2 = sectors per head
000736  160203                  30$:    sub     r2,r3           ; see how many fit
000740  103403                          bcs     40$             ; no more fit
000742  105267  177071                  incb    b$ohea          ; flip the head
000746  000773                          br      30$             ;
000750  060203                  40$:    add     r2,r3           ; backout subraction
000752  006303                          asl     r3              ; r2 * 2
000754  110367  177056                  movb    r3,b$osec       ; is the sector number
000760  000207                          return

```

```
    229 000762                                  .blkw   7               ; boot round-up
```

```
                                .sbttl  Init engine (low memory)                                (init)
000000                          .psect  xxdp                    ; location 1000
000000                          x$xini:

                                ;       Init prologue

000000  000000                  i$n50h: .word   0       ;1000   ; 60/50 hertz clock       (patch point)
000002  000000                  i$naut: .word   0       ;1002   ; automated startup flag  (patch point)
000004                          x$xgap: .blkb   2334


                                ;       IN$ENG - Initialization engine
                                ;
                                ;       Stack, terminal and EMT vector

002340  012706  013324'         in$eng: mov     #s$ystk,sp      ; stack
002344  012767  177564  010772          mov     #TPS,s$ytps     ; tps
002352  012767  177566  010766          mov     #TPB,s$ytpb     ; tpb
                                                                ;
002360  012737  012520' 000030          mov     #em$eng,@#v$eemt; low memory em$eng             (note)
002366  012737  000340  000032          mov     #340,@#v$eemt+2 ; (is never invoked)

                                ;       Size memory

002374  012737  002450' 000004          mov     #20$,@#4        ; trap to 20$
002402  012737  000340  000006          mov     #340,@#6        ;
002410  012701  004000                  mov     #4000,r1        ; 1kw counter
002414  012703  000004                  mov     #4,r3           ; kw accumulator
002420  012700  020000                  mov     #20000,r0       ; start address
                                                                ;
002424  012710  000000          10$:    mov     #0,(r0)         ; this test is repeated at in$siz
002430  004767  000154                  call    in$siz          ; count 1kw
002434  005203                          inc     r3              ; kw counter up
002436  060100                          add     r1,r0           ; advance address
002440  020027  160000                  cmp     r0,#160000      ; end of universe?
002444  103767                          blo     10$             ; not quite
002446  000402                          br      30$             ; yep

                                ;       Sizer trap
                                ;
                                ;       r0 ->   memory top - typically 160000

002450  062706  000004          20$:    add     #4,sp           ; dump SetAbt pc/ps stack frame
                                                                ;
                                                                ;memory image
002454  010067  010654          30$:    mov     r0,s$yrel       ;160000 20000
002460  162767  020000  010646          sub     #20000,s$yrel   ;140000 00000 relocation constant
002466  010067  010632                  mov     r0,s$ytra       ;
002472  162767  010000  010624          sub     #10000,s$ytra   ;150000 10000 transient area
002500  010067  010622                  mov     r0,s$yper       ;
002504  162767  006000  010614          sub     #6000,s$yper    ;152000 12000 permanent area

                                ;       Checksum the transient .5k area

002512  012702  007000'                 mov     #x$xtra,r2      ;10000  ; from 10000 to 12000
002516  062267  010674          40$:    add     (r2)+,s$y5ck    ;       ; aka s$ytra to s$yper
002522  020227  011000'                 cmp     r2,#x$xper      ;12000  ; done?
```

```
002526  001373                          bne     40$                     ; not bloodly likely
                                                                        ;
002530  010067  010606                  mov     r0,s$ysup       ;01000  ; supervisor load address
002534  162767  037000  010600          sub     #37000,s$ysup   ;       ;

                                ;       Calculate MMU page count

002542  010367  012000                  mov     r3,s$ykwd       ;#28.   ; memory kiloword size
002546  010302                          mov     r3,r2           ;       ; calculate number of MMU pages
        000005                          .rept   5                       ; there are 32 per kiloword
                                        asl     r2              ;       ; 2^5 is 32
                                        .endr                   ;       ;
002562  005302                          dec     r2              ;       ; (n*32)-1 (to make it tangible)
002564  010267  012004                  mov     r2,s$ypgs       ;511.   ; 32w pages in system - 1

                                ;       Copy the kword digits for later display

002570  006303                          asl     r3              ;       ; kwords * 2
002572  116367  002726' 000220          movb    i$nkws(r3),i$nkwd       ; first
002600  116367  002727' 000213          movb    i$nkws+1(r3),i$nkwd+1   ; second
002606  000407                          br      in$con                  ; skip to init continuation

                                ;       Test 1kw of memory at a time

002610  010002                  in$siz: mov     r0,r2           ; address
002612  012722  000000          10$:    mov     #0,(r2)+        ; test location with write
002616  032702  003777                  bit     #3777,r2        ; until we wrap
002622  001373                          bne     10$             ;
002624  000207                          return                  ; one kw more

                                ;       Relocation list, list ends negative

002626  016703  010502          in$con: mov     s$yrel,r3       ; r3 = relocation constant
002632  012702  002670'                 mov     #i$nrel,r2      ; r2 -> relocation list
002636  005712                  10$:    tst     (r2)            ; end of table?
002640  100402                          bmi     20$             ; yup
002642  060332                          add     r3,@(r2)+       ; relocate another brother
002644  000774                          br      10$             ; more

                                ;       Copy up monitor and jump to high copy

002646  012701  020000          20$:    mov     #20000,r1       ; top of low memory monitor
002652  010067  010474                  mov     r0,s$ytop       ; top of memory
002656  014140                  30$:    mov     -(r1),-(r0)     ; copy monitor up
002660  020127  003026'                 cmp     r1,#in$hgh      ;
002664  101374                          bhi     30$             ;
002666  000110                          jmp     (r0)            ; ransfer to x$xhgh:

                                ;       Relocation list
                                ;
                                ;       For a binary exact restoration of XXDP+ we need to follow the
                                ;       random order of entries in the table below.
                                ;
                                ;       Each entry is created by a "movr src,dst,idx" macro, where "idx"
                                ;       is the table index (which can be elided for for a non-exact clone).

                                        .MACRO  MOVR SRC,DST,IDX
```

```
                                                i$nr'idx' = .+2
                                                mov     src,dst
                                                .ENDM

        002670  003170'                 i$nrel: .word   i$nr1   ;  1
        002672  015262'                         .word   i$nr2   ;  2
        002674  003470'                         .word   i$nr3   ;  3
        002676  004502'                         .word   i$nr4   ;  4
        002700  004662'                         .word   i$nr5   ;  5
        002702  014724'                         .word   i$nr6   ;  6
        002704  014664'                         .word   i$nr7   ;  7
        002706  014730'                         .word   i$nr8   ;  8
        002710  015044'                         .word   i$nr9   ;  9
        002712  003454'                         .word   i$nr10  ; 10
        002714  004736'                         .word   i$nr11  ; 11
        002716  003124'                         .word   i$nr12  ; 12
        002720  015520'                         .word   i$nr13  ; 13
        002722  015544'                         .word   i$nr14  ; 14
        002724  177777                          .word   -1      ; stopper

        002726     040     060     040  i$nkws: .ascii  " 0 1 2 3 4 5 6 7 8 9"  ; memory size table
        002731     061     040     062
        002734     040     063     040
        002737     064     040     065
        002742     040     066     040
        002745     067     040     070
        002750     040     071
        002752     061     060     061          .ascii  "10111213141516171819"
        002755     061     061     062
        002760     061     063     061
        002763     064     061     065
        002766     061     066     061
        002771     067     061     070
        002774     061     071
        002776     062     060     062          .ascii  "202122232425262728"
        003001     061     062     062
        003004     062     063     062
        003007     064     062     065
        003012     062     066     062
        003015     067     062     070

        003020     040     040     113  i$nkwd: .ascii  "  K "  ;"28K " ; memory size filled in
        003023     040
        003024     000     000                  .byte   0,0             ; disassembly                   (note)
                                        ;       .byte   0,12    ;10.    ; XXDP+ live    (savm?) (slip?) (note)
                                                .even
```

```
                                .sbttl  Init engine (high memory)                               (init)

                                ;       Beginning of high area

003026  012706  013324'         in$hgh: mov     #s$ystk,sp      ; use the (unrelocated) system stack
003032  010700                          mov     pc,r0           ;
003034  062700  177772                  add     #in$hgh-.,r0    ; abort restarts us
003040                                  SetAbt  ;in$hgh         ; (EMT vector-> low memory EM$ENG)   (note)

003042  005002                          clr     r2              ; 0:1000 trap catchers
003044  012703  000002                  mov     #2,r3           ;
003050  010322                  10$:    mov     r3,(r2)+        ; 000000: .word .+2
003052  005022                          clr     (r2)+           ; 000002: .word 0       ; aka HALT
003054  062703  000004                  add     #4,r3           ; and so on
003060  020227  001000                  cmp     r2,#1000        ; upto 1000
003064  001371                          bne     10$             ;

003066  004767  007402                  call    em$rst          ; relocate EMT vector
003072  004767  001562                  call    in$iob          ; relocate the IOB
                                                                ;
003076  004775  177774                  call    @dr.dev(r5)     ; get ye device name "DD"
003102  010067  010232                  mov     r0,s$ydev       ; s$ydev -> d$rdev: .ascii "DDu"<med>
003106  116767  011512  011424          movb    d$runi,s$yuni   ; s$yuni
003114  016767  011506  011412          mov     d$rcsr,s$ycsr   ; s$ycsr
                                                                ;
003122                                  movr    #c$lbuf,c$llin,12; c$olin - relocated
003130  012767  000052  010224          mov     #clavl.,c$llen  ; line available length
                                                                ;
003136  013767  000002' 005744          mov     @#i$naut,c$laut ; copy automated startup flag
                                                                ;
003144  010700                          mov     pc,r0           ;
003146  062700  000062                  add     #50$-.,r0       ;
003152                                  SetAbt  ;50$            ; abort skips messages
003154                                  TerMod                  ;
003156                                  NewLin                  ; new line even in quiet mode
                                                                ;
003160  005737  000002'                 tst     @#i$naut        ; quiet mode?
003164  001021                          bne     50$             ; yes
                                                                ;
003166                                  movr    #i$mmon,r0,1    ;
003172                                  TypMsg                  ; CHMDLD0 XXDP+ DL MONITOR"
003174  012700  003765'                 mov     #i$mboo,r0      ;
003200                                  TypMon                  ; BOOTED VIA UNIT
003202  116700  011416                  movb    d$runi,r0       ;
003206  042700  177770                  bic     #^c7,r0         ; isolate unit number
003212  062700  000060                  add     #'0,r0          ; make it readable
003216                                  PutChk                  ;
003220                                  NewLin                  ;
003222  012700  003020'                 mov     #i$nkwd,r0      ; "28K" (#i$nkwd is unrelocated)
003226                                  TypMsg                  ;       (should be TypMon)      (bug)(note)

003230  010746                  50$:    mov     pc,-(sp)        ;
003232  062716  000130                  add     #80$-.,(sp)     ; cpu traps to 80$
003236  012637  000010                  mov     (sp)+,@#10      ;
003242  012737  000340  000012          mov     #340,@#12       ;
                                                                ;
003250  010746                          mov     pc,-(sp)        ;
```

```
003252  062716  000062                  add     #60$-.,(sp)     ;
003256  012637  000004                  mov     (sp)+,@#4       ; bus traps to 60$
003262  012737  000340  000006          mov     #340,@#6        ;
                                                                ;
003270  012737  000340  177776          mov     #340,@#PSW      ; check psw          (trap to 60$)
003276  000007                          mfpt                    ; get processor type (trap to 80$)
003300  022700  000003                  cmp     #3,r0           ; F11: qbus 11/23, unibus 11/24
003304  001027                          bne     90$             ; not F11
003306                                  NewLin                  ; F11 can be unibus or qbus
003310  012700  004107'                 mov     #i$mubq,r0      ; unibus question
003314                                  TypMon                  ; DOES THIS SYSTEM HAVE A UNIBUS (Y/N CR=Y)
003316                                  GetLin                  ; get response
003320                                  ParFld                  ; parse it
003322  000240                           nop                    ; ignore errors
003324  121027  000116                  cmpb    (r0),#'N        ; only recognized response is "N"
003330  001402                          beq     70$             ; not a unibus system
003332  000414                          br      90$             ; anything else
003334  022626                  60$:    cmp     (sp)+,(sp)+     ; PSW bus trap - not unibus
003336  052767  000010  011176  70$:    bis     #syNUB$,s$ycfg  ; no unibus
                                                                ;
003344  005737  000002'                 tst     @#i$naut        ; quiet?
003350  001013                          bne     100$            ; yes
003352  012700  004062'                 mov     #i$mnon,r0      ;
003356                                  TypMon                  ; "NON"
003360  000401                          br      90$             ; then "UNIBUS"

003362  022626                  80$:    cmp     (sp)+,(sp)+     ; MFPT cpu trap
003364  005737  000002'         90$:    tst     @#i$naut        ; quiet?
003370  001003                          bne     100$            ; yes
003372  012700  004067'                 mov     #i$mubs,r0      ;
003376                                  TypMon                  ; "UNIBUS SYSTEM"
                                                                ;
003400  005737  000000'         100$:   tst     @#i$n50h        ; 50 hertz time zone?
003404  001411                          beq     110$            ; no
003406  012767  000062  011142          mov     #50.,s$yltk     ; setup LTC line clock tick counters
003414  012767  000062  011144          mov     #50.,s$yktk     ; setup KWP programmable clock counters
003422  052767  000020  011112          bis     #sy50H$,s$ycfg  ; flag 50hz present

003430  010700                  110$:   mov     pc,r0           ;
003432  062700  000006                  add     #105$-.,r0      ; abort address
003436                                  SetAbt  ;105$           ; abort repeats test
003440  005737  000002'         105$:   tst     @#i$naut        ; boot auto mode?
003444  001021                          bne     in$hdw          ; yes - skip date prompt and idents

                                ;       Get the date

003446  004767  000244                  call    in$dat          ; get the date
003452                                  movr    #in$hdw,r0,10   ; relocated
003456                                  SetAbt                  ; abort skips messages
003460  012700  003744'                 mov     #i$mrad,r0      ;
003464                                  TypMon                  ; "RESTART ADDR: "
003466                                  movr    #xx$rst,r0,3    ; XXDP system restart address
003472  016701  007662                  mov     c$llin,r1       ; a buffer
003476                                  OctAsc                  ; ascify it
003500  105011                          clrb    (r1)            ; terminate string
003502  016700  007652                  mov     c$llin,r0       ; get the buffer again
003506                                  TypMsg                  ; display restart address
```

```
                                                                ;
003510                          20$:    fall    in$hdw          ; get hardware configuration

                                ;       Hardware, Config, System launch
                                ;
                                ;       KW11P, line clock, line printer

003510  012737  003544' 000004  in$hdw: mov     #10$,@#4        ; trap to 10$
003516  066737  007612  000004          add     s$yrel,@#4      ;
003524  005737  172540                  tst     @#KWP           ; KW11P
003530  016767  007576  011022          mov     h$wkwp,s$ykwp   ;
003536  052767  000002  010776          bis     #syKWP$,s$ycfg  ; KWP$ flag
                                                                ;
003544  012737  003600' 000004  10$:    mov     #20$,@#4        ; trap to 20$
003552  066737  007556  000004          add     s$yrel,@#4      ;
003560  005737  177546                  tst     @#LTC           ; line clock
003564  016767  007540  010756          mov     h$wltc,s$yltc   ;
003572  052767  000001  010742          bis     #syLTC$,s$ycfg  ; LTC$ flag
                                                                ;
003600  012737  003634' 000004  20$:    mov     #30$,@#4        ; trap to 30$
003606  066737  007522  000004          add     s$yrel,@#4      ;
003614  005737  177514                  tst     @#LPT           ; test line printer
003620  052767  000004  010714          bis     #syLPT$,s$ycfg  ; LPT$ flag
003626  012767  177514  010710          mov     #LPT,s$ylpt     ; line printer
003634                          30$:    fall    in$fin

003634  005737  000002'         in$fin: tst     @#i$naut        ; automated?
003640  001010                          bne     10$             ; yes
003642  010700                          mov     pc,r0           ;
003644  062700  000016                  add     #10$-.,r0       ;
003650                                  SetAbt  ;10$            ; abort skips message
003652                                  NewLin                  ;
003654  012700  004163'                 mov     #i$mxdp,r0      ; THIS IS XXDP+ ...
003660                                  TypMon                  ;

                                ;       Reset s$ypad because it had preset boot-time info       (note)

003662  112767  000001  007533  10$:    movb    #1,s$ypad       ; reset line padding

                                ;       Fill in missing trap catchers between 0..20
                                ;
                                ;       Diagnostics usually fill 0000:1000 with trap catchers

003670  012700  000020                  mov     #20,r0          ; r0 -> top of the area
003674  012701  000016                  mov     #16,r1          ; r1 -> a word below
003700  005040                  20$:    clr     -(r0)           ; 016: .word  0
003702  010140                          mov     r1,-(r0)        ; 014: .word  16 ...
003704  162701  000004                  sub     #4,r1           ; 002: .word  0
003710  003373                          bgt     20$             ; 000: .word  2
                                                                ;
                                ;       Launch XXDP monitor     ;
                                                                ;
003712  000167  005072                  jmp     xx$rst          ; system start and restart
```

```
                                   .sbttl  Init date, messages, relocation                         (init)

                                   ;       IN$DAT - Date input

003716  012700  004010'            in$dat: mov     #i$ment,r0      ; ENTER DATE..
003722                                     TypMon                  ;
003724  004767  000314                     call    in$pdt          ; parse date
003730  000401                              br     10$             ; failed
003732  000207                             return                  ; fine
003734  012700  004043'            10$:    mov     #i$mdat,r0      ; INVALID DATE
003740                                     TypMon                  ;
003742  000765                             br      in$dat          ; try, try again

                                   ;       I$M... - Init messages

003744     015     012     122     i$mrad: .asciz  <cr><lf>"RESTART ADDR: "
003747     105     123     124
003752     101     122     124
003755     040     101     104
003760     104     122     072
003763     040     000
003765     015     012     102     i$mboo: .asciz  <cr><lf>"BOOTED VIA UNIT "
003770     117     117     124
003773     105     104     040
003776     126     111     101
004001     040     125     116
004004     111     124     040
004007     000
004010     015     012     105     i$ment: .asciz  <cr><lf>"ENTER DATE (DD-MMM-YY): "
004013     116     124     105
004016     122     040     104
004021     101     124     105
004024     040     050     104
004027     104     055     115
004032     115     115     055
004035     131     131     051
004040     072     040     000
004043     077     040     111     i$mdat: .asciz  "? INVALID DATE"
004046     116     126     101
004051     114     111     104
004054     040     104     101
004057     124     105     000
004062     116     117     116     i$mnon: .asciz  "NON-"
004065     055     000
004067     125     116     111     i$mubs: .asciz  "UNIBUS SYSTEM"<cr><lf>
004072     102     125     123
004075     040     123     131
004100     123     124     105
004103     115     015     012
004106     000
004107     104     117     105     i$mubq: .asciz  "DOES THIS SYSTEM HAVE A UNIBUS? (Y/N CR=Y) "
004112     123     040     124
004115     110     111     123
004120     040     123     131
004123     123     124     105
004126     115     040     110
004131     101     126     105
```

```
004134     040     101     040
004137     125     116     111
004142     102     125     123
004145     077     040     050
004150     131     057     116
004153     040     103     122
004156     075     131     051
004161     040     000
004163     124     110     111  i$mxdp: .ascii  |THIS IS XXDP+.  TYPE "H" OR "H/L" FOR HELP.|<cr><lf>
004166     123     040     111
004171     123     040     130
004174     130     104     120
004177     053     056     040
004202     040     124     131
004205     120     105     040
004210     042     110     042
004213     040     117     122
004216     040     042     110
004221     057     114     042
004224     040     106     117
004227     122     040     110
004232     105     114     120
004235     056     015     012
                                        .even

                                ;       HMDLD0  HMDLD0
                                ;       .bin    CHMDLD0 CHMDKB1 CHMDLB0*
                                ; 5240  0       104000  104001  000040
                                ;               [210|0] [210|1] [0|40]
                                ; 5242  0       0       0       1

004240  000000                          .word   0       ;5240   ;                         (savm)(slip)(note)
004242  000000                          .word   0       ;5242   ;

                                ;       IN$PDT - Parse date
                                ;
                                ;       Convert DD-MON-YY to DOSbatch date format
                                ;
                                ;       date = day-of-year + (year * 1000.)
                                ;
                                ;       day-of-year begins with 1, not 0

004244  005067  007126          in$pdt: clr     s$ydat          ; clear result
004250                                  GetLin                  ; get a response
004252                                  ParFld                  ; parse the first field
004254  000240                           nop                    ;
004256  105710                          tstb    (r0)            ; got a response?
004260  001436                          beq     30$             ; nope - set the default date
                                                                ;
004262                                  RptFld                  ; repeat the field
004264  004767  000142                  call    dt$day          ; and parse it as a day
004270  000443                           br     50$             ;  errors return without a date setup
004272  010067  007100                  mov     r0,s$ydat       ; s$ydat = day (1..31)
004276  004767  000162                  call    dt$mon          ; month
004302  000436                           br     50$             ;  error
004304  010004                          mov     r0,r4           ; r4 = month (0..11)
004306  004767  000310                  call    dt$yea          ; year
```

```
004312  000432                           br     50$             ;  error
004314  010002                          mov     r0,r2           ; r2 = year

                                ;       Calculate the number of days since 1-Jan-70

004316  012700  004402'         10$:    mov     #60$,r0         ; cumulative days in month
004322  066700  007006                  add     s$yrel,r0       ; r0 -> table
004326  006304                          asl     r4              ; make month word offset
004330  060004                          add     r0,r4           ; relocate
004332  011400                          mov     (r4),r0         ; r0 = days in year
004334  060067  007036                  add     r0,s$ydat       ; add to the date
                                                                ; ??? dec r2|bmi 40$|etc
004340  005702                  20$:    tst     r2              ; any more years?
004342  001414                          beq     40$             ; no
004344  062767  001750  007024          add     #1000.,s$ydat   ; yes, add another thousand (^o1750)
004352  005302                          dec     r2              ; loop control
004354  000771                          br      20$             ;

004356  012767  000001  007012  30$:    mov     #1,s$ydat       ; day = 1
004364  005004                          clr     r4              ; month = 0
004366  005002                          clr     r2              ; year = 0
004370  005000                          clr     r0              ; redundant: r0 is set at 10$   (note)
004372  000751                          br      10$             ;

004374  062716  000002          40$:     add    #2,(sp)         ; was a good date
004400  000207                  50$:    return                  ; wasn't a fun evening

                                ;       Data area

004402  000000                  60$:    .word   0
004404  000037                          .word   31.             ; days in january
004406  000073                          .word   59.             ; january + february
004410  000132                          .word   90.             ; etc
004412  000170                          .word   120.
004414  000227                          .word   151.
004416  000265                          .word   181.
004420  000324                          .word   212.
004422  000363                          .word   243.
004424  000421                          .word   273.
004426  000460                          .word   304.
004430  000516                          .word   334.

                                ;       Parse Day
                                ;
                                ; out   r0      day (1:31)

004432                          dt$day: ParDec                  ; get day-in-month
004434  000410                           br     10$             ; error
004436  120127  000055                  cmpb    r1,#'-          ; must be dd-mmm-yy
004442  001007                          bne     20$             ; but isn't
004444  005700                          tst     r0              ; can't be zero
004446  003405                          ble     20$             ; but is
004450  020027  000037                  cmp     r0,#31.         ; cant exceed 31.
004454  003002                          bgt     20$             ; but does
004456  062716  000002          10$:     add    #2,(sp)         ; good
004462  000207                  20$:    return                  ; not good

```

```
                                ;       Parse month
                                ;
                                ; out   r0      month (0:11)

004464                          dt$mon: ParFld                  ; get something
004466  000432                           br     50$             ; that was too much to ask
004470  120127  000055                  cmpb    r1,#'-          ; must have "-" separator
004474  001027                          bne     50$             ; also too much to ask
004476  010002                          mov     r0,r2           ; r0 -> user input
004500                                  movr    #60$,r1,4       ; month names relocated
004504  005000                          clr     r0              ; month-in-year result
004506  121112                  10$:    cmpb    (r1),(r2)       ;
004510  001010                          bne     20$             ;
004512  126162  000001  000001          cmpb    1(r1),1(r2)     ;
004520  001004                          bne     20$             ;
004522  126162  000002  000002          cmpb    2(r1),2(r2)     ;
004530  001407                          beq     40$             ; wow - we found it
004532  005200                  20$:    inc     r0              ; doh - next month, perhaps
004534  062701  000003                  add     #3,r1           ; skip month name
004540  020027  000013                  cmp     r0,#11.         ; more months to come?
004544  003760                  30$:    ble     10$             ; yep
004546  000402                          br      50$             ; way too much to ask
004550  062716  000002          40$:     add    #2,(sp)         ; fine
004554  000207                  50$:    return                  ; fail

                                ;       Month table

004556     112     101     116  60$:    .ascii  "JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC"
004561     106     105     102
004564     115     101     122
004567     101     120     122
004572     115     101     131
004575     112     125     116
004600     112     125     114
004603     101     125     107
004606     123     105     120
004611     117     103     124
004614     116     117     126
004617     104     105     103
                                        .even

                                ;       Parse year
                                ;
                                ; out   r0      year-1970

004622                          dt$yea: ParDec                  ; get a year
004624  000414                           br     10$             ;  out of joint
004626  005701                          tst     r1              ; terminator must EOL
004630  001012                          bne     10$             ;  oh cursed spite
004632  005700                          tst     r0              ; can't be negative
004634  002410                          blt     10$             ;  that I was born
004636  020027  000143                  cmp     r0,#99.         ; 1999 ends our world
004642  003005                          bgt     10$             ;  to set it right
004644  162700  000106                  sub     #70.,r0         ; 1970 starts it
004650  002402                          blt     10$             ;  into the breach once more
004652  062716  000002                   add    #2,(sp)         ; good
004656  000207                  10$:    return
```

```

                                ;       IO$IOB - Relocate IOB                                           (init)

004660                          in$iob: movr    #d$riob,r5,5    ; IOB
                                                                ; driver header in r5?
004664  012702  014614'                 mov     #d$rdis,r2      ; driver function dispatch table
004670  066702  006440                  add     s$yrel,r2       ; relocation constant
004674  012704  000004                  mov     #4,r4           ; counter
004700  011200                  10$:    mov     (r2),r0         ; get a pointer
004702  060700                          add     pc,r0           ; 2-step relocation
004704  162700  004704'                 sub     #.,r0           ;
004710  010022                          mov     r0,(r2)+        ; put it back
004712  005304                          dec     r4              ; do more
004714  001371                          bne     10$             ;

                                ;       Relocate driver IO.UFD and IO.BUF

004716  016500  000010                  mov     io.ufd(r5),r0   ; relocate 10(r5)
004722  060700                          add     pc,r0           ; 2-step relocation
004724  162700  004724'                 sub     #.,r0           ;
004730  010065  000010                  mov     r0,io.ufd(r5)   ; put it back
004734                                  movr    #f$ibuf,dr.buf(r5),11 ; relocated
004742  000207                          return

004744     015     012     103  i$mmon: .asciz  <cr><lf>"CHMDLD0 XXDP+ DL MONITOR"
004747     110     115     104
004752     114     104     060
004755     040     130     130
004760     104     120     053
004763     040     104     114
004766     040     115     117
004771     116     111     124
004774     117     122     000
                                        .even
                                .sbttl  Batch engine                                          (batch)
  005000                        x$xbat:

                                ;       Start of batch area
                                ;
                                ;       The stack pointer below is used for abort/restart
                                ;       And to switch context between batch/interactive modes
                                ;
                                ;       The batch engine is copied to the overlay region for execution.
                                ;       Because of this all it's relative addresses that access
                                ;       the monitor outside it's local area need to be offset.

          004000                        $$ = o$vreg-b$areg;4000 ; offset for relative mode relocation
                                                                ; 6000- 7414 - batch region

                                ;       BA$ENG - Batch engine

  005000                        b$areg:
  005000  000000                b$astk: .word   0               ;\ batch stack
  005002  010667  177772        ba$eng: mov     sp,b$astk       ;/+batch over EPT
  005006  005067  002402                clr     f$isck-$$       ; invalidate batch file checksum

                                ;       Command loop

  005012  010700                ba$cmd: mov     pc,r0
  005014  062700  000022                add     #ba$abt-.,r0    ;
  005020                                SetAbt  ;ba$abt         ; generic batch command abort
  005022                                GetLin                  ; get something
  005024  103001                         bcc    bc$qut          ; a whole lot of nothing
  005026  000473                        br      ba$dis          ; dispatch command


                                ;       Batch QUIT command
                                ;
                                ;       Cancels batch file and returns to CLI

  005030  016706  177744        bc$qut: mov     b$astk,sp       ; nix - restore stack
  005034  000207                        return


                                ;       BA$ABT - Batch abort
                                ;
                                ;       Abort routine (from set abort above)

  005036  005700                ba$abt: tst     r0              ; got an abort message?
  005040  001404                        beq     30$             ; nope
  005042  060700                        add     pc,r0           ; relocate
  005044  162700  005044'               sub     #.,r0           ;
  005050                                TypBrk                  ; breakthrough message
  005052  000766                30$:    br      bc$qut          ; quit batch


                                ;       Batch dispatch table
                                ;
                                ;       Batch command routine pointers
                                ;       The dispatcher adds the constant 6250 to form addresses
                                ;
                                ;       SMI CMI L S R E C
                                ;       GOTO WAIT QUIET PRINT
                                ;       END QUIT IFLMD IFERR IF

 005054  012274'                b$adis: .word   bc$smi+$$  ; SMI
 005056  012304'                        .word   bc$cmi+$$  ; CMI
 005060  011362'                        .word   bc$loa+$$  ; Load
 005062  011446'                        .word   bc$sta+$$  ; Start
 005064  011670'                        .word   bc$run+$$  ; Run
 005066  012366'                        .word   bc$enb+$$  ; Enable
 005070  011714'                        .word   bc$chn+$$  ; Chain
 005072  011742'                        .word   bc$gto+$$  ; Goto
 005074  012062'                        .word   bc$wai+$$  ; Wait
 005076  012104'                        .word   bc$qui+$$  ; Quiet
 005100  012112'                        .word   bc$prt+$$  ; Print
 005102  012212'                        .word   bc$end+$$  ; End
 005104  011030'                        .word   bc$qut+$$  ; Quit
 005106  012244'                        .word   bc$ilm+$$  ; Iflmd
 005110  012264'                        .word   bc$ier+$$  ; Iferr
 005112  012146'                        .word   bc$ift+$$  ; If
 005114  000000                         .word   0

 005116     123     115     111  b$aloo: .asciz  "SMI"
 005121     000
 005122     103     115     111          .asciz  "CMI"
 005125     000
 005126     114     000                  .asciz  "L"
 005130     123     000                  .asciz  "S"
 005132     122     000                  .asciz  "R"
 005134     105     000                  .asciz  "E"
 005136     103     000                  .asciz  "C"
 005140     107     117     124          .asciz  "GOTO"
 005143     117     000
 005145     127     101     111          .asciz  "WAIT"
 005150     124     000
 005152     121     125     111          .asciz  "QUIET"
 005155     105     124     000
 005160     120     122     111          .asciz  "PRINT"
 005163     116     124     000
 005166     105     116     104          .asciz  "END"
 005171     000
 005172     121     125     111          .asciz  "QUIT"
 005175     124     000
 005177     111     106     114          .asciz  "IFLMD"
 005202     115     104     000
 005205     111     106     105          .asciz  "IFERR"
 005210     122     122     000
 005213     111     106     000          .asciz  "IF"
                                         .even

                                ;       BA$DIS - Dispatch batch command

 005216                         ba$dis: ParFld                  ; get a command name
 005220  000240                          nop                    ; ignore errors
 005222  105710                         tstb    (r0)            ; empty line?
 005224  001452                         beq     60$             ; yes
 005226  121027  000073                 cmpb    (r0),#';        ; comment?
```

```
005232  001447                          beq     60$             ; yes
005234  120127  000072                  cmpb    r1,#':          ; label: ?
005240  001444                          beq     60$             ; yes

                                ;       r0 ->   field
                                ;       r1 =    terminator
                                ;       r2 ->   b$aloo
                                ;       r4 ->   b$adis

005242  012704  177604                  mov     #b$adis-5$,r4   ; r4 -> dispatch table
005246  060704                          add     pc,r4           ;
005250  012702  177640          5$:     mov     #b$aloo-10$,r2  ; r2 -> lookup table
005254  060702                          add     pc,r2           ;
005256  010003                  10$:    mov     r0,r3           ; r0/r3 -> field
005260  120163  177777          20$:    cmpb    r1,-1(r3)       ; hit terminator?
005264  001402                          beq     30$             ; yes
005266  122322                          cmpb    (r3)+,(r2)+     ; skip until we do
005270  001773                          beq     20$             ;
005272  105762  177777          30$:    tstb    -1(r2)          ; end of lookup table entry?
005276  001007                          bne     40$             ; no - do the next
005300  105763  177777                  tstb    -1(r3)          ; also the end of the field?
005304  001414                          beq     50$             ; yes - we have a command
005306  126327  177777  000040          cmpb    -1(r3),#space   ; space is also field end
005314  001410                          beq     50$             ; we have a command
005316  105722                  40$:    tstb    (r2)+           ; skip remainder of name
005320  001376                          bne     40$             ;
005322  005724                          tst     (r4)+           ; more lookup entries to come?
005324  001354                          bne     10$             ; yep
005326  012700  005354'                 mov     #m$sber,r0      ; no - Batch "?ER"
005332                                  TypMon                  ;
005334  000406                          br      60$             ; wind back to batch engine

                                ;       Dispatch batch command

005336  011404                  50$:    mov     (r4),r4         ; r4 -> dispatch entry
005340  066704  001770                  add     s$yrel-$$,r4    ; relocate
                                                                ;
005344  004714                          call    (r4)            ; call command
                                                                ;
005346  004767  001346                  call    mo$rst-$$       ; restore monitor
005352  000617                  60$:    br      ba$cmd          ; get another command (again)

005354     077     105     122  m$sber: .asciz  "?ER"<cr><lf>   ; Batch "?ER" (the only batch error message)
005357     015     012     000
```

```
                                .sbttl  Load Start Run Chain Goto Wait Quiet Print              (batch)

                                ;       Batch LOAD file command
                                ;
                                ;       L filespec
                                ;
                                ;       bc$loa  Batch Load command EPT
                                ;       bu$loa  Batch Run command EPT

005362  105267  002032          bc$loa: incb    s$yloa-$$       ; LOAD in-progress (not RUN)
005366                          bu$loa: ParFld                  ; get "FILNAM"
005370  000425                           br     20$             ; error
005372  010046                          mov     r0,-(sp)        ; save field start
005374  122001                  10$:    cmpb    (r0)+,r1        ; hunt for terminator in r1
005376  001376                          bne     10$             ; until end of string
005400  112760  000056  177777          movb    #'.,-1(r0)      ; replace with "."
005406  112720  000102                  movb    #'B,(r0)+       ; add "BIC"
005412  112720  000111                  movb    #'I,(r0)+       ;
005416  112720  000103                  movb    #'C,(r0)+       ;
005422  105010                          clrb    (r0)            ; terminate string
005424  012600                          mov     (sp)+,r0        ; get field address back
005426  062767  000004  001730          add     #4,c$lnxt-$$    ; add four to the line end pointer
                                                                ; to accommodate the added ".BIC"
                                                                ; r0 -> filespec
005434  005001                          clr     r1              ; r1 = base address
005436                                  LoaFil                  ; load image
005440  105067  001754                  clrb    s$yloa-$$       ; clear LOAD in-progress flag
005444  000207                  20$:    return


                                ;       Batch START program command
                                ;
                                ;       S [/repeat] [address]

005446  004767  000002          bc$sta: call    bu$sta          ; get the start address
005452  000425                          br      bu$act          ; activate

                                ;       BU$STA - Get start address and repeat count for batch Run and Start
                                ;
                                ;       S 200           start at 200
                                ;       S/5             repeat five times
                                ;       S/5 200         start 200, repeat five times
                                ;
                                ;       Odd start addresses are silently rejected               (note)

005454  012767  000001  001724  bu$sta: mov     #1,s$ysta-$$    ; default start state
005462  012767  000001  001646          mov     #1,s$yrpt-$$    ; default repeat count
005470  020127  000057                  cmp     r1,#'/          ; decimal switch?
005474  001004                          bne     10$             ; nope, octal address
005476                                  ParDec                  ; translate decimal
005500  000411                           br     20$             ; oops
005502  010067  001630                  mov     r0,s$yrpt-$$    ; store repeat count, check for start address
                                                                ;
005506                          10$:    ParOct                  ; get the octal start address
005510  000405                           br     20$             ; no such luck
005512  032700  000001                  bit     #1,r0           ; odd address?
005516  001002                          bne     20$             ; we're not all LSI's you know
```

```
005520  010067  001662                  mov     r0,s$ysta-$$    ; start/load address
005524  000207                  20$:    return


                                ;       BU$ACT - Batch Start/Run program

                                        .enabl  lsb
005526  022767  000001  001652  bu$act: cmp     #1,s$ysta-$$;   ; LDA?
005534  001404                          beq     10$             ; yes
005536  016767  001644  001644          mov     s$ysta-$$,s$yact-$$ ; image activate address
005544  000407                          br      20$             ;
005546  022767  000001  001634  10$:    cmp     #1,s$yact-$$    ; activate address likewise #1
005554  001003                          bne     20$             ; no - has something real
005556  012767  000200  001624          mov     #200,s$yact-$$  ; yes -default to @#200 activation address

                                ;       Configure low-memory syscom

005564                          20$:    GetDev                  ; r0 ->
005566  116001  000002                  movb    dv.uni(r0),r1   ; get the ascii unit number
005572  162701  000060                  sub     #'0,r1          ; make a digit
005576  110137  000040                  movb    r1,@#40         ; 40: unit number
005602  116037  000003  000041          movb    dv.med(r0),@#41 ; 41: device media code
005610  016737  001566  000030          mov     s$yemt-$$,@#30  ; 30: emt vector
005616  016737  001562  000032          mov     s$yemt+2-$$,@#32; 32: ditto

005624  010746                          mov     pc,-(sp)
005626  062716  000466                  add     #bu$ret-.,(sp)  ; image return path
005632  012637  000042                  mov     (sp)+,@#42      ; 42: -> app return path
005636  010667  000024                  mov     sp,30$          ; save stack
                                                                ;
                                ;       Activate batch program  ;
                                                                ;
005642  004777  001542                  call    @s$yact-$$      ; call ye app
                                                                ;
005646  016706  000014          bu$exi: mov     30$,sp          ; restore stack
005652  004767  000472                  call    bu$pr7          ; back to PR7
005656  004767  000612                  call    em$rst-$$       ; rebuild emt vector
005662  005000                          clr     r0              ; r0=0 status
005664  000207                          return
005666  000000                  30$:    .word   0
                                        .dsabl  lsb


                                ;       Batch RUN command
                                ;
                                ;       R file[/repeat][address]

005670                          bc$run: ParFld                  ; get a filespec
005672  000207                           return                 ; filename error
005674  010046                          mov     r0,-(sp)        ; save field
005676  004767  177552                  call    bu$sta          ; get a start address or repeat count
005702  012667  001456                  mov     (sp)+,c$lnxt-$$ ; restore field
005706  004767  177454                  call    bu$loa          ; load image
005712  000705                          br      bu$act          ; activate


                                ;       Batch CHAIN command
                                ;
                                ;       C filespec [/switches]
                                ;
                                ;       Batch supports one level of chain nesting
                                ;       No checks are made to see if additional nesting takes place
                                ;       More accurately, batch allows a single return level
                                ;
                                ;       cl$chn appends ".CCC" to the "filnam" but bc$chn does not   (note)

005714                          bc$chn: ParFld                  ; get a filespec
005716  000410                           br     10$             ; fail
005720                                  PshBat                  ; up chain level and open batch file
005722  016746  177052                  mov     b$astk,-(sp)    ; save the current batch stack
005726  004767  177050                  call    ba$eng          ; call the batch engine
005732  012667  177042                  mov     (sp)+,b$astk    ; restore our stack
005736                                  PopBat                  ; pop nest chain file, return to prior
005740  000207                  10$:    return


                                ;       Batch GOTO command
                                ;
                                ;       GOTO label
                                ;
                                ;       label
                                ;
                                ;       Will GOTO will loop forever if a label is not found?    (note)

005742  012702  013140'         bc$gto: mov     #s$ygto,r2      ; batch goto buffer
005746  066702  001362                  add     s$yrel-$$,r2    ; relocate
005752  010204                          mov     r2,r4           ; r2/r4 -> buffer
005754                                  ParFld                  ; get the goto label
005756  000440                           br     70$             ; wrong - just return
005760  112024                  10$:    movb    (r0)+,(r4)+     ; copy field to buffer
005762  121001                          cmpb    (r0),r1         ; until we see the terminator
005764  001375                          bne     10$             ;
005766  105014                          clrb    (r4)            ; zero the end of the buffer string
005770  105367  001433          20$:    decb    s$yqui-$$       ; mute quiet mode for messages
005774                          30$:    GetLin                  ; search forward for the label
005776  103404                           bcs    40$             ; got a good line
                                                                ;
006000  005067  001362                  clr     f$ipos-$$       ; hit EOF - rewind batch file
006004  005067  001404                  clr     f$isck-$$       ; clear save checksum to force read
                                                                ; ParFld fails back to 30$ for us
006010                          40$:    ParFld                  ; get the first field on the line
006012  000770                           br     30$             ; no field
006014  120127  000072                  cmpb    r1,#':          ; LABEL: ?
006020  001365                          bne     30$             ; no - get next line
006022  010204                          mov     r2,r4           ; r4 -> target label
006024  122420                  50$:    cmpb    (r4)+,(r0)+     ; r0 -> candidate label
006026  001776                          beq     50$             ; still fits
006030  126001  177777                  cmpb    -1(r0),r1       ; did candidate end with ":"?
006034  001357                          bne     30$             ; no
006036  126427  177777  000057          cmpb    -1(r4),#'/      ; did target end with "/"?
006044  001403                          beq     60$             ; yes
006046  105764  177777                  tstb    -1(r4)          ; did target hit end of line?
006052  001350                          bne     30$             ; no - try the next chain file line
006054  105267  001347          60$:    incb    s$yqui-$$       ; put that back how we found it
```

```
006060  000207                  70$:    return                  ; we are located at the label


                                ;       Batch WAIT command
                                ;
                                ;       WAIT
                                ;       ...
                                ;       [ctrl/x]

006062                          bc$wai: GetAvl                  ; wait for operator ^X
006064  105067  001335                  clrb    s$ypnd-$$       ; no pending character
006070  004767  174720                  call    te$ctc-$$       ; check ctrl/c
006074  120027  000030                  cmpb    r0,#ctrlx       ; ctrl/x?
006100  001370                          bne     bc$wai          ; nope
006102  000406                          br      bu$new          ; newline and out


                                ;       Batch QUIET command

006104  105167  001317          bc$qui: comb    s$yqui-$$       ; flip the quiet flop
006110  000207                          return


                                ;       Batch PRINT command

006112                          bc$prt: ParFld                  ; parse anything into the field
006114  000401                           br     bu$new          ; nothing - so just newline
006116                                  TypBrk                  ; r0 -> display message
006120                                  fall    bu$new          ; add newline

006120  012700  003726'         bu$new: mov     #t$enew-$$,r0   ; .byte cr,lf,0
006124  060700                          add     pc,r0           ; double
006126  162700  006126'                 sub     #.,r0           ; relocation
006132                                  TypBrk                  ; display message
006134  000207                          return
```

```
                                .sbttl  If IfLMD IfERR CMI SMI Enable                           (batch)

                                ;       No terminator separates "THEN" from "END"               (note)

006136     124     110     105  b$athn: .ascii  "THEN"
006141     116
006142     105     116     104  b$aend: .ascii  "END"<377>
006145     377


                                ;       BATCH IF command
                                ;
                                ;       IF <switch> THEN
                                ;       ...
                                ;       END

006146                          bc$ift: ParFld                  ; r0 -> condition switch
006150  000434                           br     bu$nop          ; no hope
006152  004767  000242                  call    cu$swi-$$       ; parse chain condition
006156  103413                          bcs     bu$fal          ; false
006160                                  fall    bu$tru          ;


                                ;       BU$TRU - Condition true - gobble "THEN"
                                ;
                                ;       B$ATHN actually points to "THENEND"                     (note)

006160                          bu$tru: ParFld                  ; get yet another field
006162  000424                           br     bu$err          ; bummer
006164  012702  177744                  mov     #b$athn-10$,r2  ; THEN
006170  060702                          add     pc,r2           ;
006172  122022                  10$:    cmpb    (r0)+,(r2)+     ; shall I compare thee?
006174  001776                          beq     10$             ; to a summer's "THENEND"
006176  105760  177777                  tstb    -1(r0)          ; did we complete?
006202  001014                          bne     bu$err          ; nope - bummer
006204  000416                          br      bu$nop          ; yep

                                ;       BU$FAL - Condition false - search for "END"

006206                          bu$fal: GetLin                  ; search chain file for "END"
006210  103014                           bcc    bu$nop          ; EOF - we're all done
006212                                  fall    bc$end          ;

                                ;       Batch END command

006212  012702  177722          bc$end: mov     #b$aend-10$,r2  ; END
006216  060702                          add     pc,r2
006220  122022                  10$:    cmpb    (r0)+,(r2)+     ; compare strings
006222  001776                          beq     10$             ; while they match
006224  105760  177777                  tstb    -1(r0)          ; source completed?
006230  001366                          bne     bu$fal          ; no - keep searching
006232  000403                          br      bu$nop          ; found - return
006234  012700  005354'         bu$err: mov     #m$sber,r0      ; batch "?ER"
006240                                  TypMon                  ;
006242  000207                  bu$nop: return                  ; True
```

```
                                ;       Batch IFLMD (If Low Media) command
                                ;
                                ;       IFLMD <media code> THEN
                                ;       ...
                                ;       END

006244                          bc$ilm: ParOct                  ; get the media code
006246  000775                           br     bu$nop          ; just return for error
006250  016701  001064                  mov     s$ydev-$$,r1    ; r1 -> device info
006254  120061  000003                  cmpb    r0,dv.med(r1)   ; media match?
006260  001737                          beq     bu$tru          ; true
006262  000751                          br      bu$fal          ; false


                                ;       Batch IFERR (If Error) command
                                ;
                                ;       IFERR THEN
                                ;       ...
                                ;       END

006264  005767  002306          bc$ier: tst     s$yerr-$$       ; has a program reported an error?
006270  001333                          bne     bu$tru          ; true - yes
006272  000745                          br      bu$fal          ; false


                                ;       Batch SMI (Set Manual Intervention) command

006274  052737  000001  000052  bc$smi: bis     #scMAN$,@#52    ; SMI - Set manual
006302  000207                          return


                                ;       Batch CMI (Clear Manual Intervention) command

006304  042737  000001  000052  bc$cmi: bic     #scMAN$,@#52    ; CMI - Clear manual
006312  000207                          return


                                ;       BU$RET - Batch managed image exit
                                ;
                                ;       BU$RET has two cases and three paths:
                                ;
                                ;       42=0    Utility exit
                                ;       42!=0   Diagnostic exit
                                ;
                                ;       Batch apps return here via @#42 with the protocol below:
                                ;       In the initialization section @#42 is set to $endad.
                                ;
                                ;       .=42    .word   $endad  ;;set loc.46 to address of $endad in .$eop
                                ;       .=52    .word   0       ;;
                                ;
                                ;       endpas: mov     @#42,r0 ;; get monitor address
                                ;               beq     $doagn  ;; branch if no monitor
                                ;               reset           ;; clear the world
                                ;       $endad: call    (r0)    ;; goto monitor (or loop forever)
                                ;               nop             ;; save room
                                ;               nop             ;; for
                                ;               nop             ;;  act11
                                ;       $doagn: jmp     @(pc)+  ;; return
                                ;       $rtnad: .word   rstart  ;;


006314  004767  000030          bu$ret: call    bu$pr7          ; PR7
006320  005367  001012                  dec     s$yrpt-$$;      ; all iterations done (usually just one)?
006324  001407                          beq     10$             ; yes - start over
006326  005737  000042                  tst     @#42            ; diagnostic or utility?
006332  001404                          beq     10$             ; utility - exit and start over
006334  005767  002230                  tst     s$yqvs-$$       ; /QV quick verify switch?
006340  001001                          bne     10$             ; yes - start over
                                                                ;
                                ;       Return for another pass ;
                                                                ;
006342  000207                          return                  ; repeat app
                                                                ;
006344  000167  177276          10$:    jmp     bu$exi          ; exit image


                                ;       BU$PR7 - Set PR7 with RTI
                                ;
                                ;       Classic kernel-mode SPL 7 routine

006350  012746  000340          bu$pr7: mov     #340,-(sp)      ; the PSW to be
006354  012746  000002                  mov     #20$-10$,-(sp)  ;  and
006360  060716                          add     pc,(sp)         ;   the PC to be
006362  000002                  10$:    rti                     ; I now prounounce thee...
006364  000207                  20$:    return                  ; what! where'd they go?


                                ;       Batch ENABLE command
                                ;
                                ;       E <unit number>
                                ;
                                ;       @dr.dev (dr$dev) updates the device ascii unit (dv.uni)
                                ;       The CLI code for this command is identical

006366                          bc$enb: ParOct                  ; get a unit number
006370  000410                           br     10$             ; failed
006372  110067  006226                  movb    r0,d$runi       ; store new unit number
006376  012705  014626'                 mov     #d$riob,r5      ;
006402  066705  004726                  add     s$yrel,r5       ; IOB
006406  004775  177774                  call    @dr.dev(r5)     ; update driver
006412  000207                  10$:    return

        001414                          balen. = .-b$areg       ; length of batch overlay

006414                                  .blkw   172     ;122.   ; round-up to 10000
```

```
                                .sbttl  Terminal                                             (terminal)
   007000                       x$xtra:

                                ;       TE$PUT - Display a single character
                                ;
                                ;       in      r0 =    character
                                ;
                                ;       out     r0 =    character

   007000  105777  004340       te$put: tstb    @s$ytps         ; TPS ready?
   007004  100375                       bpl     te$put          ; not yet
   007006  110077  004334               movb    r0,@s$ytpb      ; out damned spot
   007012  000207                       return


                                ;       TE$CTC - Check ctrl/c
                                ;
                                ;       in      r0 =    character to check
                                ;
                                ;               call    te$ctc
                                ;
                                ;       true    display "^C" and abort without an r0 message
                                ;       false   return

                                        .enabl  lsb
   007014  120027  000003       te$ctc: cmpb    r0,#ctrlc       ; ^C - are you looking at me?
   007020  001042                       bne     40$             ; no  - return
   007022  000416                       br      10$             ; yes - display


                                ;       TE$CTL - Check control key
                                ;
                                ;       in      r0 =    character
                                ;
                                ;               call    te$ctl
                                ;       false   bcc     is not a control key
                                ;       true    bcs     is a control key
                                ;
                                ;       out     r0 =    character (whether true or false)
                                ;               "^x"    if control key and not null,tab,^Q,^S
                                ;
                                ;       abort   "^C"    if ctrl/c
                                ;
                                ;       DRS control keys:
                                ;
                                ;       ctrl/c  Temporarily halt a DRS batch file
                                ;       ctrl/z  Terminate a DRS batch file

   007024  105700               te$ctl: tstb    r0              ; ^@ - null (GetCmd EPT)
   007026  001436                       beq     30$             ;
   007030  120027  000032               cmpb    r0,#ctrlz       ; ^Z - Terminate DRS batch file
   007034  003033                       bgt     30$             ;
   007036  120027  000011               cmpb    r0,#ht          ; ^I - tab
   007042  001430                       beq     30$             ;
   007044  120027  000021               cmpb    r0,#ctrlq       ; ^Q - resume output
   007050  001425                       beq     30$             ;
   007052  120027  000023               cmpb    r0,#ctrls       ; ^S - pause output
```

```
007056  001422                          beq     30$             ;

                                ;       Put control character

007060  105067  004341          10$:    clrb    s$ypnd          ; clear character pending
007064  010046                          mov     r0,-(sp)        ; save character
007066  112700  000136                  movb    #'^,r0          ; "^"
007072                                  PutChk                  ; out
007074  011600                          mov     (sp),r0         ; control code
007076  052700  000100                  bis     #100,r0         ; ascii letter
007102                                  PutChk                  ; "^C"
007104  012600                          mov     (sp)+,r0        ; get the code back
                                                                ;
007106  120027  000003                  cmpb    r0,#ctrlc       ; ctrl/c?
007112  001002                          bne     20$             ; nope
007114  005000                          clr     r0              ; no message
007116                                  JmpAbt                  ; we're done here
                                                                ;
007120  000241                  20$:    clc                     ; false - not a control key
007122  000401                          br      40$             ;
007124  000261                  30$:    sec                     ; true - fine
007126  000207                  40$:    return                  ;
                                        .dsabl  lsb
                                .sbttl  GetLin, ParFld                                          (EMT)

                                ;       GetLin - Get Command Line service                       (EMT 0)
                                ;
                                ;       GetLin inputs a terminal or batch file command line.
                                ;       GetLin restores the batch file input buffer and position if needed
                                ;       PopBat triggers a buffer restore to return to a prior batch file
                                ;
                                ;               GetLin
                                ;                bcc    EOF             ; batch EOF
                                ;
                                ;       out     r0 ->   command line
                                ;               c$lnxt->command line
                                ;
                                ;       fail    abort   "? RD ERR"      ; batch file read error

                                        .enabl  lsb
007130  105767  005436          GetLin: tstb    s$ybat          ; in a chain file?
007134  001425                          beq     20$             ; nope
                                                                ;
                                ;       Check batch context     ;
                                                                ;
007136  012705  014626'                 mov     #d$riob,r5      ; r5 -> IOB
007142  066705  004166                  add     s$yrel,r5       ; relocate it
007146  004767  006264                  call    rb$chk          ; checksum the file block
007152  026767  004236  004232          cmp     f$isck,f$irck   ; has it changed behind our backs?
007160  001413                          beq     20$             ; nope - pristine
                                                                ;
                                ;       Restore batch context   ;
                                                                ;
007162  012700  013520'                 mov     #b$afnm,r0      ; r0 -> file spec
007166  066700  004142                  add     s$yrel,r0       ; relocate it
007172                                  OpnFil                  ; open sesame
007174  016704  004166                  mov     f$ipos,r4       ; current file position
                                                                ;
007200  005304                  10$:    dec     r4              ; advance to the current file location
007202  100402                          bmi     20$             ; one byte at a time
007204                                  ReaByt                  ; errors abort
007206  000774                          br      10$             ;

                                ;       Common CLI/batch stream

007210  005003                  20$:    clr     r3              ; rubout comes back to here
007212  016702  004142                  mov     c$llin,r2       ; r2 -> line
007216  010267  004142                  mov     r2,c$lnxt       ; r2 -> first/next field
007222                          30$:    GetChk                  ; get another character
007224  103071                           bcc    110$            ; some error
007226  105700                          tstb    r0              ; got anything?
007230  001774                          beq     30$             ; a null
                                                                ;
007232  120027  000012                  cmpb    r0,#lf          ; linefeed
007236  001437                          beq     70$             ;
007240  120027  000015                  cmpb    r0,#cr          ; carriage return
007244  001435                          beq     80$             ;
007246  004767  177552                  call    te$ctl          ; control character?
007252  103031                          bcc     70$             ; yes
007254  120027  000177                  cmpb    r0,#del         ; rubout
```

```
007260  001422                          beq     60$             ;
                                                                ; uppercase conversion
007262  120027  000141                  cmpb    r0,#'a          ; lowercase?
007266  103405                          bcs     40$             ; nope
007270  120027  000172                  cmpb    r0,#'z          ; really lowercase?
007274  101002                          bhi     40$             ; nope
007276  162700  000040                  sub     #40,r0          ; make it uppercase
007302  020367  004054          40$:    cmp     r3,c$llen       ; at end of buffer?
007306  002002                          bge     50$             ; yes - echo and forget hack    (note)
007310  005203                          inc     r3              ; nope - advance
007312  110022                          movb    r0,(r2)+        ; and store
007314  105767  004107          50$:    tstb    s$yqui          ; quietly?
007320  100740                          bmi     30$             ; yes
007322                                  PutChk                  ; no - echo character
007324  000736                          br      30$             ; and get another
                                                                ;
007326  005303                  60$:    dec     r3              ; delete/rubout
007330  100727                          bmi     20$             ; too far - restart line
007332  114200                          movb    -(r2),r0        ; echo erased character
007334  000767                          br      50$             ; echo and get next

                                ;       End of line

007336  110022                  70$:    movb    r0,(r2)+        ; lf
007340  105022                  80$:    clrb    (r2)+           ; cr
007342  010001                          mov     r0,r1           ; save character
007344                                  NewLin                  ; newline
007346  120127  000025                  cmpb    r1,#ctrlu       ; delete line?
007352  001716                          beq     20$             ; yes - start over

                                ;       In batch mode:
                                ;
                                ;       r1=cr   gobble succeeding lf
                                ;       r1=lf   thus never occurs in batch mode
                                ;
                                ;       Save the read block checksum at each line end

007354  105767  005212                  tstb    s$ybat          ; in batch mode?
007360  001406                          beq     90$             ; no
007362  005267  004000                  inc     f$ipos          ; yes - skip the lf position
007366                                  ReaByt                  ; and gobble the lf byte
                                                                ;
                                ;       Accumulate the batch file checksum
                                ;
                                ;       ReaBlk checksum (f$irck) is copied to f$isck at each end of line.
                                ;       Why? Because GetLin has no way of knowing when a new block
                                ;       has been read in, however it does know that f$irck always has
                                ;       a valid block checksum.

007370  016767  004016  004016          mov     f$irck,f$isck   ; save the block checksum

007376  016700  003762          90$:    mov     c$lnxt,r0       ; fine - r0 -> start of line (first field)
007402  052766  000001  000012  gl$sec: bis     #cbit,sp.ps+2(sp); set return cbit (GetChk branches here)
007410  000207                  110$:   return
                                        .dsabl  lsb


                                        ;       ParFld - Parse Field service                            (EMT 1)
                                        ;
                                        ;               ParFld
                                        ;       fail     br     EOL     ; end-of-command reached
                                        ;         or     nop            ; @r0=0 used as EOL test
                                        ;
                                        ;       fine    r0 ->   field line segment
                                        ;               r0->0   EOL
                                        ;               r1 =    terminator
                                        ;
                                        ;       abort   r0 ->   "? Er"

                                                .enabl  lsb
    007412  016700  003746          ParFld: mov     c$lnxt,r0       ; current line position
    007416  105760  177777                  tstb    -1(r0)          ; past EOL?
    007422  001423                          beq     60$             ; yes
    007424  112001                  10$:    movb    (r0)+,r1        ; next line character
    007426  001410                          beq     40$             ; are no more
    007430  010703                          mov     pc,r3           ;
    007432  062703  000042          20$:    add     #70$-20$,r3     ; terminators

    007436  120123                  30$:    cmpb    r1,(r3)+        ; this a terminator?
    007440  001403                          beq     40$             ; yes
    007442  105713                          tstb    (r3)            ; got more to come?
    007444  001374                          bne     30$             ; yes
    007446  000766                          br      10$             ; no - look at next line character

                                        ;       Good return

    007450  016767  003710  003714  40$:    mov     c$lnxt,c$lfld   ; save field starting point
    007456  010067  003702                  mov     r0,c$lnxt       ; setup new field
    007462  016700  003704                  mov     c$lfld,r0
    007466                          gf$sec:                         ; gf$sec called by ParOct
    007466  062716  000002          50$:     add    #2,(sp)         ; good return
    007472  000207                  60$:    return

    007474     040     057     072  70$:    .asciz  " /:-=<"<ht>    ; terminator list
    007477     055     075     074
    007502     011     000
                                                .even
                                                .dsabl  lsb
```

```
                                .sbttl  TypMon PutStr TypBrk PutChk GetAvl GetChk NewLin PutTab (EMT)

                                ;       TypMon - Type monitor message service                   (EMT 2)
                                ;
                                ;       in      r0 ->   message
                                ;
                                ;               TypMon

007504  060700                  TypMon: add     pc,r0
007506  162700  007506'         10$:    sub     #10$,r0         ; relocate monitor string
007512                                  fall    TypMsg


                                ;       TypMsg - Type message service                           (EMT 3)
                                ;
                                ;       zero terminates

007512  105767  003711          TypMsg: tstb    s$yqui          ; are we quiet?
007516  100405                          bmi     20$             ; yes - ignore this
007520  010002                          mov     r0,r2           ; make a pointer
007522  112200                  10$:    movb    (r2)+,r0        ; another
007524  001402                          beq     20$             ; done
007526                                  PutChk                  ; out it goes
007530  000774                          br      10$             ; more
007532  000207                  20$:    return


                                ;       TypBrk - Type Breakthrough message service              (EMT 44)
                                ;
                                ;       Breakthrough type
                                ;       Display message even in quiet mode

007534  116746  003667          TypBrk: movb    s$yqui,-(sp)    ; save quiet mode flag
007540  105067  003663                  clrb    s$yqui          ; switch off quiet mode
007544                                  TypMsg                  ; tell the world
007546  112667  003655                  movb    (sp)+,s$yqui    ; restore quiet mode flag
007552  000207                          return


                                ;       PutChk - Put Character and check for ctrl/c service     (EMT 4)
                                ;
                                ;       in      r0      character
                                ;
                                ;               PutChk
                                ;
                                ;       abort   ctrl/c

007554  004767  005542          PutChk: call    PutCha          ; output char and check keyboard
007560  004767  177230                  call    te$ctc          ; check ctrl/c
007564  000207                          return

                                ;
                                ;       GetAvl - Get Available character service                (EMT 5)
                                ;
                                ;       fine    r0      character
                                ;               s$ypnd  character as pending
                                ;
```

```
                               ;       GetAvl is followed by GetChk to gobble the pending character
                               ;       GetAvl is also called by GetChk and PutCha

007566  010146                 GetAvl: mov     r1,-(sp)        ; save r1
007570  105737  177560                 tstb    @#TKS           ; are we relevant?
007574  100026                         bpl     40$             ; apparently not
007576  105046                         clrb    -(sp)           ; result character
007600  005000                         clr     r0              ;  a flag
007602  010001                 10$:    mov     r0,r1           ; r1=ctrls => loop
007604  105737  177560         20$:    tstb    @#TKS           ; who is waiting for us?
007610  100375                         bpl     20$             ; nobody
007612  113700  177562                 movb    @#TKB,r0        ; the good old TKB
007616  042700  177600                 bic     #^c177,r0       ; 7 bits only
007622  120027  000021                 cmpb    r0,#ctrlq       ; ^Q - continue output
007626  001406                         beq     30$             ; yes - done
007630  120027  000023                 cmpb    r0,#ctrls       ; ^S?
007634  001762                         beq     10$             ; yes - wait for ctrl/q
007636  110016                         movb    r0,(sp)         ; save anything else
007640  105701                         tstb    r1              ; seen ctrl/s?
007642  001360                         bne     20$             ; yes - wait for ctrl/q
007644  111667  003555         30$:    movb    (sp),s$ypnd     ; pending input character
007650  112600                         movb    (sp)+,r0        ; return it in r0
007652  012601                 40$:    mov     (sp)+,r1        ; restore that
007654  000207                         return


                               ;       GetChk - Get character, check ctrl/c                    (EMT 6)
                               ;
                               ;               GetChk
                               ;       fail    bcc     EOF     ; batch EOF only
                               ;
                               ;       fine    r0 =    char
                               ;
                               ;       abort   "Rd Er"         ; batch file read error

007656  105767  004710         GetChk: tstb    s$ybat          ; batch?
007662  001405                         beq     10$             ; nope
                                                               ; batch
007664                                 ReaByt                  ; get yet another
007666  103016                          bcc    50$             ; failed
007670  005267  003472                 inc     f$ipos          ; count it
007674  000412                         br      40$             ; return
                                                               ; keyboard
007676  116700  003523         10$:    movb    s$ypnd,r0       ; got pending input character?
007702  001005                         bne     30$             ; yes - use that
                                                               ;
                               ;       Keyboard spin loop      ;
                                                               ;
007704                         20$:    GetAvl                  ; get available
007706  005700                         tst     r0              ; got nothing
007710  001775                         beq     20$             ; loop until we do
                                                               ;
007712  004767  177076                 call    te$ctc          ; check ctrl/c
007716  105067  003503         30$:    clrb    s$ypnd          ; pend no more
007722  000627                 40$:    br      gl$sec          ; fine: EMT c=1 (GetLin gl$sec sets carry)
007724  000207                 50$:    return                  ; fail: EMT c=0 (batch EOF only)

```

```

                               ;       NewLin - NewLine service                                (EMT 7)

007726     015     012     000  t$enew: .byte   cr,lf,0,0       ; also used by bu$new
007731     000

007732  012700  007726'         NewLin: mov     #t$enew,r0      ; newline string
007736                                  TypMon                  ; out, relocated
007740  000207                          return                  ; life can be easy sometimes


                               ;       PutTab service                                          (EMT 10)
                               ;
                               ;       Advances to next tab stop

007742  112700  000040          PutTab: movb    #space,r0       ; a space
007746                                  PutChk                  ; output
007750  132767  000007  003446          bitb    #7,s$ycol       ; check the column
007756  001371                          bne     PutTab          ; more columns, more columns
007760  000207                          return
```

```
                                .sbttl  ParOct OpnFil                                           (EMT)

                                ;       ParOct - Parse octal service                            (EMT 11)
                                ;
                                ;       in      field   "12345"
                                ;
                                ;               ParOct
                                ;       fail     br     error
                                ;
                                ;       fine    r0 =    octal value
                                ;               r1 =    terminator

007762                          ParOct: ParFld                  ; r0 -> field
007764  000432                           br     30$             ; error return
007766  010104                          mov     r1,r4           ; r4 = terminator
007770  005003                          clr     r3              ; r3 = result octal
007772  111001                          movb    (r0),r1         ; check end-of-line conditions
007774  001426                          beq     30$             ; zero is EOL
007776  121027  000012                  cmpb    (r0),#lf        ; and so is line feed
010002  001423                          beq     30$             ; done
                                                                ; digit loop:
010004  112002                  10$:    movb    (r0)+,r2        ; r2 = next character
010006  120204                          cmpb    r2,r4           ; is this the terminator?
010010  001416                          beq     20$             ; yes
010012  162702  000060                  sub     #'0,r2          ; de-ascii
010016  100415                          bmi     30$             ; that ain't no digit
010020  020227  000007                  cmp     r2,#7           ; over seven?
010024  003012                          bgt     30$             ; that ain't no digit
010026  006303                          asl     r3              ; multiply accumulator by eight
010030  006303                          asl     r3              ;
010032  006303                          asl     r3              ;
010034  060203                          add     r2,r3           ; and add us in
010036  121027  000012                  cmpb    (r0),#lf        ; end-of-line?
010042  001360                          bne     10$             ; no
010044  111001                          movb    (r0),r1         ; yes - reply with terminator
010046  010300                  20$:    mov     r3,r0           ; r0 = result; r1 = terminator
010050  000606                          br      gf$sec          ; GetFil set carry exit
010052  000207                  30$:    return


                                ;       OpnFil - Open file service                              (EMT 12)
                                ;
                                ;       Space-fill filename area
                                ;       Move in file name
                                ;
                                ;       Converts rad50 filename to 12-byte ascii string
                                ;       Where "_" represents the space, the name "XXX.SYS" becomes:
                                ;
                                ;       "XXX___.SYS"
                                ;        0123456789"
                                ;
                                ;       r1 is modified
                                ;
                                ; fix   r3 not updated after "."

010054  012705  014626'         OpnFil: mov     #d$riob,r5      ; IOB
010060  066705  003250                  add     s$yrel,r5       ; r5 -> IOB
```

```
 010064  005067  003264                    clr     f$iptr          ; file pointer ground zero
 010070  005067  003254                    clr     f$ibct          ; null byte count
 010074  010501                            mov     r5,r1           ;
 010076  062701  000012                    add     #io.spc,r1      ; r1 -> io.spc
 010102  010102                            mov     r1,r2           ; r2 -> io.spc
                                                                   ;
 010104  012703  000012                    mov     #10.,r3         ; .asciz "123456.89A"
 010110  112722  000040           10$:     movb    #space,(r2)+    ; space fill the name
 010114  005303                            dec     r3              ; all ten
 010116  001374                            bne     10$             ;
                                                                   ; r1 -> io.spc
 010120  010102                            mov     r1,r2           ; r2 -> io.spc
 010122  012703  000012                    mov     #10.,r3         ; r3 = count = 10.
 010126  105710                   20$:     tstb    (r0)            ; end of string?
 010130  001411                            beq     40$             ; surely
 010132  121027  000056                    cmpb    (r0),#'.        ; at the file type?
 010136  001003                            bne     30$             ; no
 010140  010102                            mov     r1,r2           ; yes, position at byte six
 010142  062702  000006                    add     #6,r2           ; of the output string
                                  ;;;      mov     #3,r3           ; count is now three, for the file type
 010146  112022                   30$:     movb    (r0)+,(r2)+     ; copy one more
 010150  005303                            dec     r3              ; until all done
 010152  003365                            bgt     20$             ;
 010154  004775  177766           40$:     call    @dr.opn(r5)     ; the driver opens the file
 010160  016565  177764  000006            mov     dr.sbl(r5),io.blk(r5) ; file start block
 010166                                    fall    CloFil          ; exit via CloFil return


                                  ;        CloFil - Close file service                             (EMT 13)
                                  ;
                                  ;        CloFil is deprecated in the XXDP+ and XXDP V2 monitors

 010166  000207                   CloFil:  return                  ; much ado about nothing
```

```
                                .sbttl  SetLin OctAsc                                            (EMT)

                                ;       SU$UNP - Convert Rad50 to Ascii utility
                                ;
                                ;       Invalid characters are cheerfully converted to nonsense
                                ;       Called only by SpcAsc
                                ;
                                ; in    r0 =    rad50 word to translate
                                ;       r2 ->   output ascii (no zero byte terminator)
                                ;
                                ;
                                ; out   r0      burnt
                                ;       r2 ->   past ascii
                                ;       r3/r4   burnt

010170  010704                  su$unp: mov     pc,r4   ;mova   ;
010172  062704  000066                  add     #80$-.,r4       ; rad50 divisors
010176  005003                  20$:    clr     r3              ; result integer
010200  021400                  30$:    cmp     (r4),r0         ; got another subtraction?
010202  101003                          bhi     40$             ; nope
010204  161400                          sub     (r4),r0         ; subtract
010206  005203                          inc     r3              ; and count
010210  000773                          br      30$             ;
010212  005703                  40$:    tst     r3              ; nulls are spaces
010214  001406                          beq     50$             ; (14+9+9=32)
010216  120327  000033                  cmpb    r3,#27.         ; a rad50 dollar sign?
010222  001407                          beq     70$             ; yes - range 27-27 (27+9=36='$')
010224  003004                          bgt     60$             ; digit
010226  062703  000040                  add     #32.,r3         ; alphabet range 1:26 (1+32+14+9+9=65="A")
010232  062703  000016          50$:    add     #14.,r3         ; space
010236  062703  000011          60$:    add     #9.,r3          ; digit range 30:39 (30+9+9=48='0')
010242  062703  000011          70$:    add     #9.,r3          ; $
010246  110322                          movb    r3,(r2)+        ; store the byte
010250  005724                          tst     (r4)+           ; next divisor
010252  005714                          tst     (r4)            ; end of list?
010254  001350                          bne     20$             ; nope
010256  000207                          return

010260  003100  000050  000001  80$:    .word   3100, 50, 1, 0  ; rad50 divisors (1600.,40.,1,0)
010266  000000


                                ;       SetLin - Set command line service                        (EMT 26)
                                ;
                                ; in    r0      = buffer address
                                ;       r1      = buffer length
                                ;       r0=0    Use default defaults (c$lbuf, cllen.)
                                ;
                                ;       SetLin
                                ;
                                ; out   r0      = effective buffer address
                                ;       r1      = effective buffer length

        000000                          cl.ptr  = 0             ; command line base pointer
        000002                          cl.len  = 2             ; command line length
        000054                          cllen.  = 44.   ;^o54   ; command line length
        000052                          clavl.  = 42.   ;^o52   ; available characters
```

```

010270  005700                  SetLin: tst     r0              ; default?
010272  001006                          bne     10$             ; no - explicit
010274  012700  013432'                 mov     #c$lbuf,r0      ; c$olin
010300  066700  003030                  add     s$yrel,r0       ;
010304  012701  000054                  mov     #cllen.,r1      ; 44. byte command line
                                                                ;
010310  010067  003044          10$:    mov     r0,c$llin       ; line pointer
010314  162701  000002                  sub     #2,r1           ; length - 2 for termination
010320  010167  003036                  mov     r1,c$llen       ; store available length
010324  010067  003034                  mov     r0,c$lnxt       ; next is current
010330  000207                          return


                                ;       GetDat - Get date service                               (EMT 27)
                                ;
                                ;       out     r0      system date

010332  016700  003040          GetDat: mov     s$ydat,r0       ; 1970-1999
010336  000207                          return


                                ;       OctAsc - Octal to Ascii service                         (EMT 30)
                                ;
                                ;       Convert an octal value to an ascii string
                                ;       Strings are zero-filled (e.g. value=1 => string="000001")
                                ;
                                ;       r0 =    value
                                ;       r1 ->   output buffer
                                ;
                                ;       OctAsc
                                ;
                                ;       r1 ->   past last (sixth) digit
                                ;       r0      burnt

010340  010003                  OctAsc: mov     r0,r3           ; r3 = value
010342  012704  000006                  mov     #6,r4           ; r4 = counter
010346  005000                          clr     r0              ; r0 = result digit
010350  006303                          asl     r3              ; high order single bit out
010352  006100                          rol     r0              ; into r0 as the low order bit
010354  062700  000060          10$:    add     #'0,r0          ; make it ascii
010360  110021                          movb    r0,(r1)+        ; store a byte
010362  005304                          dec     r4              ; all digits done?
010364  003410                          ble     20$             ; nope
010366  005000                          clr     r0              ; reset accumulator
        000003                          .rept   3               ; rotate full digit into r0
                                        asl     r3              ;
                                        rol     r0              ;
                                        .endr                   ;
010404  000763                          br      10$             ; go store it
010406  000207                  20$:    return
```

```
                                      .sbttl  Lpt/TerMod LoaSup ParDec PadTer Psh/PopBat GetCom       (EMT)

                                      ;       This code page finishes exactly at the 12000 boundary
                                      ;       It must have been linked /high


                                      ;       LptMod - Output to printer service                      (EMT 33)

 010410  016702  004130               LptMod: mov     s$ylpt,r2       ; got a printer or something else?
 010414  001405                               beq     10$             ; nope
 010416  010267  002722                       mov     r2,s$ytps       ; csr
 010422  005722                               tst     (r2)+           ;
 010424  010267  002716                       mov     r2,s$ytpb       ; buffer
 010430  000207                       10$:    return

                                      ;       TerMod - Output to terminal service                     (EMT 34)

 010432  012767  177564  002704       TerMod: mov     #TPS,s$ytps     ; csr
 010440  012767  177566  002700               mov     #TPB,s$ytpb     ; buffer
 010446  000207                               return

                                      ;       LoaSup - Load DRS-11 supervisor HSAA??.SYS service      (EMT 35)
                                      ;
                                      ;       Batch mode activates supervisor directly
                                      ;       Takes EMT return path (via cu$qvs return)
                                      ;
                                      ;       CLI mode activates the supervisor via cu$act
                                      ;       Treats supervisor return as image exit, jumping to cl$cmd

 010450  010700                       LoaSup: mov     pc,r0           ;
 010452  062700  000044                       add     #20$-.,r0       ; r0 -> "HSAA??.SYS"
 010456  016701  002660                       mov     s$ysup,r1       ; location
 010462                                       LoaFil                  ; read it in
 010464  005067  004106                       clr     s$yerr          ;
 010470  105767  004076                       tstb    s$ybat          ; in batch mode?
 010474  001404                               beq     10$             ; nope
                                                                      ;
                                      ;       Batch-mode activation   ;
                                                                      ;
 010476  004777  002706                       call    @s$yact         ; Batch supervisor activation
 010502  000167  001140                       jmp     cu$ret          ; return via cu$ret return      (note)
                                                                      ;
                                      ;       CLI-mode activation     ;
                                                                      ;
 010506  004767  001516               10$:    call    cu$act          ; CLI supervisor activation
 010512  000167  000322                       jmp     cl$cmd          ; supervisor image exit to CLI engine

 010516     110     123     101       20$:    .asciz  "HSAA??.SYS"    ; supervisor file spec
 010521     101     077     077
 010524     056     123     131
 010527     123     000
                                              .even


                                      ;       ParDec - Parse decimal service                          (EMT 36)
                                      ;
                                      ;       in      command line field
```

```
                                ;
                                ;               ParDec
                                ;       fail     br     error   ; invalid string
                                ;
                                ;       fine    r0 =    decimal number

010532                          ParDec: ParFld                  ; isolate the field
010534  000423                           br     30$             ; errors have a fail return
010536  005002                          clr     r2              ; clear result
010540  112003                  10$:    movb    (r0)+,r3        ; next digit
010542  120103                          cmpb    r1,r3           ; is this the terminator (in r1)?
010544  001414                          beq     20$             ; yes
010546  162703  000060                  sub     #60,r3          ; de-ascii it
010552  002414                          blt     30$             ; below the digit range
010554  020327  000011                  cmp     r3,#9.          ; above the range?
010560  003011                          bgt     30$             ; yes
010562  006302                          asl     r2              ; r2 * 2
010564  060203                          add     r2,r3           ; save r2 * 2
010566  006302                          asl     r2              ; r2 * 4
010570  006302                          asl     r2              ; r2 * 8
010572  060302                          add     r3,r2           ; plus r2*2 = r2 * 10
010574  000761                          br      10$             ; try for another
010576  010200                  20$:    mov     r2,r0           ; result to r0
010600  062716  000002                   add    #2,(sp)         ; fine skip
010604  000207                  30$:    return                  ; fail return


                                ;       PadTer - Pad terminal service                           (EMT 37)
                                ;
                                ;       Write s$ypad nulls to terminal
                                ;
                                ;       PutCha invokes PadTer after outputting CR

010606  116702  002611          PadTer: movb    s$ypad,r2       ; get a counter
010612  105000                  10$:    clrb    r0              ; nulls to pad with
010614                                  PutCha                  ; at least one goes out
010616  005302                          dec     r2              ; count
010620  003374                          bgt     10$             ; more
010622  000207                          return


                                ;       PopBat - Pop batch chain file service                   (EMT 40)
                                ;
                                ;       Restore prior chain file or CLI context
                                ;
                                ;       out     r0/r1 preserved

010624  105267  002576          PopBat: incb    s$ypop          ; flag pop (not push)
010630                                  fall    PshBat          ; combine code path


                                ;       PshBat - Push batch chain file service                  (EMT 41)
                                ;
                                ;       in      r0 ->   "filnam"
                                ;
                                ;       out     r0 ->   end of copied "filnam" string
                                ;               r1 =    terminator (unused)
```

```

010630  005067  002560          PshBat: clr     f$isck          ; invalidate batch saved checksum
                                                                ;
010634  012702  013520'                 mov     #b$afnm,r2      ; current file spec
010640  066702  002470                  add     s$yrel,r2       ; r2  -> current file spec
010644  010246                          mov     r2,-(sp)        ;(sp) -> ditto
                                                                ;
010646  012703  013506'                 mov     #b$asfn,r3      ;
010652  066703  002456                  add     s$yrel,r3       ; r3 -> saved file spec
                                                                ;
010656  105767  002544                  tstb    s$ypop          ; pushing or popping?
010662  001403                          beq     10$             ; pushing
010664  010304                          mov     r3,r4           ; popping - reverse the pointers
010666  010203                          mov     r2,r3           ; r2 -> r3
010670  010402                          mov     r4,r2           ; r3 -> r4 -> r2

                                ;       Copy loop

010672  012704  000012          10$:    mov     #10.,r4         ; filespec counter
010676  112223                  20$:    movb    (r2)+,(r3)+     ; copy
010700  005304                          dec     r4              ; count
010702  001375                          bne     20$             ; more
010704  012602                          mov     (sp)+,r2        ;
                                                                ;
010706  105767  002514                  tstb    s$ypop          ; pop batch?
010712  001015                          bne     40$             ; yup

                                ;       PshBat coda

010714  112022                  30$:    movb    (r0)+,(r2)+     ; push - copy in new filename
010716  121001                          cmpb    (r0),r1         ; r1 = gtfld terminator
010720  001375                          bne     30$             ;
010722  105012                          clrb    (r2)            ; terminate string
010724  016767  002436  002436          mov     f$ipos,f$isvp   ; save current batch level position
010732  005067  002430                  clr     f$ipos          ; clear forces GetLin to open new file
010736  010200                          mov     r2,r0           ; r0 -> end of "filnam"
010740  105267  003626                  incb    s$ybat          ; => GetLin opens/reads the chain file
010744  000407                          br      50$             ;

                                ;       PopBat coda
                                ;
                                ;       Decrement the batch "stack" and restore the prior file position
                                ;       GetLin does all the rest of the work

010746  105367  003620          40$:    decb    s$ybat          ; decrement batch file stack
010752  105067  002450                  clrb    s$ypop          ; clear one-shot emt 40/41 flag
010756  016767  002406  002402          mov     f$isvp,f$ipos   ; restore prior file position
010764  000207                  50$:    return


                                ;       GetCom - Get communication area address service         (EMT 42)
                                ;
                                ;       GetCom returns a pointer to s$ycom, the system communication area
                                ;
                                ;               GetCom
                                ;
                                ;       out     r0 ->   s$ycom
```

```
    169
    170 010766  012700  014534'         GetCom: mov     #s$ycom,r0      ; point to s$ycsr
    171 010772  066700  002336                  add     s$yrel,r0       ; relocate
    172 010776  000207                          return
```

```
                                 .sbttl  XXDP CLI Engine                                          (CLI)
   011000                        x$xper:                         ; 12000

                                 ;       The area 12000:14000 is reread from disk on chain exit
                                 ;       Thus there should be no impure data in this area
                                 ;       However, there is one impure value from the boot

   011000                        o$vreg:                         ; overlay region
   011000                        o$vcli:                         ; CLI overlay block

                                 ;       CL$ABT - CLI abort routine

   011000  005700                cl$abt: tst     r0              ; got a message?
   011002  001402                        beq     cl$eng          ; no - just start over
   011004                                TypMon                  ; display message
   011006                                NewLin                  ; newline
   011010                                fall    xx$rst          ;

                                 ;       XX$RST - XXDP monitor start and restart
                                 ;
                                 ;       The init process completes by jumping to XX$RST
                                 ;       XX$RST is the advertised system restart address
                                 ;
                                 ;       RESTART ADDR: 152010
                                 ;       THIS IS XXDP+...

   011010                        xx$rst:                         ; XXDP system start and restart address
   011010  012706  013324'       cl$eng: mov     #s$ystk,sp      ; restore stack
   011014  066706  002314                add     s$yrel,sp       ; relocate
   011020  004767  001450                call    em$rst          ; restore EMT vector

                                 ;       Calls cl$cmd immediately below with a dummy RTI set PR7

   011024  012746  000340                mov     #340,-(sp)      ; build dummy int. frame
   011030  010746                        mov     pc,-(sp)        ;
   011032  062716  000006                add     #cl$cmd-.,(sp)  ; 12040 below
   011036  000002                        rti                     ; rti-as-call

                                 ;       CLI engine command loop
                                 ;
                                 ;       Image exit path
                                 ;       Command prompt/parse

   011040  010700                cl$cmd: mov     pc,r0
   011042  062700  177736                add     #cl$abt-.,r0    ; abort restarts XXDP
   011046                                SetAbt  ;cl$abt         ; generic CLI abort
   011050  004767  001644                call    mo$rst          ; restore monitor
   011054                                TerMod                  ; cancel LPT mode
   011056                                NewLin                  ;
   011060  112700  000056                movb    #'.,r0          ; command prompt "."
   011064                                PutChk                  ; say so
   011066  005000                        clr     r0              ; reset the line buffer
   011070                                SetLin
   011072  005767  000012                tst     c$laut          ; has copy of @#i$naut from the boot
   011076  001001                        bne     10$             ; is automated
   011100                                GetLin                  ; get a command line
   011102  004767  000004        10$:    call    cl$dis          ; dispatch command
```

```
011106  000740                          br      cl$eng          ; and get another

                                ;       The boot value at @#i$naut is copied here

011110  000000                  c$laut: .word   0               ;@#i$naut flag

                                ;       CL$DIS - CLI command dispatch

011112  005767  177772          cl$dis: tst     c$laut          ; automated startup?
011116  001406                          beq     10$             ; nope
011120  005067  177764                  clr     c$laut          ; yep (but once-only)
011124  010704                          mov     pc,r4           ;
011126  062704  000272                  add     #c$ltst-.,r4    ; point at TEST command entry
011132  000446                          br      70$             ; dispatch that directly

011134                          10$:    ParFld                  ; get a command name
011136  000240                           nop                    ; test below suffices
011140  105710                          tstb    (r0)            ; got a command?
011142  001446                          beq     80$             ; not this time
011144  012704  000226                  mov     #c$ldis-20$,r4  ;
011150  060704                          add     pc,r4           ; r4 -> dispatch table
011152  012702  000173          20$:    mov     #c$lloo-30$,r2  ;
011156  060702                          add     pc,r2           ; r2 -> command table

                                ;       Command lookup loop

011160  010003                  30$:    mov     r0,r3           ; r0/r3 -> command name field
011162  120163  177777          40$:    cmpb    r1,-1(r3)       ; just passed the terminator (in r1)?
011166  001402                          beq     50$             ; yes
011170  122322                          cmpb    (r3)+,(r2)+     ; same?
011172  001773                          beq     40$             ; when you're on a good thing...
011174  105762  177777          50$:    tstb    -1(r2)          ; end of the command entry?
011200  001013                          bne     60$             ; no - not a match
011202  105763  177777                  tstb    -1(r3)          ; matched to end of input?
011206  001420                          beq     70$             ; yes - got a command - dispatch it
011210  126327  177777  000040          cmpb    -1(r3),#space   ; space?
011216  001414                          beq     70$             ; yes - that's a match too
011220  126327  177777  000057          cmpb    -1(r3),#'/      ; a switch?
011226  001410                          beq     70$             ; yes - likewise good
011230  105722                  60$:    tstb    (r2)+           ; didn't match this entry
011232  001376                          bne     60$             ; skip to the end of this entry
011234  005724                          tst     (r4)+           ; pop the dispatch list
011236  001350                          bne     30$             ; nothing more to dispatch
011240  012700  011272'                 mov     #m$scmd,r0      ; invalid command
011244                                  TypMon                  ; "? INVALID COMMAND";
011246  000404                          br      80$             ; return

011250  011404                  70$:    mov     (r4),r4         ; r4 = command address
011252  066704  002056                  add     s$yrel,r4       ; relocate
                                                                ;
                                ;       Call CLI command routine;
                                ;                               ;
011256  004714                          call    (r4)            ; command dispatch
011260  000207                  80$:    return                  ; good

011262  012700  011314'         cl$ifn: mov     #m$sfnm,r0      ; invalid filename
011266                                  TypMon                  ; "? INVALID FILENAME"
```

```
011270  000207                          return

011272     077     040     111  m$scmd: .asciz  "? INVALID COMMAND"
011275     116     126     101
011300     114     111     104
011303     040     103     117
011306     115     115     101
011311     116     104     000
011314     077     040     111  m$sfnm: .asciz  "? INVALID FILENAME"
011317     116     126     101
011322     114     111     104
011325     040     106     111
011330     114     105     116
011333     101     115     105
011336     000
011337     077     040     102  m$sadr: .asciz  "? BAD ADDR"
011342     101     104     040
011345     101     104     104
011350     122     000
011352     377                          .byte   -1              ; why not .even?                (note)

                                ;       CLI command lookup table

011353     114     000          c$lloo: .asciz  "L"             ; Load
011355     123     000                  .asciz  "S"             ; Start
011357     122     000                  .asciz  "R"             ; Run
011361     103     000                  .asciz  "C"             ; Chain
011363     106     000                  .asciz  "F"             ; Fill
011365     104     000                  .asciz  "D"             ; Directory
011367     105     000                  .asciz  "E"             ; Enable
011371     110     000                  .asciz  "H"             ; Help
011373     124     105     123          .asciz  "TEST"          ; Test
011376     124     000
                                        .even

                                ;       CLI command dispatch table

011400  012064'                 c$ldis: .word   cl$loa          ; Load
011402  012154'                         .word   cl$sta          ; Start
011404  012352'                         .word   cl$run          ; Run
011406  011476'                         .word   cl$chn          ; Chain
011410  011740'                         .word   cl$fil          ; Fill
011412  012026'                         .word   cl$dir          ; Directory
011414  012000'                         .word   cl$enb          ; Enable
011416  011650'                         .word   cl$hlp          ; Help
011420  011440'                 c$ltst: .word   cl$tst          ; TEST
011422  000000                          .word   0
```

```
                                .sbttl  Test Chain Help Fill Enable Dir Load Start Run          (CLI)

                                ;       CLI TEST command
                                ;
                                ;       TEST[/QV]
                                ;
                                ;       Equivalent to "C SYSTEM.CCC"

                                        .enabl  lsb
011424     123     131     123  10$:    .asciz  "SYSTEM.CCC"    ; the system chain file
011427     124     105     115
011432     056     103     103
011435     103     000
                                        .even

011440  004767  000140          cl$tst: call    cu$qvs          ; parse /QV
011444  010702                          mov     pc,r2           ;
011446  062702  177756                  add     #10$-.,r2       ; "SYSTEM.CCC"
011452  016700  001702                  mov     c$llin,r0       ; current line pointer
011456  112220                  20$:    movb    (r2)+,(r0)+     ; copy string
011460  001376                          bne     20$             ; all of it
011462  016700  001672                  mov     c$llin,r0       ; r0 -> file spec
011466                                  PshBat                  ; start a batch level
011470  000420                          br      cu$chn          ; join Chain/TEST common code
                                        .dsabl  lsb

                                ;       Chain /QV switch

011472     121     126     000  c$sqvs: .ascii  "QV"<0><0>      ; "QV" - Quick Verify switch
011475     000


                                ;       CLI CHAIN command
                                ;
                                ;       C filnam[/QV]

                                        .enabl  lsb
011476                          cl$chn: ParFld.                 ; get the field
011500  000437                           br     20$             ; invalid filename
011502                                  PshBat                  ; set batch mode
011504  112720  000056                  movb    #'.,(r0)+       ; r0 -> past "filnam"
011510  012702  000003                  mov     #3,r2           ; counter
011514  112720  000103          10$:    movb    #'C,(r0)+       ; ".CCC"
011520  005302                          dec     r2              ; all three
011522  001374                          bne     10$             ;
011524  105010                          clrb    (r0)            ; terminate string
011526  004767  000052                  call    cu$qvs          ; parse optional /QV
011532                                  fall    cu$chn

                                ;       CU$CHN - CLI Chain/Test common
                                ;
                                ;       CU$CHN calls MO$CHN to copy/activate the batch process
                                ;       This area is overwritten by the batch overlay copy      (note)

011532  012702  000006          cu$chn: mov     #mobat.,r2      ; block = 6 (location 6000)
011536  016703  001562                  mov     s$ytra,r3       ; 10000
011542  162703  002000                  sub     #2000,r3        ; 6000 - batch area
```

```
011546  010346                          mov     r3,-(sp)        ; save r3
011550  004767  001206                  call    mo$rea          ; copy overlay
011554  012603                          mov     (sp)+,r3        ; source
011556  042737  000001  000052          bic     #scMAN$,@#52    ; CMI clear manual intervention
011564  016702  001536                  mov     s$yper,r2       ; dest overlay area
011570  012701  001414                  mov     #ovlen.,r1      ; bytes to copy
011574  000167  001252                  jmp     mo$chn          ; safely copy and initiate overlay
011600  000167  177456          20$:    jmp     cl$ifn          ; invalid filename
                                        .dsabl  lsb

                                ;       CU$QVS - Parse /QV quick verify switch
                                ;
                                ;       Called by CLI Chain and TEST commands
                                ;       There is no error reported for "/XX" etc.               (note)

011604  012702  013110'         cu$qvs: mov     #s$yswi,r2      ; /switch buffer
011610  066702  001520                  add     s$yrel,r2       ;
011614                                  ParFld                  ; get an alphanumeric field
011616  000401                           br     10$             ; which we got
011620  005300                          dec     r0              ; assume "/" precedes and back up to /
011622  112022                  10$:    movb    (r0)+,(r2)+     ; copy zero terminated string
011624  001376                          bne     10$             ;
                                                                ;
011626  012700  177636                  mov     #c$sqvs-20$,r0  ; .ascii "QV"
011632  060700                          add     pc,r0           ;
011634  004767  000560          20$:    call    cu$swi          ; parse the switch
011640  103402                          bcs     cu$ret          ; it wasn't "/QV" (and ignores others) (note)
011642  005267  002722                  inc     s$yqvs          ; set /QV quick verify switch
011646  000207                  cu$ret: return                  ; (LoaSup return path passes through here)


                                ;       CLI HELP command
                                ;
                                ;       H/L     Help Lineprinter
                                ;
                                ;       Documented in the XXDP+ User Guide
                                ;       Not listed in the XXDP+ HELP command
                                ;       Not documented the XXDPx User Guide

011650                          cl$hlp: ParFld
011652  000404                           br     10$             ; is no "/L" field
011654  121027  000114                  cmpb    (r0),#'L        ; lineprinter out?
011660  001001                          bne     10$             ; nope
011662                                  LptMod                  ; use paper
                                                                ;
011664  010700                  10$:    mov     pc,r0           ; "HELP.TXT"
011666  062700  000040                  add     #30$-.,r0       ;
011672                                  OpnFil                  ; open it (or abort)
011674                          20$:    ReaBlk                  ; read a block
011676  016765  001630  000006          mov     f$inxt,io.blk(r5) ; next block next time
011704  012700  013534'                 mov     #f$irec,r0      ; r0 -> data record
011710  066700  001420                  add     s$yrel,r0       ;
011714                                  TypMsg                  ; display input buffer
011716  005767  001610                  tst     f$inxt          ; got more?
011722  001364                          bne     20$             ; yes
011724  000207                          return
```

```
011726     110     105     114  30$:    .asciz  "HELP.TXT"      ; XXDP help text file
011731     120     056     124
011734     130     124     000
                                        .even


                                ;       CLI FILL command

011740  116700  001457          cl$fil: movb    s$ypad,r0       ; the prevaling state
011744  016701  001410                  mov     c$llin,r1       ; temporary buffer
011750                                  OctAsc                  ; octal r0 to string r1
011752  105011                          clrb    (r1)            ; terminate string
011754  016700  001400                  mov     c$llin,r0       ; get the pointer again
011760                                  TypMsg                  ; display it
011762                                  PutTab                  ; tab separator
011764                                  GetLin                  ; get a response
011766                                  ParOct                  ; ascii to octal
011770  000402                           br     10$             ; fail
011772  110067  001425                  movb    r0,s$ypad       ; set padding count
011776  000207                  10$:    return


                                ;       CLI ENABLE command
                                ;
                                ;       @dr.dev (dr$dev) updates the device ascii unit (dv.uni)
                                ;       The CLI code for this command is identical

012000                          cl$enb: ParOct                  ; get a number
012002  000410                           br     10$             ; not a number
012004  110067  002614                  movb    r0,d$runi       ; new unit
012010  012705  014626'                 mov     #d$riob,r5      ; update driver
012014  066705  001314                  add     s$yrel,r5       ;
012020  004775  177774                  call    @dr.dev(r5)     ; advise driver of change
012024  000207                  10$:    return


                                ;       CLI DIRECTORY command

012026  010700                  cl$dir: mov     pc,r0           ; point at the file spec
012030  062700  000020                  add     #10$-.,r0       ; "HUDI??.SYS"
012034  005001                          clr     r1              ; default start address
012036                                  LoaFil                  ; load it
012040  012767  000001  001340          mov     #1,s$ysta       ; default start address
012046  000470                          br      cu$act          ; setup/start
012050     110     125     104  10$:    .asciz  "HUDI??.SYS"    ; the XXDP directory cusp
012053     111     077     077
012056     056     123     131
012061     123     000
                                        .even


                                ;       CLI LOAD command
                                ;
                                ;       L filename
                                ;
                                ;       XXDP appends the file type ".BI?" to the file name
                                ;
                                ;       cu$loa called by cl$run

012064  105267  001330          cl$loa::incb    s$yloa          ; display file spec
012070                          cu$loa: ParFld                  ; get ye field
012072  000426                           br     20$             ; invalid filename (doesn't clear s$yloa)
012074  010046                          mov     r0,-(sp)        ; r0 -> field
012076  122001                  10$:    cmpb    (r0)+,r1        ; same as terminator?
012100  001376                          bne     10$             ; no - loop until that happens
012102  112760  000056  177777          movb    #'.,-1(r0)      ; "."
012110  112720  000102                  movb    #'B,(r0)+       ; ".B"
012114  112720  000111                  movb    #'I,(r0)+       ; ".BI"
012120  112720  000077                  movb    #'?,(r0)+       ; ".BI?"
012124  105010                          clrb    (r0)            ; ".BI?"<0>
012126  012600                          mov     (sp)+,r0        ; r0 -> "filnam.BI?"
012130  062767  000004  001226          add     #4,c$lnxt       ; advance next field pointer
012136  005001                          clr     r1              ; load address default
012140                                  LoaFil                  ; and read another psuedo papertape
012142  105067  001252                  clrb    s$yloa          ; disable display
012146  000207                          return

012150  000167  177106          20$:    jmp     cl$ifn          ; invalid file name


                                ;       CLI START command
                                ;
                                ;       S [address]

                                        .enabl  lsb
012154  004767  000004          cl$sta: call    cu$sta          ; get a start address
012160  103422                          bcs     30$             ; c=1 error - return
012162  000422                          br      cu$act          ; activate


                                ;       CU$STA - Get start address for RUN and START
                                ;
                                ;       in      command field
                                ;
                                ;               call    cu$sta
                                ;               bcs     fail    ; note: bcs fail, bcc fine
                                ;
                                ;       fine    r0      start address
                                ;               s$ysta  start address or #1
                                ;
                                ;       abort   "BAD ADDR."     ; for odd addresses

012164  012767  000001  001214  cu$sta: mov     #1,s$ysta       ; assume default start
012172                                  ParOct                  ; get another start address
012174  000413                           br     20$             ; fine - no address, use default
012176  032700  000001                  bit     #1,r0           ; odd addresses are just odd
012202  001003                          bne     10$             ; celebrate oddness
012204  010067  001176                  mov     r0,s$ysta       ; we have a start address
012210  000405                          br      20$             ; fine

012212  012700  011337'         10$:    mov     #m$sadr,r0      ; "BAD ADDR."
012216                                  TypMon                  ;
012220  000261                          sec                     ; c=1 => error
012222  000401                          br      30$             ;
```

```
012224  000241                  20$:    clc                     ; c=0 => fine
012226  000207                  30$:    return
                                        .dsabl  lsb


                                ;       CU$ACT - Activate CLI image
                                ;
                                ;       Called by LoaSup, Dir, Run and Start

012230  022767  000001  001150  cu$act: cmp     #1,s$ysta       ; maintenance app?
012236  001404                          beq     10$             ; yes
012240  016767  001142  001142          mov     s$ysta,s$yact   ; copy image start address
012246  000407                          br      20$
012250  022767  000001  001132  10$:    cmp     #1,s$yact       ; default image start address
012256  001003                          bne     20$             ; no - explicit
012260  012767  000200  001122          mov     #200,s$yact     ; yes- use standard start address
                                                                ;
012266                          20$:    GetDev                  ; get device info
012270  116001  000002                  movb    dv.uni(r0),r1   ; pluck off the unit digit
012274  162701  000060                  sub     #'0,r1          ; elide ascii
012300  110137  000040                  movb    r1,@#40         ; 40 - device unit
012304  116037  000003  000041          movb    dv.med(r0),@#41 ; 41 - device media code
012312  016737  001064  000030          mov     s$yemt,@#30     ; 30 - copy saved/overwritten EMT vector
012320  016737  001060  000032          mov     s$yemt+2,@#32   ; 32 -
012326  005037  000042                  clr     @#42            ; 42 -  no co-routine exit
012332  012737  000001  000052          mov     #scMAN$,@#52    ; 52 - set manual intervention (SMI)
                                                                ; (all other @#52 references bic/bis)(note)
                                ;       Activate CLI image      ;
                                                                ;
012340  004777  001044                  call    @s$yact         ; image start address
                                                                ;
012344  004767  000124                  call    em$rst          ; restore EMT vector
012350  000207                          return                  ; return to CLI engine


                                ;       CLI RUN command
                                ;
                                ;       R filnam [start address]
                                ;
                                ;       CL$LOA to parses the filename and loads the image
                                ;       However, CL$LOA does not accept a start address
                                ;       So, CL$RUN skips past the file spec (ParFld)
                                ;       Saves the C$LFLD pointer
                                ;       Calls CU$STA to pickup the file spec
                                ;       Restores the filespec pointer
                                ;
                                ;       s$yrun is checked as the high byte of s$yloa

012352                          cl$run: ParFld                  ; get filespec
012354  000417                           br     20$             ; invalid filename
012356  010046                          mov     r0,-(sp)        ; save current field
012360  004767  177600                  call    cu$sta          ; get the start address
012364  012667  000774                  mov     (sp)+,c$lnxt    ; c=? so cl$loa can reparse filename (note)
                                                                ; c=? (we must pop the stack in both cases)
012370  103410                          bcs     10$             ; c=? start address was bad
012372  105267  001023                  incb    s$yrun          ; set run-in-progress flag
012376  004767  177466                  call    cu$loa          ; load the program
```

```
012402  105067  001013                  clrb    s$yrun          ; clear run-in-progress flag
012406  004767  177616                  call    cu$act          ; full activation
012412  000207                  10$:    return

                                ;       End of CLI/Batch overlay region

        001414                          ovlen. = .-o$vreg
012414                                  assume balen. eq ovlen.

012414  000167  176642          20$:    jmp     cl$ifn          ; invalid file name


                                ;       CU$SWI - Check Batch IF and CLI CHAIN/TEST /QV switches
                                ;
                                ;       CU$SWI sits just outside the CLI/Batch overlay region
                                ;
                                ;       in      r0 ->   candidate switch "XX"
                                ;
                                ;               call    cu$swi
                                ;
                                ;       fail    bcs     fail
                                ;       fine    bcc     fine                            (note bcc fine)
                                ;
                                ;       r2/r4   burnt

012420  012702  000462          cu$swi: mov     #s$yswi-10$,r2  ; the switch buffer
012424  060702                          add     pc,r2           ;
012426  105712                  10$:    tstb    (r2)            ; end of switches?
012430  001417                          beq     50$             ; yes - fail
012432  122227  000057                  cmpb    (r2)+,#'/       ; "/"
012436  001373                          bne     10$             ; must be found
012440  010004                          mov     r0,r4           ; r4 -> candidate
012442  122422                  20$:    cmpb    (r4)+,(r2)+     ; r2 -> stored
012444  001776                          beq     20$             ; compare until missmatch
012446  126227  177777  000057  30$:    cmpb    -1(r2),#'/      ; terminated by "/"
012454  001403                          beq     40$             ; yes - multiple switches
012456  105762  177777                  tstb    -1(r2)          ; terminated at end of string?
012462  001361                          bne     10$             ; no - start over
                                                                ;
012464  000241                  40$:    clc                     ; fine - switch found
012466  000401                          br      60$             ;
012470  000261                  50$:    sec                     ; fail - no such switch
012472  000207                  60$:    return
```

```
                                .sbttl  EMT Engine                                              (EMT)

                                ;       EM$RST - Restore EMT vector

012474  012737  012520' 000030  em$rst: mov     #em$eng,@#v$eemt        ; rebuild EMT vector
012502  066737  000626  000030          add     s$yrel,@#v$eemt         ; relocate
012510  012737  000340  000032          mov     #340,@#v$eemt+2         ; PR7
012516  000207                          return

                                ;       EM$ENG - EMT dispatch engine
                                ;
                                ;       r0 is undefined for many services
                                ;       r1 is wilful

012520                                  stack   r2,r3,r4,pc,ps
012520  010446                  em$eng: mov     r4,-(sp)                ; r5 shared
012522  010346                          mov     r3,-(sp)                ; r2/r3/r4 saved
012524  010246                          mov     r2,-(sp)                ; r0/r1 - arguments/results
                                                                        ;
012526  042766  000001  000010          bic     #cbit,sp.ps(sp)         ; clear return c-bit
012534  016604  000006                  mov     sp.pc(sp),r4            ; get the pc
012540  116404  177776                  movb    -2(r4),r4               ; get the (unsigned) EMT code (sanity)
012544  006304                          asl     r4                      ; make bytes, not words
012546  012703  000032                  mov     #e$mdis-10$,r3          ; dispatch table address
012552  060703                          add     pc,r3                   ; relocate
012554  060304                  10$:    add     r3,r4                   ; add offset and table
012556  011404                          mov     (r4),r4                 ; get the table entry
012560  066704  000550                  add     s$yrel,r4               ; and relocate that
                                ;                                       ;
                                ;       Call EMT service                ;
                                ;                                       ;
012564  004714                          call    (r4)                    ;\call the thing
012566  000403                          br      20$             ;plain  ;|don't alter return address
012570  062766  000002  000006           add    #2,sp.pc(sp)    ;skip   ;/propagate skip return
                                                                        ;
012576  012602                  20$:    mov     (sp)+,r2                ; restore registers
012600  012603                          mov     (sp)+,r3                ;
012602  012604                          mov     (sp)+,r4                ;
012604  000002                          rti                             ; return to caller

                                ;       EMT dispatch list

012606  007130'                 e$mdis: .word   GetLin  ; 0  GetLin
012610  007412'                         .word   ParFld  ; 1  ParFld
012612  007504'                         .word   TypMon  ; 2  TypMon
012614  007512'                         .word   TypMsg  ; 3  TypMsg
012616  007554'                         .word   PutChk  ; 4  PutChk
012620  007566'                         .word   GetAvl  ; 5  GetAvl
012622  007656'                         .word   GetChk  ; 6  GetChk
012624  007732'                         .word   NewLin  ; 7  NewLin
012626  007742'                         .word   PutTab  ; 10 PutTab
012630  007762'                         .word   ParOct  ; 11 ParOct
012632  010054'                         .word   OpnFil  ; 12 OpnFil
012634  010166'                         .word   CloFil  ; 13 CloFil
012636  014654'                         .word   LoaFil  ; 14 LoaFil
012640  015204'                         .word   ReaWrd  ; 15 ReaWrd
012642  015224'                         .word   ReaByt  ; 16 ReaByt
```

```
012644  015322'                         .word   PutCha  ; 17 PutCha
012646  015376'                         .word   ReaNxt  ; 20 ReaNxt
012650  015416'                         .word   ReaBlk  ; 21 ReaBlk
012652  015466'                         .word   SetAbt  ; 22 SetAbt
012654  015474'                         .word   JmpAbt  ; 23 JmpAbt
012656  015564'                         .word   CmpSpc  ; 24 CmpSpc
012660  015626'                         .word   SpcAsc  ; 25 SpcAsc
012662  010270'                         .word   SetLin  ; 26 SetLin
012664  010332'                         .word   GetDat  ; 27 GetDat
012666  010340'                         .word   OctAsc  ; 30 OctAsc
012670  015500'                         .word   GetDev  ; 31 GetDev
012672  015506'                         .word   RptFld  ; 32 RptFld
012674  010410'                         .word   LptMod  ; 33 LptMod
012676  010432'                         .word   TerMod  ; 34 TerMod
012700  010450'                         .word   LoaSup  ; 35 LoaSup
012702  010532'                         .word   ParDec  ; 36 ParDec
012704  010606'                         .word   PadTer  ; 37 PadTer
012706  010630'                         .word   PshBat  ; 40 PshBat
012710  010624'                         .word   PopBat  ; 41 PopBat
012712  010766'                         .word   GetCom  ; 42 GetCom
012714  015516'                         .word   GetDrv  ; 43 GetDrv
012716  007534'                         .word   TypBrk  ; 44 TypBrk
                                ;MACROM .word           ; 45 ChkAbt (not in XXDPSM???)
                                ;XXDPSM .word           ; 46 LoaDat (load Date command app)
                                .sbttl  Monitor Restore, Overlay Read and Copy                  (monitor)

                                ;       The CLI restore monitor copy area ends at 140000 (in mo$rea).
                                ;
                                ;       These are the monitor block numbers of the areas of interest
                                ;
                                ;       moBAT. = 6              ; batch area block
                                ;       moTRA. = 8              ; transient area block
                                ;       moCLI. = 10.            ; cli area block
                                ;       moLEN. = 1414           ; overlay length

                                ;       MO$RST - Restore the monitor transient area
                                ;
                                ;       Checksum the transient area

                                        .enabl  lsb
012720  016700  000400          mo$rst: mov     s$ytra,r0       ;10000  ; transient area
012724  005001                          clr     r1              ;0      ; checksum
012726  062001                  10$:    add     (r0)+,r1        ;0+n    ; accumulate checksum
012730  020067  000372                  cmp     r0,s$yper       ;12000  ; reached the permanent area?
012734  001374                          bne     10$             ;       ; nope
012736  026701  000454                  cmp     s$y5ck,r1       ;       ; have we changed?
012742  001432                          beq     40$             ;       ; no - we're done

                                ;       Restore the transient area

012744  012702  000010                  mov     #motra.,r2      ;10/8.  ; block = 8
012750  016703  000350                  mov     s$ytra,r3       ;10000  ; buffer = s$ytra
012754  004767  000002                  call    mo$rea          ;       ; restore 512. words
012760  000420                          br      30$             ;       ; go say ".5K RESTORED"

                                ;       MO$REA - read 512. words from the monitor file
                                ;
                                ;       r2 =    block
                                ;       r3 ->   buffer
                                ;       512.    fixed word count
                                ;
                                ;       The call to DR.RST below

012762  012705  014626'         mo$rea: mov     #d$riob,r5      ; system IOB
012766  066705  000342                  add     s$yrel,r5       ;
012772  010265  000006                  mov     r2,io.blk(r5)   ; r2 =  block
012776  010365  000004                  mov     r3,io.buf(r5)   ; r3 -> buffer
013002                          x$xsta:                         ; overlay end, static begin
013002  012765  001000  000002          mov     #512.,io.wct(r5); word count (2*256. words)
013010  004775  177770                  call    @dr.rst(r5)     ; read
013014  005067  000374                  clr     f$isck          ; clear batch saved checksum
013020  000207                          return
013022  012700  013032'         30$:    mov     #50$,r0         ; type ".5K RESTORED"
013026                                  TypMon                  ;
013030  000207                  40$:    return
013032     056     065     113  50$:    .asciz  ".5K RESTORED"<cr><lf>
013035     040     122     105
013040     123     124     117
013043     122     105     104
013046     015     012     000
                                        .even
                                        .dsabl  lsb

                                ;       MO$CHN - Chain completion
                                ;
                                ;       Copy batch to overlay region
                                ;       Call the batch engine
                                ;       Restore monitor CLI context
                                ;
                                ;       This CLI code must reside outside the overlay region
                                ;       XXDP reads 512. words to restore CLI                    (note)
                                ;       The CLI read ends at 14000 (at x$xres, 52 bytes above)
                                ;
                                ;       r3 ->   source
                                ;       r2 ->   dest
                                ;       r1 =    byte counter

013052  112322                  mo$chn: movb    (r3)+,(r2)+       ; copy chain area
013054  005301                          dec     r1                ; byte by byte
013056  001375                          bne     mo$chn            ; leaving none out
                                                                  ;
                                ;       Call batch engine         ;
                                                                  ;
013060  004767  175716                  call    ba$eng+$$         ; call batch (only use of +$$)  (note)
                                                                  ;
013064                                  PopBat                    ; restore CLI context
013066  105067  000335                  clrb    s$yqui            ; switch off quiet mode
                                                                  ; setup and read the CLI back in
013072  012702  000012                  mov     #moCLI.,r2        ; r2 = block 10. (12)
013076  016703  000224                  mov     s$yper,r3         ; r3 = -> 12000/152000
013102  004767  177654                  call    mo$rea            ; read monitor CLI engine overlay and more
013106  000207                          return                    ; 12000-14000 / 152000-154000
```

```
                                .sbttl  System Data & Communication tables                      (data)

013110                          s$yswi: .blkw   12.     ;14110  ;  batch/CLI switch buffer
013140  000072                  s$ygto: .rept   58.     ;14140  ;  batch GOTO buffer            (hack)(note)
                                        .word   123456          ;  stack pattern
                                        .endr                   ;  58. word stack
013324                          s$ystk:                 ;14324  ;  system stack top
013324  000000                  s$ytra: .word   0;150000;14324  ;  -> .5k transient area
013326  000000                  s$yper: .word   0;152000;14326  ;  -> permanent memory area
013330  177546                  h$wltc: .word   177546  ;14330  ;  line clock
013332  172540                  h$wkwp: .word   172540  ;14332  ;  KW11P programmable clock
013334  000000                  s$yrel: .word   0;140000;14334  ;  relocation constant
013336  000000                  s$yrpt: .word   0       ;14336  ;  diagnostic repeat count
013340  000000                  s$ydev: .word   0       ;14340  ;  -> .ascii "DD"
013342  000000                  s$ysup: .word   0;137000;14342  ;  ACT supervisor load address
013344  000000                  s$ytps: .word   0;TPS   ;14344  ;  TPS/LPT csr pointer
013346  000000                  s$ytpb: .word   0;TPB   ;14346  ;  TPB/LPB buffer pointer
013350  000000                  f$ibct: .word   0       ;14350  ;  ReaByt file byte count
013352  000000                  s$ytop: .word   0;160000;14352  ;  top of memory
013354  000000                  f$iptr: .word   0       ;14354  ;  file buffer pointer
013356  000000                  f$ilck: .word   0       ;14356  ;  LDA load file read checksum
013360  000000                  c$llin: .word   0       ;14360  ;\ resident command pointer
013362  000000                  c$llen: .word   0       ;14362  ;/ line length
013364  000000                  c$lnxt: .word   0       ;14364  ;  points to next command field
013366  000000                  f$ipos: .word   0       ;14366  ;  current file position
013370  000000                  f$isvp: .word   0       ;14370  ;  saved/restored file position
013372  000000                  c$lfld: .word   0       ;14372  ;  current field pointer
013374  000000                          .word   0       ;14374  ;  ???
013376  000000                  s$ydat: .word   0       ;14376  ;  system DOSbatch date
013400  000000                  s$yabt: .word   0       ;14400  ;  setabt/jmpabt address
013402  000000  000000          s$yemt: .word   0,0     ;14402  ;  saved EMT vector during image load
013406  177777                  s$ysta: .word   177777  ;14406  ;  image START command address and type
013410  177777                  s$yact: .word   177777  ;14410  ;  image activate address
013412  177777                  f$irck: .word   177777  ;14412  ;  ReaBlk checksum
013414  000000                  f$isck: .word   0       ;14414  ;  Batch saved ReaBlk checksum
013416  000000                  s$y5ck: .word   0       ;14416  ;  .5k area checksum
                                                                ;
013420     000                  s$yloa: .byte   0       ;14420  ;\ LOAD in-progress flag
013421     000                  s$yrun: .byte   0       ;14421  ;/ cli RUN in-progress flag
013422     000                  s$yhlt: .byte   0       ;14422  ;  halt after load flag         .
013423     014                  s$ypad: .byte   12.;14  ;14423  ;  fill count (reset to 1)(preset)(note)
013424     000                  s$ycol: .byte   0       ;14424  ;  column (for tabbing)
013425     000                  s$ypnd: .byte   0       ;14425  ;  pending input character
013426     000                  s$ypop: .byte   0       ;14426  ;  PopBat flag (i.e. not PshBat)
013427     000                  s$yqui: .byte   0       ;14427  ;  negative => quiet mode
013430     000     377                  .byte   0,-1    ;14430  ;\ command line backstop
013432                          c$lbuf: .blkb   clavl.  ;14432  ;| command line buffer
013504  000000                          .word   0       ;14504  ;/ command line terminator
013506  000000                  b$asfn: .word   0       ;14506  ;\ fil - saved batch file name
013510  000000                          .word   0       ;14510  ;| nam
013512  000000                          .word   0       ;14512  ;| typ
013514  000000                          .word   0       ;14514  ;| buf
013516  000000                          .word   0       ;14516  ;/ nxt
013520  000000                  b$afnm: .word   0       ;14520  ;\ fil - batch file name
013522  000000                          .word   0       ;14522  ;| nam
013524  000000                          .word   0       ;14524  ;| typ
013526  000000                          .word   0       ;14526  ;| buf
```

```
013530  000000                          .word   0       ;14530  ;/ nxt
013532                          f$ibuf: ;blkw   256.    ;14532  ;| file system block buffer
013532  000000                  f$inxt: .word   0       ;14532  ;X -> to next file block
013534                          f$irec: .blkw   256.-1  ;14534  ;| input record
                                        ;....           ;15531  ;| record top
014532  000000                          .word   0       ;15532  ;/ buffer parse & print terminator

                                ;       Monitor communication area

014534                          s$ycom:
014534  000000                  s$ycsr: .word   0       ;15534  ;\ CSR address                  (init)
014536  000000                          .word   0       ;15536  ;| ???
014540  000000                  s$yuni: .word   0       ;15540  ;| unit number                  (init only)
014542  000000                  s$ycfg: .word   0       ;15542  ;| config flags (LPT$)
014544  000000                  s$ylpt: .word   0       ;15544  ;| LPT CSR if present
014546  000000                  s$ykwd: .word   0       ;15546  ;| kwords memory size
014550  000000                  s$yltc: .word   0       ;15550  ;|\ LTC ISR and block
014552  000006                          .word   6       ;15552  ;|| LTC priority
014554  000100                          .word   100     ;15554  ;|| LTC clock vector
014556  000074                  s$yltk: .word   60.     ;15556  ;|/ LTC clock-ticks    (50hz=50.)(init)
014560  000000                  s$ykwp: .word   0       ;15560  ;|\ KWP ISR and block
014562  000006                          .word   6       ;15562  ;|| KWP priority
014564  000104                          .word   104     ;15564  ;|| KWP vector
014566  000074                  s$yktk: .word   60.     ;15566  ;|/ KWP clock-ticks    (50hz=50.)(init)
014570  000000                  s$yqvs: .word   0       ;15570  ;| /QV quick verify switch
014572  000000                  s$ybat: .word   0       ;15572  ;| 1=batch mode and level
014574  000777                  s$ypgs: .word   512.-1  ;15574  ;| MMU 32w-pages - 1 (777=16kw-1pg)
014576  000000                  s$yerr: .word   0       ;15576  ;/ apps report errors to batch here

                                ;       Driver area

014600  000000                  d$rcom: .word   0       ;15600  ;752    ;\ dr.buf
014602  000000                          .word   0       ;15602  ;754    ;| dr.ent - entry number in segment
014604  000000                  d$rfnm: .rad50  /   /   ;15604  ;756    ;| dr.fnm - .rad50 /filnamtyp/
014606  000000                          .rad50  /   /   ;15606  ;760    ;|          copied here by dr$opn
014610  000000                          .rad50  /   /   ;15610  ;762    ;|          along with dr.sbl below
014612  000000                          .word   0       ;15612  ;764    ;| dr.sbl - first file block
                                                                        ;  begin GetDrv copy area
014614  016372'                 d$rdis: .word   dr$opn  ;15614  ;766    ;\ dr.opn - open file       (init)
014616  016642'                         .word   dr$rst  ;15616  ;770    ;| dr.rst - restore monitor (init)
014620  015702'                         .word   dr$tra  ;15620  ;772    ;| dr.tra - transfer        (init)
014622  016342'                         .word   dr$dev  ;15622  ;774    ;| dr.dev - device name     (init)
014624                          p$scsu:;.word   dlcsu.  ;15624\ ;       ;: CSR with unit in place   (preset)
014624     000                  d$runi::.byte   0       ;15624/ ;776    ;| dr.uni - device unit
014625     000                          .byte   0       ;15625  ;777    ;| dr.sts - operation status
014626                          IOB:
014626                          d$riob:
014626  174400                  d$rcsr::.word   dlcsr.  ;15626  ; 00    ;| CSR
014630  000000                          .word   0       ;15630  ; 02    ;| io.wct
014632  000000                          .word   0       ;15632  ; 04    ;| io.buf
014634  000000                          .word   0       ;15634  ; 06    ;| io.blk
014636  015656'                         .word   d$pufd  ;15636  ; 10    ;| io.ufd - relocated   (init)
014640                          d$rend:                                 ;/ end GetDrv copy area
014640                          i$ospc: .blkb   12.;14  ;15640  ; 12    ;  io.spc  .asciz "filnam.typ"<0>
                                        ;word   0       ;15642  ; 14    ;
                                        ;word   0       ;15644  ; 16    ;
                                        ;word   0       ;15646  ; 20    ;
```

```
    115                                         ;word   0       ;15650  ; 22    ;
    116                                         ;word   0       ;15652  ; 24    ;
```

```
                                .sbttl  LoaFil                                                  (EMT)

                                ;       LoaFil - Load file service                              (EMT 14)
                                ;
                                ;       in      r0 ->   ascii filespec
                                ;               r1 =    load address
                                ;
                                ;               LoaFil
                                ;
                                ;       abort   r0 ->   "? CKERR" (or "? RD ERR")

014654  010103                  LoaFil: mov     r1,r3           ; r3 = base address
014656  105067  176540                  clrb    s$yhlt          ; clear halt flag
                                                                ;
014662                                  movr    #s$yemt,r2,7    ; EMT vector can't be modified during loading
014666  013722  000030                  mov     @#v$eemt,(r2)+  ; so we save it to a temp buffer
014672  013722  000032                  mov     @#v$eemt+2,(r2)+; which is later copied into place
                                                                ;
014676  012737  000137  000200          mov     #137,@#200      ; setup a default start address
014704  012737  002100  000202          mov     #2100,@#202     ; 200: jmp @#2100; what is this?   (note)

014712                                  OpnFil                  ; look for the file
014714  005767  176500                  tst     s$yloa          ; batch/cli LOAD or cli RUN command?
014720  001410                          beq     10$             ; nope (this tests s$yloa and s$yrun)
                                                                ; display filespec
014722                                  movr    #d$rfnm,r1,6    ; rad50 filename
014726                                  movr    #i$ospc,r2,8    ; to ascii filespec
014732                                  SpcAsc                  ; ascify rad50 filename
014734  010200                          mov     r2,r0           ; point to ascii
014736                                  TypMsg                  ; display the name
014740                                  NewLin                  ;

                                ;       r0      incoming byte
                                ;       r1      store address
                                ;       r2      record size
                                ;       r3      load base address (from caller r1)

014742  005067  176410          10$:    clr     f$ilck          ; zap load checksum
014746                                  ReaByt                  ; looking for start-of-record
014750  120027  000001                  cmpb    r0,#1           ; which is a one
014754  001372                          bne     10$             ; try again
014756                                  ReaByt                  ; which is followed by a null
014760  005700                          tst     r0              ; got a null?
014762  001367                          bne     10$             ; not today
                                                                ;
014764                                  ReaWrd                  ; next comes the record byte count
014766  010002                          mov     r0,r2           ; r2 = record size
                                                                ;
014770                                  ReaWrd                  ; now we want an address
014772  010001                          mov     r0,r1           ; which goes into r1
014774  060301                          add     r3,r1           ; plus the load base address
014776  162702  000006                  sub     #6,r2           ; subtract header size from byte count
015002  001450                          beq     80$             ; zero means we have a transfer record
015004  003440                          ble     70$             ; strange choice of branch      (note)

                                ;       Read LDA record

```

```
015006                          20$:    ReaByt                  ; read the next byte

                                ;       Force console start if LOAD overwrites transient area   (note)
                                ;       (because we can't return there for normal completion)
                                ;       (it's not a problem for RUN because it doesn't return)

015010  020167  176310                  cmp     r1,s$ytra       ; overwriting transient area?
015014  103412                          blo     30$             ; no
015016  105767  176376                  tstb    s$yloa          ; is this a LOAD command?
015022  001407                          beq     30$             ; nope
015024  012700  015152'                 mov     #m$scon,r0      ; yes - and that has console consequences
015030                                  TypMon                  ; "USE CPU CONSOLE TO START"
015032  105067  176362                  clrb    s$yloa          ; once-only flag (don't repeat message)
015036  105267  176360                  incb    s$yhlt          ; flag halt after load (below)

                                ;       Handle EMT vector overwrite

015042                          30$:    movr    #s$yemt-1,r4,9  ; save app emt vector in s$yemt
015046  012746  000027                  mov     #v$eemt-1,-(sp) ; which run/load copies into place
015052  005216                  40$:    inc     (sp)            ; first inc points at v$eemt (@#30)
015054  005204                          inc     r4              ; and s$yemt, a temporary copy area
015056  021627  000034                  cmp     (sp),#v$eemt+4  ; passed emt vector pair?
015062  103005                          bhis    50$             ; yes
015064  020116                          cmp     r1,(sp)         ; is the v$eemt area 30,31,32,33?
015066  001371                          bne     40$             ; no - keep going
015070  110014                          movb    r0,(r4)         ; yes - squirrel it away
015072  105721                          tstb    (r1)+           ; skip it
015074  000401                          br      60$             ; and proceed as if nothing had happened

                                ;       Store a byte and loop

015076  110021                  50$:    movb    r0,(r1)+        ; wow - actually store a byte
015100  005726                  60$:    tst     (sp)+           ; pop the (sp) temp
015102  005302                          dec     r2              ; more bytes in record?
015104  001340                          bne     20$             ; yes

                                ;       End of record, handle checksum error

015106                          70$:    ReaByt                  ; read checksum byte
015110  105767  176242                  tstb    f$ilck          ; the load checksum must be zero
015114  001712                          beq     10$             ; it is - get next record
015116  012700  015142'                 mov     #m$schk,r0      ; "CKERR" (abort routine relocates)
015122                                  JmpAbt                  ; we are finished

                                ;       End of load
                                ;
                                ;       Halt if Load overwrites transient area

015124  010167  176260          80$:    mov     r1,s$yact       ; store activate address
015130  105767  176266                  tstb    s$yhlt          ; forced halt?
015134  001401                          beq     90$             ; no - caller completes activation
015136  000000                          HALT                    ; HALT for user to run program manually
015140  000207                  90$:    return                  ; CONTINUE returns to activate

015142     077     040     103  m$schk: .asciz  "? CKERR"       ; checksum error
015145     113     105     122
015150     122     000
```

```
    113 015152     125     123     105  m$scon: .asciz  "USE CPU CONSOLE TO START"<ht>
        015155     040     103     120
        015160     125     040     103
        015163     117     116     123
        015166     117     114     105
        015171     040     124     117
        015174     040     123     124
        015177     101     122     124
        015202     011     000
    114                                          .even
```

```
                                .sbttl  ReaWrd ReaByt PutCha ReaNxt ReaBlk                      (EMT)

                                ;       ReaWrd - Read word service                              (EMT 15)
                                ;
                                ;       Read two bytes, replacing missing bytes with zero
                                ;
                                ;       in      r5 ->   file IOB
                                ;
                                ;       out     bcs     fine
                                ;               bcc     fail
                                ;
                                ;       fine    r0      word
                                ;       fail    r0      undefined

                                        .enabl  lsb
015204                          ReaWrd: ReaByt                  ; get another byte
015206  103044                           bcc    40$             ; end of file - return
015210  010004                          mov     r0,r4           ; save first byte
015212                                  ReaByt                  ; read another
015214  103041                           bcc    40$             ; oops - end of file
015216  000300                          swab    r0              ; new byte to high byte
015220  050400                  10$:    bis     r4,r0           ; combine
015222  000433                          br      30$             ; propagate good cbit return


                                ;       ReaByt - Read byte service                              (EMT 16)
                                ;
                                ;       in      r5 -> file IOB
                                ;
                                ;               ReaByt
                                ;       fail     bcc    eof
                                ;
                                ;       fine    r0 =    byte
                                ;
                                ;       abort   "Rd Er" from ReaBlk

015224  005767  176120          ReaByt: tst     f$ibct          ; got more bytes to eat?
015230  003016                          bgt     20$             ; yes
015232  005765  000006                  tst     io.blk(r5)      ; got a real block?
015236  001430                          beq     40$             ; fail - return
015240  004767  000152                  call    ReaBlk          ; ReaBlk
015244  016765  176262  000006          mov     f$inxt,io.blk(r5); link file forward
015252  012767  000776  176070          mov     #512.-2,f$ibct  ; 512 - 2 for the header word
015260                                  movr    #f$irec,f$iptr,2; relocated

                                ;       Get next byte

015266  117700  176062          20$:    movb    @f$iptr,r0      ; actually get a character
015272  042700  177400                  bic     #^c377,r0       ; clean up to 7-bit ascii
015276  060067  176054                  add     r0,f$ilck       ; add to  to the LoaFil checksum
015302  005267  176046                  inc     f$iptr          ; buffer pointer
015306  005367  176036                  dec     f$ibct          ; count down
015312  052766  000001  000012  30$:    bis     #cbit,sp.ps+2(sp); fine                         (note)
015320  000207                  40$:    return
                                        .dsabl  lsb


                                ;       PutCha - Put character service                          (EMT 17)
                                ;
                                ;       in      r0      output character
                                ;
                                ;       out     r0      available input character or zero

015322  120027  000012          PutCha: cmpb    r0,#lf          ; linefeed?
015326  001002                          bne     10$             ; nope
015330  105067  176070                  clrb    s$ycol          ; column zero
015334  120027  000011          10$:    cmpb    r0,#ht          ; tab?
015340  001003                          bne     20$             ; nope
015342  004767  172374                  call    PutTab          ; go tab
015346  000411                          br      30$             ;
015350  004767  171424          20$:    call    te$put          ; out to TPS
015354  105267  176044                  incb    s$ycol          ; up column count
015360  120027  000015                  cmpb    r0,#cr          ; carriage return?
015364  001002                          bne     30$             ; nope
015366  004767  173214                  call    PadTer          ; some mechanical time
015372                          30$:    GetAvl                  ; returns available character
015374  000207                          return


                                ;       ReaNxt - Read next block service                        (EMT 20)
                                ;
                                ;       in      f$inxt  next block
                                ;
                                ;               ReaNxt
                                ;       fail     br     eof
                                ;               ...
                                ;
                                ;       abort   abort   "Rd Er"

015376  016765  176130  000006  ReaNxt: mov     f$inxt,io.blk(r5); advance to next block
015404  001403                          beq     10$             ; there is no next block
015406                                  ReaBlk                  ; read it
015410  062716  000002                   add    #2,(sp)         ; we're always good
015414  000207                  10$:    return                  ; we ain't


                                ;       Reablk - Read Block service                             (EMT 21)
                                ;
                                ;       Read block and compute block checksum
                                ;
                                ;       in      r5 ->   IOB
                                ;               io.blk  block number
                                ;
                                ;               ReaBlk
                                ;       fine    ...
                                ;       fail    abort   "? RD ERR"
                                ;
                                ;               call    rb$chk
                                ;       out     r2:r4   burnt
                                ;               f$irck  block checksum

015416  016565  177752  000004  ReaBlk: mov     dr.buf(r5),io.buf(r5); buffer
015424  012765  000400  000002          mov     #256.,io.wct(r5); word count
015432  004775  177772                  call    @dr.tra(r5)     ; transfer
```

```

                                 ;       The checksum includes the block-chain header word which helps
                                 ;       differentiate data blocks that are entirely composed of zeroes.

015436                           rb$chk:                         ; GetLin entry point
015436  016502  177752                   mov     dr.buf(r5),r2   ; point to buffer
015442  012703  000400                   mov     #256.,r3        ; our counter
015446  005004                           clr     r4              ; our checksum
015450  062204                   10$:    add     (r2)+,r4        ; accumulate
015452  005303                           dec     r3              ; count
015454  001375                           bne     10$             ;
015456  005204                           inc     r4              ; avoid matching zero checksum  (note)
015460  010467  175726                   mov     r4,f$irck       ; new file read checksum
015464  000207                           return
```

```
                                .sbttl  SetAbt JmpAbt GetDev RptFld GetDrv CmpSpc SpcAsc        (EMT)

                                ;       SetAbt - Set abort address service                      (EMT 22)
                                ;
                                ;       in      r0 ->   abort function
                                ;
                                ;               SetAbt
                                ;
                                ;       out     s$yabt->abort function

015466  010067  175706          SetAbt: mov     r0,s$yabt       ; save address
015472  000207                          return


                                ;       JmpAbt - Jump to abort routine service                  (EMT 23)
                                ;
                                ;       in      r0 ->   (unrelocated)abort message
                                ;               r0=0    no message
                                ;
                                ;               JmpAbt
                                ;
                                ;       The abort routine is responsible for relocating monitor
                                ;       messages.

015474  000177  175700          JmpAbt: jmp     @s$yabt         ; jump to abort


                                ;       GetDev - Get device information service                 (EMT 31)
                                ;
                                ;       out     r0 ->   device info block
                                ;
                                ;       dv.nam  = 0             ;.ascii "DL"    ; driver name
                                ;       dv.uni  = 2             ;.byte  "0"     ; device unit
                                ;       dv.med  = 3             ;.byte  dlMED.  ; media code
                                ;        dvRK5. = 2             ; DK: disk
                                ;        dvRL1. = 1     4       ; DL: disk

015500  016700  175634          GetDev: mov     s$ydev,r0       ; r0 -> "DD"
015504  000207                          return


                                ;       RptFld Repeat command field service                     (EMT 32)
                                ;
                                ;       Repeat field supports command line look-ahead parsing
                                ;       See IN$PDT (init parse date) for an example

015506  016767  175660  175650  RptFld: mov     c$lfld,c$lnxt   ; next field is current field
015514  000207                          return


                                ;       GetDrv - Get driver service                             (EMT 43)
                                ;
                                ;       in      r0 ->   driver output area
                                ;               r1 ->   driver dispatch table output area
                                ;
                                ;               GetDrv
                                ;
                                ;       out     r0 ->   monitor driver area
                                ;               r1 ->   past Monitor dispatch table
                                ;
                                ;       GetDrv is deprecated in XXDPV2

015516                          GetDrv: movr    #x$xdrv,r4,13   ; r4 -> x$xdrv driver region
015522  010402                          mov     r4,r2           ; r2 -> ditto
015524  016703  175622                  mov     s$ytop,r3       ; r3 -> x$xtop - top of memory
015530  160403                          sub     r4,r3           ; length of driver
015532  006203                          asr     r3              ; as words
015534  012220                  10$:    mov     (r2)+,(r0)+     ; r0 -> copy area
015536  005303                          dec     r3              ; count them
015540  003375                          bgt     10$             ;
                                                                ;
015542                                  movr    #d$rdis,r2,14   ; r2 -> driver dispatch interface base
015546  012700  000012                  mov     #10.,r0         ; r0 =  copy count
015552  012221                  20$:    mov     (r2)+,(r1)+     ; r1 -> copy output
015554  005300                          dec     r0              ; count
015556  002375                          bge     20$             ; more
015560  010400                          mov     r4,r0           ; r0 -> monitor driver area
015562  000207                          return                  ; r1 -> past monitor dispatch table


                                ;       CmpSpc  - Compare file specs service                    (EMT 24)
                                ;
                                ;       in      r0 ->   wildcard spec   (e.g. "filnam.bi?")
                                ;               r2 ->   candidate space (e.g. "mydiag.bic" or ".bin")
                                ;
                                ;               CmpSpc
                                ;       fail     br     fail
                                ;       fine    ...
                                ;
                                ;       out     r0      burnt

015564  012704  000012          CmpSpc: mov     #10.,r4         ; counter
015570  122710  000077          10$:    cmpb    #'?,(r0)        ; "?"?
015574  001403                          beq     20$             ; yes
015576  122710  000045                  cmpb    #'%,(r0)        ; "%"?
015602  001002                          bne     30$             ; yes
015604  122022                  20$:    cmpb    (r0)+,(r2)+     ; match anything
015606  000402                          br      40$             ; advance
015610  122022                  30$:    cmpb    (r0)+,(r2)+     ; specific match
015612  001004                          bne     50$             ; fail
015614  005304                  40$:    dec     r4              ; more?
015616  001364                          bne     10$             ; more
015620  062716  000002                   add    #2,(sp)         ; fine
015624  000207                  50$:    return


                                ;       SpcAsc - Convert rad50 spec to ascii service            (EMT 25)
                                ;
                                ;       in      r1 ->   .rad50  /filnamtyp/
                                ;               r2 ->   output buffer
                                ;
                                ;               SpcAsc
                                ;
                                ;       out     r0/r1   burnt
```

```

015626  012100                  SpcAsc: mov     (r1)+,r0        ; "fil"
015630  004767  172334                  call    su$unp          ; unpack rad50 word
015634  012100                          mov     (r1)+,r0        ; "filnam"
015636  004767  172326                  call    su$unp          ;
015642  112722  000056                  movb    #'.,(r2)+       ; "filnam."
015646  012100                          mov     (r1)+,r0        ; "filnam.typ"
015650  004767  172314                  call    su$unp          ;
015654  000207                          return
```

```
                                .sbttl  Driver Transfer function                                (driver)
   015656                       x$xdrv:

                                ;       Separate driver code exists for each XXDP-supported system device
                                ;
                                ;       dp.ufd  = 0     ;3      ; UFD directory start block (from init)
                                ;       dp.spc  = 2     ;"FN.T" ; space filled "filnam.typ"
                                ;       dpspc.  = 10.   ;       ; 10-char name (6+1+3)
                                ;       dp.ter  = 12    ;.word 0; 1-word zero terminator
                                ;       dpbbs.  = 14            ; block length

   015656                       d$rlow:
   015656  000003               d$pufd: .word   3       ;       ; first UFD/directory block (savm)(boot)
   015660                       d$pspc: .blkb   12.     ;       ; .asciz "filnam.typ"<0>
   015674  000252               d$pmon: .word   170.    ;252    ; first xmonitor block      (savm)(boot)
   015676  000000                       .word   0       ;       ; local
   015700  000000               d$ptwc: .word   0       ;       ; transaction word count

                                ;       DR$TRA - Driver transfer function
                                ;
                                ;       in      r5 ->   IOB
                                ;                       io.blk
                                ;                       io.buf
                                ;                       io.wct
                                ;                       dr.uni
                                ;
                                ;               call    @dr.tra(r5)
                                ;
                                ;       fine    r0/r1   unchanged
                                ;               r2..r4  burnt
                                ;
                                ;       fail    abort   "? RD ERR"
                                ;
                                ;       While the boot and the driver both support partial block reads,
                                ;       all monitor system device reads are for full blocks (see MO$CHN).

   015702  010046               dr$tra: mov     r0,-(sp)        ; we do our own thing
   015704  010146                       mov     r1,-(sp)        ;
   015706  011500                       mov     (r5),r0         ; r0 -> csr
   015710  105065  177777               clrb    dr.sts(r5)      ; assume happiness
   015714  004767  000772               call    du$res          ; reset dl:

   015720  005046                       clr     -(sp)           ; (sp) = result track
   015722  016503  000006               mov     io.blk(r5),r3   ; r3 = requested block
   015726  012702  000050               mov     #40.,r2         ; r2 = sectors-per-track
   015732  160203               10$:    sub     r2,r3           ; more tracks?
   015734  103402                       bcs     20$             ; oops - too far
   015736  005216                       inc     (sp)            ; another track
   015740  000774                       br      10$             ; loop
   015742  060203               20$:    add     r2,r3           ; backup from too far
   015744  005004                       clr     r4              ; r4 =
   015746  006202                       asr     r2              ; blocks-per-track now
   015750  160203               30$:    sub     r2,r3           ;
   015752  103402                       bcs     40$             ;
   015754  005204                       inc     r4              ;
   015756  000774                       br      30$             ;
   015760  060203               40$:    add     r2,r3           ;
```

```
015762  006303                          asl     r3              ;
015764  012701  000007                  mov     #7,r1           ; compute cylinder
015770  006316                  50$:    asl     (sp)            ; shift left
015772  005301                          dec     r1              ;
015774  001375                          bne     50$             ;
                                                                ;
015776  012667  016676'                 mov     (sp)+,16676     ; cylinder
016002  016546  000002                  mov     io.wct(r5),-(sp);

                                ;       Block loop
                                ;
                                ;       (sp)    running word count
                                ;       d$ptwc  transaction word count

016006  162716  000400          60$:    sub     #256.,(sp)      ; got more than a block?
016012  101404                          blos    70$             ; no
016014  012767  000400  177656          mov     #256.,d$ptwc    ; yes - transaction wct
016022  000405                          br      80$             ;
016024  011667  177650          70$:    mov     (sp),d$ptwc     ; no - restore
016030  062767  000400  177642          add     #256.,d$ptwc    ;
016036  010001                  80$:    mov     r0,r1           ; r1 -> csr
016040  062701  000006                  add     #6,r1           ; r1 -> wct
016044  016546  177776                  mov     dr.uni(r5),-(sp);
016050  000316                          swab    (sp)            ; (sp) = unit
016052  052716  000010                  bis     #dlRHD.,(sp)    ; read header
016056  012610                          mov     (sp)+,(r0)      ; issue function
016060  004767  000242                  call    du$wai          ;
016064  001111                          bne     140$            ;
                                                                ;
016066  011146                          mov     (r1),-(sp)      ; (sp) = wct =
016070  012741  000001                  mov     #1,-(r1)        ; adr: see$
016074  005704                          tst     r4              ;
016076  001402                          beq     90$             ;
016100  052711  000020                  bis     #20,(r1)        ; adr: hea$
016104  042716  000177          90$:    bic     #177,(sp)       ;
016110  166716  016676'                 sub     16676,(sp)      ;
016114  103005                          bcc     100$            ;
016116  005416                          neg     (sp)            ;
016120  042716  000177                  bic     #177,(sp)       ;
016124  052711  000004                  bis     #4,(r1)         ; adr: dir$
016130  052611                  100$:   bis     (sp)+,(r1)      ;
016132  042710  000016                  bic     #dlFUN$,(r0)    ; clear function bit field
016136  052710  000006                  bis     #dlSEE.,(r0)    ; set function to seek
016142  004767  000154                  call    du$opr          ; perform seek and wait
016146  001060                          bne     140$            ;

016150  032710  000001          110$:   bit     #1,(r0)         ; wait for drive ready
016154  001775                          beq     110$            ;
016156  016746  016676'                 mov     16676,-(sp)     ; cylinder
016162  050316                          bis     r3,(sp)         ; sector
016164  005704                          tst     r4              ; head flag
016166  001402                          beq     120$            ;
016170  052716  000100                  bis     #100,(sp)       ; head
016174  012621                  120$:   mov     (sp)+,(r1)+     ; adr
016176  016746  177476                  mov     d$ptwc,-(sp)    ;
016202  005416                          neg     (sp)            ;
016204  012611                          mov     (sp)+,(r1)      ; wct
```

```
016206  016560  000004  000002          mov     io.buf(r5),2(r0);
016214  042710  000016                  bic     #dlFUN$,(r0)    ;
016220  052710  000014                  bis     #dlREA.,(r0)    ; read data
016224  004767  000072                  call    du$opr          ;
016230  001027                          bne     140$            ;

016232  005716                          tst     (sp)            ; hows the word count?
016234  003421                          ble     135$            ; we are done
016236  062703  000002                  add     #2,r3           ; next block
016242  020327  000050                  cmp     r3,#40.         ; sectors-per-track
016246  002410                          blt     130$            ; still within track
016250  005003                          clr     r3              ; sector/block = 0
016252  005204                          inc     r4              ; switch head
016254  042704  177776                  bic     #177776,r4      ; isolate head flag
016260  001003                          bne     130$            ; positive
016262  062767  000200  016676'         add     #128.,16676     ; advance cylinder
016270  062765  001000  000004  130$:   add     #512.,io.buf(r5); advance buffer pointer
016276  000643                          br      60$             ;

                                ;       Transfer completed

016300  005726                  135$:   tst     (sp)+           ; dump temp
016302  012601                          mov     (sp)+,r1        ; restore
016304  012600                          mov     (sp)+,r0        ;
016306  000207                          return

                                ;       Transfer aborted

016310  105365  177777          140$:   decb    dr.sts(r5)      ; dr.sts = -1 - I/O error
016314  012700  016760'                 mov     #m$srer,r0      ; "? RD ERR" (abort routine relocates)
016320                                  JmpAbt                  ; abort

                                ;       DU$OPR - Initiate operation, wait and check errors
                                ;
                                ;       call    du$opr
                                ;       beq     fine
                                ;       bne     fail

016322  042710  000200          du$opr: bic     #dlGO$,(r0)     ; ready?
016326  032710  100200          du$wai: bit     #dlERR$!dlGO$,(r0) ; error|ready
016332  001775                          beq     du$wai          ; we wait a lot
016334  100401                          bmi     10$             ; awful
016336  000264                          sez                     ; wonderful
016340  000207                  10$:    return
```

```
                                .sbttl  Get Device, Open File, Restore Driver functions         (driver)

                                ;       DR$DEV - Get Device name/unit/media function
                                ;
                                ;       in      r5 ->   IOB
                                ;
                                ;               call    dr.dev(r5)
                                ;
                                ;       out     r0 ->   d$rdev: drTdev structure
                                ;
                                ;       Translate the dr.uni ordinal to dv.uni ascii
                                ;       Called by GetDev service and the Enable command

016342  116500  177776          dr$dev: movb    dr.uni(r5),r0   ; unit ordinal
016346  062700  000060                  add     #'0,r0          ; asciify it
016352  110067  000012                  movb    r0,d$rdev+dv.uni; store past "DL"
016356  010700                          mov     pc,r0           ;
016360  062700  000006                  add     #d$rdev-.,r0    ; point to it
016364  000207                          return                  ;
                                                                ;
016366     104     114          d$rdev: .ascii  "DL"    ;0      ;\ dv.nam - driver device name ("DL")
016370     000                          .ascii  ""<0>   ;2      ;| dv.uni - driver device unit ("0")
016371     014                          .byte   dvRL1.  ;3 14   ;/ dv.med - driver device media code


                                ;       DR$OPN - Open file function
                                ;
                                ;       No status is sent back to the caller because only
                                ;       succesful opens return. Failed opens issue abort.
                                ;
                                ;       in      io.spc  .asciz "filnam.typ"
                                ;
                                ;               call    dr.opn(r5)
                                ;
                                ;       out     r0      burnt
                                ;               r1 ->   .asciz "filnam.typ"
                                ;               dr.fnm  .rad50 /filnamtyp
                                ;               di.sbl  .word   n       ; start block file
                                ;
                                ;       fail    abort   "? NOT FOUND filnam.typ"
                                ;               r0 =    0

016372  004767  000024          dr$opn: call    du$loo          ; lookup
016376  010103                          mov     r1,r3           ; r1 -> entry filnamtyp
016400  010502                          mov     r5,r2           ; copy filnamtyp and first block
016402  062702  177756                  add     #dr.fnm,r2      ; r5 -> d$rfnm
016406  012322                          mov     (r3)+,(r2)+     ; dr.fil
016410  012322                          mov     (r3)+,(r2)+     ; dr.nam
016412  012322                          mov     (r3)+,(r2)+     ; dr.typ
016414  016312  000004                  mov     4(r3),(r2)      ; dr.sbl - first file block
016420  000207                          return


                                ;       DU$LOO - Lookup file
                                ;
                                ;       in      io.spc  -> .asciz "filnam.typ"
                                ;
                                ;       fine    r1 ->   directory entry: .rad50 /filnamtyp/
                                ;       fail    abort   "? NOT FOUND"

016422  105065  177777          du$loo: clrb    dr.sts(r5)      ; reset errors
016426  004767  000154                  call    du$mfd          ; MFD -> directory
016432  017565  000010  000006          mov     @io.ufd(r5),io.blk(r5)
016440  005003                          clr     r3              ;
016442                                  ReaBlk                  ; read directory
016444  005065  177754                  clr     dr.ent(r5)      ;
                                                                ; Block Loop
016450  016504  177752          10$:    mov     dr.buf(r5),r4   ; r4 -> buffer
016454  005724                          tst     (r4)+           ; r4 -> buffer record (skip next block link)
016456  062703  000034                  add     #28.,r3         ; 28. entries per block
                                                                ; Entry Loop
016462  005265  177754          20$:    inc     dr.ent(r5)      ; next (or first) entry
016466  026503  177754                  cmp     dr.ent(r5),r3   ; more entries?
016472  101405                          blos    30$             ; yes
016474  005365  177754                  dec     dr.ent(r5)      ; no - backup
016500                                  ReaNxt                  ; read next directory block
016502  000423                           br     60$             ; end of file
016504  000761                          br      10$             ; restart block loop
016506  005714                  30$:    tst     (r4)            ; empty/deleted entry?
016510  001413                          beq     40$             ; affirmative
016512  010702                          mov     pc,r2           ; convert rad50 directory entry
016514  062702  177144                  add     #d$pspc-.,r2    ; r2 -> driver spec ascii buffer
016520  010401                          mov     r4,r1           ; r1 -> .rad50 /filnamtyp/
016522                                  SpcAsc                  ; unradify
016524  010500                          mov     r5,r0           ;
016526  062700  000012                  add     #io.spc,r0      ; r0 -> I/O ascii spec
016532                                  CmpSpc                  ; and the verdict is?
016534  000401                           br     40$             ; missmatch
016536  000403                          br      50$             ; match - we are done
016540  062704  000022          40$:    add     #18.,r4 ;22     ; next entry
016544  000746                          br      20$             ; and off we go again
016546  010401                  50$:    mov     r4,r1           ; r1 -> result ascii filespec
016550  000207                          return

                                ;       File not found message and abort

016552  105265  177777          60$:    incb    dr.sts(r5)      ; dr.sts = 1 - file not found error
016556  010700                          mov     pc,r0           ;
016560  162700  016560'                 sub     #.,r0           ; r0 = monitor origin
016564  062700  016742'                 add     #m$sfnf,r0      ; "? NOT FOUND"
016570                                  TypBrk                  ;
016572  010500                          mov     r5,r0           ;
016574  062700  000012                  add     #io.spc,r0      ; .asciz "filnam.typ"
016600                                  TypBrk                  ; "? NOT FOUND filnam.typ"
016602  005000                          clr     r0              ; no message
016604                                  JmpAbt                  ; begone


                                ;       DU$MFD - Read MFD block
                                ;
                                ;       Called by du$loo and dr$rst
                                ;
                                ;       out     d$pufd = UFD start block
                                ;               d$pmon = monitor start block
                                ;
                                ;       This routine is only required for disks which have variable
                                ;       disk locations for the UFD and/or monitor. This is true for
                                ;       DL: but not for DK:.

016606  012765  000001  000006  du$mfd: mov     #1,io.blk(r5)   ; MFD block
016614                                  ReaBlk                  ; read it
016616  016500  000004                  mov     io.buf(r5),r0   ; get the input buffer
016622  005720                          tst     (r0)+           ; skip the block linkage
016624  016067  000000  177024          mov     mf.ufd(r0),d$pufd ; first UFD block  (mov 0(r0),...)(note)
016632  016067  000024  177034          mov     mf.mon(r0),d$pmon ; first monitor block
016640  000207                          return


                                ;       DR$RST - Restore monitor function
                                ;
                                ;       XXDP needs a special function to read the monitor disk image
                                ;       because it is a contiguous file.
                                ;
                                ;       in      io.wct  = word count
                                ;               io.buf  = store address
                                ;               io.blk  = monitor relative block
                                ;                         d$pmon is the monitor base block
                                ;
                                ;               call    dr.rst(r5)
                                ;
                                ;       out     io.buf = restored area
                                ;
                                ;       DR$RST reads the MFD to get d$pmon (which ENABLE can modify)
                                ;       The assumption is that an ENABLED disk has the same monitor version

016642  016546  000002          dr$rst: mov     io.wct(r5),-(sp); save context
016646  016546  000004                  mov     io.buf(r5),-(sp);
016652  016546  000006                  mov     io.blk(r5),-(sp); monitor-relative block
016656  004767  177724                  call    du$mfd          ; get mfd block
016662  012665  000006                  mov     (sp)+,io.blk(r5); restore
016666  012665  000004                  mov     (sp)+,io.buf(r5);
016672  012665  000002                  mov     (sp)+,io.wct(r5);
                                                                ;
016676  066765  176772  000006          add     d$pmon,io.blk(r5); relocate monitor block
016704  004775  177772                  call    @dr.tra(r5)     ; transfer
016710  000207                          return


                                ;       DU$RES - Device reset

016712  016546  177776          du$res: mov     dr.uni(r5),-(sp); .byte unit, function
016716  000316                          swab    (sp)            ;
016720  052716  000004                  bis     #dlSTA.,(sp)    ; get status
016724  052760  000013  000004          bis     #dlREP$,dl.adr(r0) ; reset, get status
016732  012610                          mov     (sp)+,(r0)      ; issue disk function
016734  004767  177366                  call    du$wai          ; du$wai
016740  000207                          return

                                ;       Driver error messages

016742     077     040     116  m$sfnf: .asciz  "? NOT FOUND: "
```

```
        016745     117     124     040
        016750     106     117     125
        016753     116     104     072
        016756     040     000
    172 016760     077     040     122  m$srer: .asciz  "? RD ERR"<cr><lf>
        016763     104     040     105
        016766     122     122     015
        016771     012     000
    173                                         .even
    174
    175
    176                                 ;       Monitor end
    177
    178 016774                                  .blkw   2               ; round-up driver and monitor
    179                                                                 ;
    180 017000  000001                  x$xtop: .end                    ; 20000
```

```
BALEN.= 001414          B$OCYL  000034          DLDIR$= 000004          DV.MED= 000003          IN$DAT  003716R     002
BA$ABT  005036R     002 B$OHEA  000037          DLERR$= 100000          DV.NAM= 000000          IN$ENG  002340R     002
BA$CMD  005012R     002 B$OMFD= 001002          DLFUN$= 000016          DV.UNI= 000002          IN$FIN  003634R     002
BA$DIS  005216R     002 B$OMON= 001026          DLGO$ = 000200          D$PMON  015674R     002 IN$HDW  003510R     002
BA$ENG  005002R     002 B$OSEC  000036          DLHDS.= 000002          D$PSPC  015660R     002 IN$HGH  003026R     002
BC$CHN  005714R     002 B$OTRK  000026          DLHEA$= 000020          D$PTWC  015700R     002 IN$IOB  004660R     002
BC$CMI  006304R     002 B$OTWC  000446          DLMRK$= 000001          D$PUFD  015656R     002 IN$PDT  004244R     002
BC$ENB  006366R     002 B$OWCT  000030          DLNOP.= 000000          D$RCOM  014600R     002 IN$SIZ  002610R     002
BC$END  006212R     002 CBIT  = 000001          DLRDX.= 000016          D$RCSR  014626RG    002 IOB     014626R     002
BC$GTO  005742R     002 CLAVL.= 000052          DLRDY$= 000001          D$RDEV  016366R     002 IO.BLK= 000006
BC$IER  006264R     002 CLLEN.= 000054          DLREA.= 000014          D$RDIS  014614R     002 IO.BUF= 000004
BC$IFT  006146R     002 CLOFIL  010166R     002 DLREP$= 000013          D$REND  014640R     002 IO.SPC= 000012
BC$ILM  006244R     002 CL$ABT  011000R     002 DLRHD.= 000010          D$RFNM  014604R     002 IO.UFD= 000010
BC$LOA  005362R     002 CL$CHN  011476R     002 DLRL2$= 000200          D$RIOB  014626R     002 IO.WCT= 000002
BC$PRT  006112R     002 CL$CMD  011040R     002 DLRL2.= 000002          D$RLOW  015656R     002 I$MBOO  003765R     002
BC$QUI  006104R     002 CL$DIR  012026R     002 DLRST$= 000010          D$RUNI  014624RG    002 I$MDAT  004043R     002
BC$QUT  005030R     002 CL$DIS  011112R     002 DLSEE$= 000001          EM$ENG  012520R     002 I$MENT  004010R     002
BC$RUN  005670R     002 CL$ENB  012000R     002 DLSEE.= 000006          EM$RST  012474R     002 I$MMON  004744R     002
BC$SMI  006274R     002 CL$ENG  011010R     002 DLSIZ.= 024000          ENBBS = 000024          I$MNON  004062R     002
BC$STA  005446R     002 CL$FIL  011740R     002 DLSTA.= 000004          EN.DAT= 000006          I$MRAD  003744R     002
BC$WAI  006062R     002 CL$HLP  011650R     002 DLSTS$= 000002          EN.FFB= 000012          I$MUBQ  004107R     002
BOWCT.= 007400          CL$IFN  011262R     002 DLTRK$= 177600          EN.FIL= 000000          I$MUBS  004067R     002
BO$ADR  000374          CL$LOA  012064RG    002 DLTRK.= 000100          EN.FLG= 000022          I$MXDP  004163R     002
BO$CHK  000356          CL$RUN  012352R     002 DLUNI$= 001400          EN.LEN= 000016          I$NAUT  000002R     002
BO$CON  000022          CL$STA  012154R     002 DLUN$M= 176377          EN.LST= 000020          I$NKWD  003020R     002
BO$CYL  000310          CL$TST  011440R     002 DL.ADR= 000004          EN.NAM= 000002          I$NKWS  002726R     002
BO$ENG  000044          CL.LEN= 000002          DL.BUF= 000002          EN.STA= 000014          I$NREL  002670R     002
BO$GEO  000676          CL.PTR= 000000          DL.CSR= 000000          EN.TYP= 000004          I$NR1 = 003170R     002
BO$HLT  000152          CMPSPC  015564R     002 DL.DAT= 000006          E$MDIS  012606R     002 I$NR10= 003454R     002
BO$NXT  000500          CR    = 000015          DL.WCT= 000006          F$IBCT  013350R     002 I$NR11= 004736R     002
BO$OPR  000334          CTRLC = 000003          DL1SZ.= 024000          F$IBUF  013532R     002 I$NR12= 003124R     002
BO$PRI  000000          CTRLQ = 000021          DL2SZ.= 050000          F$ILCK  013356R     002 I$NR13= 015520R     002
BO$RES  000450          CTRLS = 000023          DR$DEV  016342R     002 F$INXT  013532R     002 I$NR14= 015544R     002
BO$SEE  000560          CTRLU = 000025          DR$OPN  016372R     002 F$IPOS  013366R     002 I$NR2 = 015262R     002
BO$WAI  000340          CTRLX = 000030          DR$RST  016642R     002 F$IPTR  013354R     002 I$NR3 = 003470R     002
BU$ACT  005526R     002 CTRLZ = 000032          DR$TRA  015702R     002 F$IRCK  013412R     002 I$NR4 = 004502R     002
BU$ERR  006234R     002 CU$ACT  012230R     002 DR.BUF= 177752          F$IREC  013534R     002 I$NR5 = 004662R     002
BU$EXI  005646R     002 CU$CHN  011532R     002 DR.CSR= 000000          F$ISCK  013414R     002 I$NR6 = 014724R     002
BU$FAL  006206R     002 CU$LOA  012070R     002 DR.DEV= 177774          F$ISVP  013370R     002 I$NR7 = 014664R     002
BU$LOA  005366R     002 CU$QVS  011604R     002 DR.ENT= 177754          GETAVL  007566R     002 I$NR8 = 014730R     002
BU$NEW  006120R     002 CU$RET  011646R     002 DR.FNM= 177756          GETCHK  007656R     002 I$NR9 = 015044R     002
BU$NOP  006242R     002 CU$STA  012164R     002 DR.OPN= 177766          GETCOM  010766R     002 I$N50H  000000R     002
BU$PR7  006350R     002 CU$SWI  012420R     002 DR.RST= 177770          GETDAT  010332R     002 I$OSPC  014640R     002
BU$RET  006314R     002 C$LAUT  011110R     002 DR.SBL= 177764          GETDEV  015500R     002 JMPABT  015474R     002
BU$STA  005454R     002 C$LBUF  013432R     002 DR.STS= 177777          GETDRV  015516R     002 KWP   = 172540
BU$TRU  006160R     002 C$LDIS  011400R     002 DR.TRA= 177772          GETLIN  007130R     002 LF    = 000012
B$ADIS  005054R     002 C$LFLD  013372R     002 DR.UNI= 177776          GF$SEC  007466R     002 LOAFIL  014654R     002
B$AEND  006142R     002 C$LLEN  013362R     002 DT$DAY  004432R     002 GL$SEC  007402R     002 LOASUP  010450R     002
B$AFNM  013520R     002 C$LLIN  013360R     002 DT$MON  004464R     002 HBBAS.= 001000          LPT   = 177514
B$ALOO  005116R     002 C$LLOO  011353R     002 DT$YEA  004622R     002 HBBLK.= 000001          LPTMOD  010410R     002
B$AREG  005000R     002 C$LNXT  013364R     002 DU$LOO  016422R     002 HB.MON= 000026          LTC   = 177546
B$ASFN  013506R     002 C$LTST  011420R     002 DU$MFD  016606R     002 HB.NXT= 000000          MAVAL.= 000012
B$ASTK  005000R     002 C$SQVS  011472R     002 DU$OPR  016322R     002 HB.UFD= 000002          MF.MON= 000024
B$ATHN  006136R     002 DEL   = 000177          DU$RES  016712R     002 HT    = 000011          MF.UFD= 000000
B$OBLK  000032          DLBPT.= 000024          DU$WAI  016326R     002 H$WKWP  013332R     002 MOBAT.= 000006
B$OBUF  000444          DLCSR.= 174400          DVRK5.= 000002          H$WLTC  013330R     002 MOCLI.= 000012
B$OCSR  000020          DLCYL.= 000400          DVRL1.= 000014          IN$CON  002626R     002 MOOVL.= 001414
```

```
Symbol table

MOTRA.= 000010          PSW    = 177776          SYLTC$= 000001          S$YPER  013326R     002 TKS   = 177560
MO$CHN  013052R     002 PUTCHA  015322R     002 SYNUB$= 000010          S$YPGS  014574R     002 TPB   = 177566
MO$REA  012762R     002 PUTCHK  007554R     002 SY50H$= 000020          S$YPND  013425R     002 TPS   = 177564
MO$RST  012720R     002 PUTTAB  007742R     002 S$YABT  013400R     002 S$YPOP  013426R     002 TYPBRK  007534R     002
M$SADR  011337R     002 P$SCSU  014624R     002 S$YACT  013410R     002 S$YQUI  013427R     002 TYPMON  007504R     002
M$SBER  005354R     002 RB$CHK  015436R     002 S$YBAT  014572R     002 S$YQVS  014570R     002 TYPMSG  007512R     002
M$SCHK  015142R     002 REABLK  015416R     002 S$YCFG  014542R     002 S$YREL  013334R     002 T$ENEW  007726R     002
M$SCMD  011272R     002 REABYT  015224R     002 S$YCOL  013424R     002 S$YRPT  013336R     002 V$EBUS= 000004
M$SCON  015152R     002 REANXT  015376R     002 S$YCOM  014534R     002 S$YRUN  013421R     002 V$ECPU= 000010
M$SFNF  016742R     002 REAWRD  015204R     002 S$YCSR  014534R     002 S$YSTA  013406R     002 V$EEMT= 000030
M$SFNM  011314R     002 RPTFLD  015506R     002 S$YDAT  013376R     002 S$YSTK  013324R     002 V$EKWP= 000104
M$SRER  016760R     002 SCMAN$= 000001          S$YDEV  013340R     002 S$YSUP  013342R     002 V$ELTC= 000100
NEWLIN  007732R     002 SETABT  015466R     002 S$YEMT  013402R     002 S$YSWI  013110R     002 XX$RST  011010R     002
OCTASC  010340R     002 SETLIN  010270R     002 S$YERR  014576R     002 S$YTOP  013352R     002 X$XBAT  005000R     002
OPNFIL  010054R     002 SPACE = 000040          S$YGTO  013140R     002 S$YTPB  013346R     002 X$XDRV  015656R     002
OVLEN.= 001414          SPCASC  015626R     002 S$YHLT  013422R     002 S$YTPS  013344R     002 X$XGAP  000004R     002
O$VCLI  011000R     002 SP.PC = 000006          S$YKTK  014566R     002 S$YTRA  013324R     002 X$XINI  000000R     002
O$VREG  011000R     002 SP.PS = 000010          S$YKWD  014546R     002 S$YUNI  014540R     002 X$XLOW  000000
PADTER  010606R     002 SP.R2 = 000000          S$YKWP  014560R     002 S$Y5CK  013416R     002 X$XPER  011000R     002
PARDEC  010532R     002 SP.R3 = 000002          S$YLOA  013420R     002 TERMOD  010432R     002 X$XSTA  013002R     002
PARFLD  007412R     002 SP.R4 = 000004          S$YLPT  014544R     002 TE$CTC  007014R     002 X$XTOP  017000R     002
PAROCT  007762R     002 SU$UNP  010170R     002 S$YLTC  014550R     002 TE$CTL  007024R     002 X$XTRA  007000R     002
POPBAT  010624R     002 SYKWP$= 000002          S$YLTK  014556R     002 TE$PUT  007000R     002 $$    = 004000
PSHBAT  010630R     002 SYLPT$= 000004          S$YPAD  013423R     002 TKB   = 177562

. ABS.  001000    000   (RW,I,GBL,ABS,OVR)
        000000    001   (RW,I,LCL,REL,CON)
XXDP    017
```