```
SERIES-III PL/M-86 X198
OBJECT MODULE PLACED IN :F1:BNUMFD.OBJ
COMPILER INVOKED BY:  PLM86.86 :F1:BNUMFD.P86 XREF PRINT(:F5:BNUMFD.LST)


              $title('numbd.p86')

              /*
               *  TITLE:      bnumfd.p86  SYSBOOT Numbered File Driver
               *
               *  DATE:       3-8-82
               *
               *  ABSTRACT:   Contains setup$file and read$file which implement
               *              reading a file on a numbered file system (used for
               *              named) for the System Bootstrap Loader.
               *
               *  NOTE: lifted from second stage bootstrap loader
               *
               *  LANGUAGE DEPENDENCIES: PLM/86 COMPACT
               */

              $include(:f1:bprop.asm)
  1       =   ;/*
*** ERROR 42 IN 1 (:F1:BPROP.ASM, LINE 1): INSERTED: " <ID> : DO"
          =   ; * INTEL CORPORATION PROPRIETARY INFORMATION.  THIS LISTING IS
          =   ; * SUPPLIED UNDER THE TERMS OF A LICENSE AGREEMENT WITH INTEL
          =   ; * CORPORATION AND MAY NOT BE COPIED NOR DISCLOSED EXCEPT IN
          =   ; * ACCORDANCE WITH THE TERMS OF THAT AGREEMENT.
          =   ; */

  2   1       bnumfd: DO;

              $include(:f1:common.lit)
          =       /*
          =        *      Oft-used literals.
          =        *
          =        * Written by Bob Beck, 25-MAY-79.
          =        *
          =        * lifted from second stage bootstrap loader
          =        */

  3   2   =       DECLARE
          =           TRUE            LITERALLY '0FFH',
          =           FALSE           LITERALLY '000H',
          =           FOREVER         LITERALLY 'WHILE TRUE',
          =           PTR$OVERLAY     LITERALLY 'STRUCTURE(offset WORD, base WORD)',
*** ERROR 42 IN 3 (:F1:COMMON.LIT, LINE 14), NEAR 'PTROVERLAY': INSERTED: " <ID>"
*** ERROR 41 IN 3 (:F1:COMMON.LIT, LINE 14), NEAR 'PTROVERLAY': DELETED: " LITERALLY <STRING>"
          =           DWORD           LITERALLY 'POINTER';
              $include(:f1:fnode.lit)
          =       /*
          =        *      Define disk fnode.
          =        */

  4   2   =       DECLARE
          =           FNODE$PTRS$SIZE LITERALLY '40';     /* 8 pointers */
```

```
 5   2  =       DECLARE
        =           MAX$ACCESSORS   LITERALLY '3',
        =           ACCESSOR$SIZE   LITERALLY '9';          /* 3 accessors */

        =       /*
        =        * The fnode structure is that of the ERS.
        =        */

 6   2  =       DECLARE
        =           FNODE$COMMON$1 LITERALLY
        =              'flags        WORD,
        =               type         BYTE,
        =               gran         BYTE,
        =               owner        WORD,
        =               cr$time      DWORD,
        =               access$time  DWORD,
        =               mod$time     DWORD,
        =               total$size   DWORD,
        =               total$blks   DWORD,
        =               ptr(FNODE$PTRS$SIZE)    BYTE';

 7   2  =       DECLARE
        =           FNODE$COMMON$2 LITERALLY
        =              'this$size    DWORD,
        =               ext$fnode    WORD,
        =               prev$fnode   WORD,
        =               id$count     WORD,
        =               accessor(ACCESSOR$SIZE) BYTE';

 8   2  =       DECLARE
        =           FNODE$COMMON$INFO LITERALLY 'FNODE$COMMON$1,FNODE$COMMON$2';

 9   2  =       DECLARE
        =           NUM$DISK$FNODE LITERALLY 'STRUCTURE(
        =              FNODE$COMMON$INFO,
        =              aux(1)       BYTE)';

10   2  =       DECLARE
        =           NAMED$DISK$FNODE LITERALLY 'STRUCTURE(
        =              FNODE$COMMON$INFO,
        =              parent       WORD,
        =              aux(1)       BYTE)';

        =       /*
        =        * Fnode flags bits (again, from the ERS).
        =        */

11   2  =       DECLARE
        =           FN$LONG     LITERALLY '00002H';

        $include(:f1:named.lit)
        =   $subtitle('Literals')
```

```
          =       /*
          =        *  Constants lifted from the second stage bootstrap loader
          =        */

  12   2  =       DECLARE                         /* Scan Path Constants */
          =           CR                  LITERALLY '0DH',    /* Carriage Return */
          =           LF                  LITERALLY '0AH';    /* Line Feed */

  13   2  =       DECLARE                         /* Loader Constants */
          =           MAX_PIDATA          LITERALLY '1024',   /* PIDATA staging area size */
          =           ST_ADDR_PRESENT     LITERALLY '40H',    /* Start address presence bit */
          =           LIBHED              LITERALLY '0A4H',   /* Record Types */
          =           PEDATA              LITERALLY '084H',
          =           PIDATA              LITERALLY '086H',
          =           LEDATA              LITERALLY '0A0H',
          =           MODEND              LITERALLY '08AH';

  14   2  =       DECLARE                     /* Structure to manipulate pointer to byte */
          =           BYTE_PTR_STRUCT     LITERALLY 'STRUCTURE(
          =               byt     BYTE,
          =               next    BYTE)';
          =   $include(:f1:dpsupp.ext)
          =   $save nolist
*** ERROR 49 IN 16 (:F1:DPSUPP.EXT, LINE 12): NOT AT MODULE LEVEL
  16   3  =           DECLARE
*** ERROR 49 IN 19 (:F1:DPSUPP.EXT, LINE 18): NOT AT MODULE LEVEL
  19   3  =           DECLARE dwa DWORD, swb WORD;
*** ERROR 49 IN 22 (:F1:DPSUPP.EXT, LINE 22): NOT AT MODULE LEVEL
  22   3  =           DECLARE
*** ERROR 49 IN 25 (:F1:DPSUPP.EXT, LINE 28): NOT AT MODULE LEVEL
  25   3  =           DECLARE dwa DWORD, swb WORD;
*** ERROR 49 IN 28 (:F1:DPSUPP.EXT, LINE 32): NOT AT MODULE LEVEL
  28   3  =           DECLARE dwa DWORD, swb WORD;
*** ERROR 49 IN 31 (:F1:DPSUPP.EXT, LINE 55): NOT AT MODULE LEVEL
  31   3  =           DECLARE (dwa, dwb) DWORD;
*** ERROR 49 IN 34 (:F1:DPSUPP.EXT, LINE 59): NOT AT MODULE LEVEL
  34   3  =           DECLARE (swa, swb) WORD;
*** ERROR 49 IN 37 (:F1:DPSUPP.EXT, LINE 63): NOT AT MODULE LEVEL
  37   3  =           DECLARE
*** ERROR 49 IN 40 (:F1:DPSUPP.EXT, LINE 69): NOT AT MODULE LEVEL
  40   3  =           DECLARE
*** ERROR 49 IN 43 (:F1:DPSUPP.EXT, LINE 75): NOT AT MODULE LEVEL
  43   3  =           DECLARE

                    /*
                     *  Declare long_call appropriately for use in calling
                     *  the device driver.
                     *  Long_call calls its last argument as a FAR procedure but
                     *  is callable from compact.
                     */

  45   2      long_call: PROCEDURE(unit, blk_num, buf_p, dev_proc) EXTERNAL;
*** ERROR 49 IN 46 (LINE 34): NOT AT MODULE LEVEL
  46   3              DECLARE
                          unit        WORD,
```

```
                        blk_num     DWORD,
                        buf_p       POINTER,
                        dev_proc    POINTER;
  47    3           END long_call;

                    /*
                     *  Error$halt halts WITHOUT enabling interrupts
                     */

  48    2       error_halt: PROCEDURE EXTERNAL;
*** ERROR 49 IN 49 (LINE 46): NOT AT MODULE LEVEL
  49    3           END error_halt;
              $subtitle('DWORD Overlay')
```

```
50   2          DECLARE
                    DWORD_OVERLAY   LITERALLY 'STRUCTURE(
                        lo      WORD,
                        hi      WORD)';
           $subtitle('Second Stage Static Data')
```

```
                /*
                 *  Static data variables and buffers
                 */

  51   2        DECLARE                          /* Read File data */
*** ERROR 49 IN 51 (LINE 60): NOT AT MODULE LEVEL
                   actual          WORD EXTERNAL,      /* Number of bytes read from read$file */
*** ERROR 49 IN 51 (LINE 61): NOT AT MODULE LEVEL
                   fnode_size      WORD EXTERNAL,      /* Fnode size */
*** ERROR 49 IN 51 (LINE 62): NOT AT MODULE LEVEL
                   device_driver   POINTER EXTERNAL,   /* Pointer to device$read */
*** ERROR 49 IN 51 (LINE 63): NOT AT MODULE LEVEL
                   dev_gran        WORD EXTERNAL,      /* Device granularity in bytes */
*** ERROR 49 IN 51 (LINE 64): NOT AT MODULE LEVEL
                   unit            WORD EXTERNAL,      /* Unit number */
*** ERROR 49 IN 51 (LINE 65): NOT AT MODULE LEVEL
                   vol_blks        WORD EXTERNAL,      /* Volume gran. in device blocks */
*** ERROR 49 IN 51 (LINE 66): NOT AT MODULE LEVEL
                   fnode_file_block    DWORD EXTERNAL, /* First block of fnode file */
                   fnode_ptr_index WORD,       /* Index into fnode block pointers */
*** ERROR 49 IN 51 (LINE 68): NOT AT MODULE LEVEL
                   short_block     DWORD PUBLIC,       /* First level block number */
                   short_block_o   DWORD_OVERLAY AT(@short_block),
                   short_blks      WORD,       /* First level block count */
                   indir_ptr_index WORD,       /* Index into indirect block pointers */
*** ERROR 49 IN 51 (LINE 72): NOT AT MODULE LEVEL
                   long_block      DWORD PUBLIC,       /* Second level block number */
                   long_block_o    DWORD_OVERLAY AT(@long_block),
                   long_blks       WORD,       /* Second level block count */
                   loop_type       WORD,       /* Short/Long loop control */
                   vol_block       WORD,       /* Counts device blocks in volume block */

                   /*
                    *  Next buffers must be in this order
                    */

                   fnode           NUM$DISK$FNODE, /* Stores current fnode */
                   fnode_ptr(8)    STRUCTURE(                  /* fnode block pointers */
                       num_blocks      WORD,
                       block_lo        WORD,
                       block_hi        BYTE) AT(@fnode.ptr),
                   fnode_total_size_o  DWORD_OVERLAY AT(@fnode.total$size),
*** ERROR 49 IN 51 (LINE 88): NOT AT MODULE LEVEL
                   indir_block(1)  BYTE PUBLIC,    /* current indirect block */
                   indir_ptr(1)    STRUCTURE(                  /* indirect block pointers */
                       num_blocks      BYTE,
                       block_lo        WORD,
                       block_hi        BYTE) AT(@indir_block),
*** ERROR 49 IN 51 (LINE 93): NOT AT MODULE LEVEL
                   data_block_p    POINTER EXTERNAL,   /* pointer to current data block */
                   data_block      BASED data_block_p (1) BYTE;    /* current data block */
            $subtitle('Read File')
```

```
                    /*
                     *  TITLE:      read$file - Read next block of a file
                     *
                     *  INTERFACE VARIABLES:
                     *              data_buffer_p       pointer to buffer to receive block
                     *                                  of data
                     *
                     *              Uses this module's static data to remember position
                     *              in file.
                     *
                     *  CALLS:      device driver via long_call, error_halt
                     *
                     *  ABSTRACT:   Reads next block of RMX 86 numbered file.
                     *              This whole procedure is one execution of a big loop
                     *              to read a short file or indirect blocks, with an
                     *              inner loop to read data blocks of a long file.
                     *              'loop_type' tells which loop to execute.
                     */

  52   2       read$file: PROCEDURE(data_buffer_p) REENTRANT PUBLIC;

*** ERROR 49 IN 53 (LINE 118): NOT AT MODULE LEVEL
  53   3                  DECLARE
                              data_buffer_p       POINTER;
  54   3                  DECLARE
                              buffer_p            POINTER;    /* temp buffer pointer */

                          /*
                           *  Check for running off end of file
                           */

  55   3                  IF fnode.total$size = 0 THEN
  56   3                      CALL error_halt;

  57   3                  IF loop_type <> 0 THEN
  58   3                      GOTO inner$loop;

                          /*
                           *  If file is long, read indirect block; else read data block
                           */

  59   3                  IF (fnode.flags AND FN$LONG) = 0 THEN
  60   3                      buffer_p = data_buffer_p;
  61   3                  ELSE
                              buffer_p = @indir_block;

  62   3                  IF DSCMP(SHORT_BLOCK,29A8H) = 0 THEN
  63   3                      CAUSE$INTERRUPT(3);
  64   3                  CALL long_call(unit, short_block, buffer_p, device_driver);
  65   3                  short_block = dsadd(short_block, 1);

  66   3                  IF (fnode.flags AND FN$LONG) = 0 THEN
  67   3                      GOTO no$long;
```

```
                    /*
                     *  Setup variables for this indirect block
                     */

 68   3                 loop_type = 1;
 69   3                 indir_ptr_index = 0;
 70   3                 long_block_o.lo = indir_ptr(0).block_lo;
 71   3                 long_block_o.hi = indir_ptr(0).block_hi;
 72   3                 long_block = dsmul(long_block, vol_blks);

 73   3       inner$loop:
                        CALL long_call(unit, long_block, data_buffer_p, device_driver);
 74   3                 long_block = dsadd(long_block, 1);
 75   3                 IF vol_block = vol_blks THEN
 76   3                     DO;
 77   4                         long_blks = long_blks + 1;
 78   4                         IF long_blks = indir_ptr(indir_ptr_index).num_blocks THEN
 79   4                             DO;
 80   5                                 indir_ptr_index = indir_ptr_index + 1;
 81   5                                 long_blks = 0;
 82   5                                 long_block_o.lo = indir_ptr(indir_ptr_index).block_lo;
 83   5                                 long_block_o.hi = indir_ptr(indir_ptr_index).block_hi;
 84   5                                 long_block = dsmul(long_block, vol_blks);

                                       /*
                                        * Have gone to new indirect entry. Check for end
                                        * of device block.
                                        */

 85   5                                 IF SHL(indir_ptr_index,2) = dev_gran THEN
 86   5                                     loop_type = 0;
 87   5                             END;
 88   4                     END;

 89   3       no$long:
                        IF vol_block = vol_blks THEN
 90   3                     DO;
 91   4                         short_blks = short_blks + 1;
 92   4                         vol_block = 0;
 93   4                     END;
 94   3                 vol_block = vol_block + 1;


                        /*
                         *  Check for end of blocks pointed to by this fnode block pointer
                         */

 95   3                 IF short_blks = fnode_ptr(fnode_ptr_index).num_blocks THEN
 96   3                     DO;
 97   4                         fnode_ptr_index = fnode_ptr_index + 1;
 98   4                         short_blks = 0;
 99   4                         loop_type = 0;
100   4                         short_block_o.lo = fnode_ptr(fnode_ptr_index).block_lo;
101   4                         short_block_o.hi = fnode_ptr(fnode_ptr_index).block_hi;
102   4                         short_block = dsmul(short_block, vol_blks);
103   4                     END;
```

```
                    /*
                     *  Set actual and update fnode size field to indicate
                     *  bytes left in file
                     */

 104   3              IF dscmp(fnode.total$size, dev_gran) > 0 THEN
 105   3                  actual = dev_gran;
 106   3              ELSE
                          actual = fnode_total_size_o.lo;
 107   3              fnode.total$size = dssub(fnode.total$size, actual);

 108   3          END read$file;
          $subtitle('Setup File')
```

```
                    /*
                     *  TITLE:      setup$file - prepare to read file by fnode number
                     *
                     *  INTERFACE VARIABLES:
                     *              fnode_num       Number of file to set up
                     *
                     *              Leaves setup of file in this module's static data for
                     *              use by read$file.
                     *
                     *  CALLS:      device driver via long_call, read$file, setup$file
                     *
                     *  ABSTRACT:   For fnode 0, reads directly from disk.  For any other,
                     *              sets up fnode 0, does a partial seek, then reads fnodes
                     *              file to desired fnode.  Sets up that fnode.
                     */

 109   2      setup$file: PROCEDURE(fnode_num) REENTRANT PUBLIC;

*** ERROR 49 IN 110 (LINE 240): NOT AT MODULE LEVEL
 110   3              DECLARE
                          fnode_num       WORD;

 111   3              DECLARE
                          fnode_off       DWORD,      /* offset of fnode in fnodes file */
                          fnode_off_o     DWORD_OVERLAY AT(@fnode_off),
                          dtmp            DWORD;      /* double word temporary */

 112   3              IF fnode_num = 0 THEN
 113   3                  DO;

                              /*
                               *  Read fnode 0 directly
                               */

 114   4                      CALL long_call(unit, fnode_file_block, @fnode, device_driver);

 115   4                  END;
 116   3              ELSE
                          DO;

                              /*
                               *  Read fnodes file (file 0) up to this fnode
                               */

 117   4                      CALL setup$file(0);
 118   4                      fnode_off = dssmul(fnode_num, fnode_size);

                              /*
                               * Do partial seek
                               */

 119   4                      DO WHILE dpcmp(fnode_off, dtmp := dssmul(
                                  fnode_ptr(fnode_ptr_index).num_blocks, dev_gran*vol_blks)) > 0;
 120   5                          fnode_off = dpsub(fnode_off, dtmp);
 121   5                          fnode_ptr_index = fnode_ptr_index + 1;
```

```
122   5                 END;
123   4                 short_blks = sdsdiv(fnode_off, dev_gran*vol_blks);
124   4                 short_block_o.lo = fnode_ptr(fnode_ptr_index).block_lo;
125   4                 short_block_o.hi = fnode_ptr(fnode_ptr_index).block_hi;
126   4                 short_block = dsmul(short_block, vol_blks);
127   4                 short_block = dpadd(short_block,dssmul(short_blks, vol_blks));
128   4                 fnode_off = dpsub(fnode_off, dssmul(short_blks, vol_blks*dev_gran));

                        /*
                         *  Read blocks to finish "seek"
                         */

129   4                 DO WHILE dscmp(fnode_off, dev_gran) > 0;
130   5                     CALL read$file(@data_block);
131   5                     fnode_off = dssub(fnode_off, dev_gran);
132   5                 END;

                        /*
                         *  Read one or two blocks of fnode file around desired fnode
                         *  and copy it from 'data_block' to 'fnode'
                         */

133   4                 CALL read$file(@data_block);
134   4                 IF dev_gran-fnode_off_o.lo < size(fnode) THEN
135   4                     CALL read$file(@data_block(dev_gran));
136   4                     CALL MOVB(@data_block(fnode_off_o.lo), @fnode, size(fnode));
137   4                 END;

                        /*
                         *  Initialize variables
                         */

138   3                 fnode_ptr_index = 0;
139   3                 short_block_o.lo = fnode_ptr(0).block_lo;
140   3                 short_block_o.hi = fnode_ptr(0).block_hi;
141   3                 short_block = dsmul(short_block, vol_blks);
142   3                 short_blks = 0;
143   3                 loop_type = 0;
144   3                 long_blks = 0;
145   3                 vol_block = 1;

146   3             END setup$file;

147   2         END bnumfd;
*** ERROR 42 IN 148 (LINE 321), NEAR 'SETUPFILE': INSERTED: " END ;"
```

```
 DEFN  ADDR   SIZE  NAME, ATTRIBUTES, AND REFERENCES
-----  ------ -----  ------------------------------

      0301H        <ID> . . . . . .    PROCEDURE STACK=0000H
    3 0000H     4  <ID> . . . . . .    DWORD
    5                ACCESSORSIZE . .  LITERALLY '9'          51
   51 0004H     2  ACTUAL . . . . .    WORD       105*  106*  107
   46 0000H     4  BLK_NUM. . . . .    DWORD IN PROC (LONG_CALL) PARAMETER      46
    2 0000H        BNUMFD . . . . .    LABEL
   54 FFFEH     2  BUFFER_P . . . .    POINTER IN PROC (READFILE) AUTOMATIC     60*   61*   64
   46 0000H     2  BUF_P. . . . . .    POINTER IN PROC (LONG_CALL) PARAMETER       46
   14                BYTE_PTR_STRUCT. .  LITERALLY 'STRUCTURE(bytBYTE,nextBYTE)'
   12                CR . . . . . .    LITERALLY '0DH'
   51 0000H     1  DATA_BLOCK . . .    BYTE BASED(DATA_BLOCK_P) ARRAY(1)      130  133  135  136
   51 007EH     2  DATA_BLOCK_P . .    POINTER          130  133  135  136
   53 0004H     2  DATA_BUFFER_P. .    POINTER IN PROC (READFILE) PARAMETER AUTOMATIC      53   60   73
   51 0008H     2  DEVICE_DRIVER. .    POINTER           64   73  114
   51 000AH     2  DEV_GRAN . . . .    WORD       85  104  105  119  123  128  129  131  134  135
   46 0000H     2  DEV_PROC . . . .    POINTER IN PROC (LONG_CALL) PARAMETER       46
   22 0000H     4  DP1. . . . . .    DWORD IN PROC (DPADD) PARAMETER          22
   16 0000H     4  DP1. . . . . .    DWORD IN PROC (DPCMP) PARAMETER          16
   22 0000H     4  DP2. . . . . .    DWORD IN PROC (DPADD) PARAMETER          22
   16 0000H     4  DP2. . . . . .    DWORD IN PROC (DPCMP) PARAMETER          16
   21 0000H        DPADD. . . . . .    PROCEDURE DWORD EXTERNAL(2) STACK=0000H      127
   15 0000H        DPCMP. . . . . .    PROCEDURE INTEGER EXTERNAL(0) STACK=0000H    119
   30 0000H        DPSUB. . . . . .    PROCEDURE DWORD EXTERNAL(5) STACK=0000H      120  128
   24 0000H        DSADD. . . . . .    PROCEDURE DWORD EXTERNAL(3) STACK=0000H       65   74
   18 0000H        DSCMP. . . . . .    PROCEDURE INTEGER EXTERNAL(1) STACK=0000H     62  104  129
   39 0000H        DSDIV. . . . . .    PROCEDURE DWORD EXTERNAL(8) STACK=0000H
   36 0000H        DSMUL. . . . . .    PROCEDURE DWORD EXTERNAL(7) STACK=0000H       72   84  102  126
                                        141
   33 0000H        DSSMUL . . . . . .  PROCEDURE DWORD EXTERNAL(6) STACK=0000H      118  119  127  128

   27 0000H        DSSUB. . . . . .    PROCEDURE DWORD EXTERNAL(4) STACK=0000H      107  131
  111 FFF8H     4  DTMP . . . . . .    DWORD IN PROC (SETUPFILE) AUTOMATIC     119*  120
   40 0000H     4  DWA. . . . . .    DWORD IN PROC (DSDIV) PARAMETER          40
   28 0000H     4  DWA. . . . . .    DWORD IN PROC (DSSUB) PARAMETER          28
   25 0000H     4  DWA. . . . . .    DWORD IN PROC (DSADD) PARAMETER          25
   37 0000H     4  DWA. . . . . .    DWORD IN PROC (DSMUL) PARAMETER          37
   31 0000H     4  DWA. . . . . .    DWORD IN PROC (DPSUB) PARAMETER          31
   43 0000H     4  DWA. . . . . .    DWORD IN PROC (SDSDIV) PARAMETER         43
   19 0000H     4  DWA. . . . . .    DWORD IN PROC (DSCMP) PARAMETER          19
   31 0000H     4  DWB. . . . . .    DWORD IN PROC (DPSUB) PARAMETER          31
   50                DWORD_OVERLAY. . .  LITERALLY 'STRUCTURE(loWORD,hiWORD)'      51  111
   48 0000H        ERROR_HALT . . . .  PROCEDURE EXTERNAL(11) STACK=0000H       56
    3                FALSE. . . . . .  LITERALLY '000H'
   11                FNLONG . . . . . .  LITERALLY '00002H'          59   66
   51 0028H    86  FNODE. . . . . .    STRUCTURE        114  134  136
      0000H     2      FLAGS. . . . .  WORD        59   66
      0002H     1      TYPE . . . . .  BYTE
      0003H     1      GRAN . . . . .  BYTE
      0004H     2      OWNER. . . . .  WORD
      0006H     4      CRTIME . . . .  DWORD
      000AH     4      ACCESSTIME . .  DWORD
      000EH     4      MODTIME. . . .  DWORD
```

```
        0012H     4       TOTALSIZE. . .    DWORD       51   55  104  107*  107
        0016H     4       TOTALBLKS. . .    DWORD
        001AH    40       PTR. . . . . .    BYTE ARRAY(40)          51
        0042H     4       THISSIZE . . .    DWORD
        0046H     2       EXTFNODE . . .    WORD
        0048H     2       PREVFNODE. . .    WORD
        004AH     2       IDCOUNT. . . .    WORD
        004CH     9       ACCESSOR . . .    BYTE ARRAY(9)
        0055H     1       AUX. . . . . .    BYTE ARRAY(1)
     6                  FNODECOMMON1 . . .  LITERALLY 'flagsWORD,typeBYTE,granBYTE,ownerWORD,cr$tim
                         -eDWORD,access$timeDWORD,mod$timeDWORD,total$sizeDWORD,to
                         -tal$blksDWORD,ptr(FNODE$PTRS$SIZE)BYTE'        51
     7                  FNODECOMMON2 . . .  LITERALLY 'this$sizeDWORD,ext$fnodeWORD,prev$fnodeWORD,id$count
                         -WORD,accessor(ACCESSOR$SIZE)BYTE'      51
     8                  FNODECOMMONINFO. .  LITERALLY 'FNODE$COMMON$1,FNODE$COMMON$2'       51
     4                  FNODEPTRSSIZE. . .  LITERALLY '40'          51
    51  0010H     4   FNODE_FILE_BLOCK .   DWORD      114
   110  0004H     2   FNODE_NUM. . . . .   WORD IN PROC (SETUPFILE) PARAMETER AUTOMATIC         110  112  118
   111  FFFCH     4   FNODE_OFF. . . . .   DWORD IN PROC (SETUPFILE) AUTOMATIC      111  118*  119  120*  120
                         123  128*  128  129  131*  131
   111  FFFCH     4   FNODE_OFF_0. . . .   STRUCTURE IN PROC (SETUPFILE) AT AUTOMATIC
        0000H     2       LO . . . . . .    WORD AUTOMATIC          134  136
        0002H     2       HI . . . . . .    WORD AUTOMATIC
    51  0042H    40   FNODE_PTR. . . . .   STRUCTURE ARRAY(8) AT
        0000H     2       NUM_BLOCKS . .    WORD        95  119
        0002H     2       BLOCK_LO . . .    WORD       100  124  139
        0004H     1       BLOCK_HI . . .    BYTE       101  125  140
    51  0014H     2   FNODE_PTR_INDEX. .   WORD        95   97*   97  100  101  119  121*  121  124  125
                         138*
    51  0006H     2   FNODE_SIZE . . . .   WORD       118
    51  003AH     4   FNODE_TOTAL_SIZE_0   STRUCTURE AT
        0000H     2       LO . . . . . .    WORD       106
        0002H     2       HI . . . . . .    WORD
     3                  FOREVER. . . . . .  LITERALLY 'WHILE TRUE'
    51  0080H     1   INDIR_BLOCK. . . .   BYTE ARRAY(1)           51   61
    51  0080H     4   INDIR_PTR. . . . .   STRUCTURE ARRAY(1) AT
        0000H     1       NUM_BLOCKS . .    BYTE        78
        0001H     2       BLOCK_LO . . .    WORD        70   82
        0003H     1       BLOCK_HI . . .    BYTE        71   83
    51  001CH     2   INDIR_PTR_INDEX. .   WORD        69*   78   80*   80   82   83   85
    73  0000H         INNERLOOP. . . . .   LABEL IN PROC (READFILE)      58
    13                LEDATA . . . . . .   LITERALLY '0A0H'
    12                LF . . . . . . . .   LITERALLY '0AH'
    13                LIBHED . . . . . .   LITERALLY '0A4H'
    51  0022H     2   LONG_BLKS. . . . .   WORD        77*   77   78   81*  144*
    51  001EH     4   LONG_BLOCK . . . .   DWORD       51   72*   72   73   74*   74   84*   84
    51  001EH     4   LONG_BLOCK_0 . . .   STRUCTURE AT
        0000H     2       LO . . . . . .    WORD        70*   82*
        0002H     2       HI . . . . . .    WORD        71*   83*
    45  0000H         LONG_CALL. . . . .   PROCEDURE EXTERNAL(10) STACK=0000H        64   73  114
    51  0024H     2   LOOP_TYPE. . . . .   WORD        57   68*   86*   99*  143*
     5                MAXACCESSORS . . .   LITERALLY '3'
    13                MAX_PIDATA . . . .   LITERALLY '1024'
    13                MODEND . . . . . .   LITERALLY '08AH'
                      MOVB . . . . . . .   BUILTIN          136
    10                NAMEDDISKFNODE . .   LITERALLY 'STRUCTURE(FNODE$COMMON$INFO,parentWORD,aux(1)BYTE)'
```

```
 89  0000H        NOLONG . . . . . LABEL IN PROC (READFILE)      67
  9               NUMDISKFNODE . . . LITERALLY 'STRUCTURE(FNODESCOMMONSINFO,aux(1)BYTE)'       51
 13               PEDATA . . . . . LITERALLY '084H'
 13               PIDATA . . . . . LITERALLY '086H'
  3               PTROVERLAY . . . . LITERALLY 'STRUCTURE(offset WORD, base WORD)'
 52  0000H        READFILE . . . . . PROCEDURE PUBLIC REENTRANT STACK=0000H         130  133  135
 42  0000H        SDSDIV . . . . . PROCEDURE WORD EXTERNAL(9) STACK=0000H          123
109  0000H        SETUPFILE. . . . . PROCEDURE PUBLIC REENTRANT STACK=0000H        117
                  SHL. . . . . . . BUILTIN          35
 51  001AH    2   SHORT_BLKS . . . . WORD        91*   91   95   98*  123*  127  128  142*
 51  0016H    4   SHORT_BLOCK. . . . DWORD       51   62   64   65*   65  102*  102  126*  126  127*
                                      127  141*  141
 51  0016H    4   SHORT_BLOCK_0. . . STRUCTURE AT
     0000H    2       LO . . . . . WORD       100*  124*  139*
     0002H    2       HI . . . . . WORD       101*  125*  140*
                  SIZE . . . . . . BUILTIN          134  136
 13               ST_ADDR_PRESENT. . LITERALLY '40H'
 34  0000H    2   SWA. . . . . . . WORD IN PROC (DSSMUL) PARAMETER          34
 43  0000H    2   SWB. . . . . . . WORD IN PROC (SDSDIV) PARAMETER          43
 40  0000H    2   SWB. . . . . . . WORD IN PROC (DSDIV) PARAMETER      40
 37  0000H    2   SWB. . . . . . . WORD IN PROC (DSMUL) PARAMETER      37
 34  0000H    2   SWB. . . . . . . WORD IN PROC (DSSMUL) PARAMETER          34
 28  0000H    2   SWB. . . . . . . WORD IN PROC (DSSUB) PARAMETER      28
 25  0000H    2   SWB. . . . . . . WORD IN PROC (DSADD) PARAMETER      25
 19  0000H    2   SWB. . . . . . . WORD IN PROC (DSCMP) PARAMETER      19
  3               TRUE . . . . . . LITERALLY 'OFFH'
 46  0000H    2   UNIT . . . . . . WORD IN PROC (LONG_CALL) PARAMETER        46
 51  000CH    2   UNIT . . . . . . WORD        64   73  114
 51  000EH    2   VOL_BLKS . . . . . WORD       72   75   84   89  102  119  123  126  127  128
                                      141
 51  0026H    2   VOL_BLOCK. . . . . WORD       75   89   92*   94*   94  145*
```

MODULE INFORMATION:

```
    CODE AREA SIZE     = 0000H      0D
    CONSTANT AREA SIZE = 0000H      0D
    VARIABLE AREA SIZE = 0081H    129D
    MAXIMUM STACK SIZE = 0000H      0D
    507 LINES READ
    0 PROGRAM WARNINGS
    29 PROGRAM ERRORS

END OF PL/M-86 COMPILATION
```