BOO
V03-

```
UUU         UUU  VVV         VVV        111        RRRRRRRRRRRR        000000000     MMM         MMM
UUU         UUU  VVV         VVV        111        RRRRRRRRRRRR        000000000     MMM         MMM
UUU         UUU  VVV         VVV        111        RRRRRRRRRRRR        000000000     MMM         MMM
UUU         UUU  VVV         VVV     111111        RRR         RRR  000         000  MMMMMM   MMMMMM
UUU         UUU  VVV         VVV     111111        RRR         RRR  000         000  MMMMMM   MMMMMM
UUU         UUU  VVV         VVV     111111        RRR         RRR  000         000  MMMMMM   MMMMMM
UUU         UUU  VVV         VVV        111        RRR         RRR  000         000  MMM   MMM   MMM
UUU         UUU  VVV         VVV        111        RRR         RRR  000         000  MMM   MMM   MMM
UUU         UUU  VVV         VVV        111        RRR         RRR  000         000  MMM   MMM   MMM
UUU         UUU  VVV         VVV        111        RRRRRRRRRRRR     000         000  MMM         MMM
UUU         UUU  VVV         VVV        111        RRRRRRRRRRRR     000         000  MMM         MMM
UUU         UUU  VVV         VVV        111        RRRRRRRRRRRR     000         000  MMM         MMM
UUU         UUU  VVV         VVV        111        RRR   RRR        000         000  MMM         MMM
UUU         UUU  VVV         VVV        111        RRR   RRR        000         000  MMM         MMM
UUU         UUU  VVV         VVV        111        RRR   RRR        000         000  MMM         MMM
UUU         UUU     VVV   VVV           111        RRR      RRR     000         000  MMM         MMM
UUU         UUU     VVV   VVV           111        RRR      RRR     000         000  MMM         MMM
UUU         UUU     VVV   VVV           111        RRR      RRR     000         000  MMM         MMM
UUUUUUUUUUUUUUU        VVV           111111111     RRR         RRR     000000000     MMM         MMM
UUUUUUUUUUUUUUU        VVV           111111111     RRR         RRR     000000000     MMM         MMM
UUUUUUUUUUUUUUU        VVV           111111111     RRR         RRR     000000000     MMM         MMM
```

**FILE**ID**NETBOOT                                M 10

```
NN      NN  EEEEEEEEEE  TTTTTTTTTT  BBBBBBBB       000000       000000     TTTTTTTTTT
NN      NN  EEEEEEEEEE  TTTTTTTTTT  BBBBBBBB       000000       000000     TTTTTTTTTT
NN      NN  EE              TT      BB      BB  00      00  00      00         TT
NN      NN  EE              TT      BB      BB  00      00  00      00         TT
NNNN    NN  EE              TT      BB      BB  00      00  00      00         TT
NNNN    NN  EE              TT      BB      BB  00      00  00      00         TT
NN  NN  NN  EEEEEEEE        TT      BBBBBBBB    00      00  00      00         TT
NN  NN  NN  EEEEEEEE        TT      BBBBBBBB    00      00  00      00         TT
NN    NNNN  EE              TT      BB      BB  00      00  00      00         TT
NN    NNNN  EE              TT      BB      BB  00      00  00      00         TT
NN      NN  EE              TT      BB      BB  00      00  00      00         TT          ....
NN      NN  EE              TT      BB      BB  00      00  00      00         TT          ....
NN      NN  EEEEEEEEEE      TT      BBBBBBBB       000000       000000         TT          ....
NN      NN  EEEEEEEEEE      TT      BBBBBBBB       000000       000000         TT          ....

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

NE
ELI

```
module netboot;

{****************************************************************************
{*                                                                          *
{*  Copyright (c) 1984                                                      *
{*  by DIGITAL Equipment Corporation, Maynard, Mass.                        *
{*                                                                          *
{*  This software is furnished under a license and may be used and  copied  *
{*  only  in  accordance  with  the  terms  of  such  license and with the  *
{*  inclusion of the above copyright notice.  This software or  any  other  *
{*  copies  thereof may not be provided or otherwise made available to any  *
{*  other person.  No title to and ownership of  the  software  is  hereby  *
{*  transferred.                                                            *
{*                                                                          *
{*  The information in this software is subject to change  without  notice  *
{*  and  should  not  be  construed  as  a commitment by DIGITAL Equipment  *
{*  Corporation.                                                            *
{*                                                                          *
{*  DIGITAL assumes no responsibility for the use or  reliability  of  its  *
{*  software on equipment which is not supplied by DIGITAL.                 *
{*                                                                          *
{****************************************************************************
{

{++
* FACILITY:
*
*       VAXELN and MicroVAX I
*
* ABSTRACT:
*
*       This module contains a network primary bootstrap.  It handles the
*       DNA Low-level Maintenance Protocol (MOP) messages to down-line
*       load a VAX processor with a system memory image.
*
* AUTHOR:
*
*       Len Kawell, 19-Nov-1981
*
*       V1.0-01 Len Kawell      23-May-1984
*               Call a new driver entry point to stop the device when the
*               load is complete.  This makes sure that the device does not
*               continue receiving messages once the O/S is loaded.
--}
```

```
        include
                $datalink;

        const
                memload_code            = 2;            { Memory load without xfer addr }
                request_program_code    = 8;            { Request program }
                request_load_code       = 10;           { Request memory load }
                parameters_code         = 20;           { Paramters with xfer addr }

                opsys_pgmtype           = 2;            { Operating system program type }

                transfer_parameter      = 0;            { Transfer address parameter }
                node_name_parameter     = 1;            { Node name parameter }
                node_address_parameter  = 2;            { Node address parameter }
                host_name_parameter     = 3;            { Host name parameter }
                host_address_parameter  = 4;            { Host address parameter }

                maximum_retries         = 8;            { Maximum retries }

                ss$_normal              = 1;            { Success }
                ss$_bufferovf           = %x600;        { Buffer overflow error }
                ss$_ctrlerr             = %x54;         { Device controller error }
                ss$_devinact            = %x20d4;       { Device initialization error }
                ss$_nosuchnode          = %x28c;        { No respose from load server }

        type
                byte = 0..255;                          { Byte }
                word = 0..65535;                        { Word }
                buffer = array[0..255] of char;         { I/O buffer }

                xmtmsg = packed record                  { Transmit message format}
                        code: byte;                     {  Code }
                            case byte of
                                request_program_code: (
                                        devtype: byte;  {  Device type }
                                        mopver: byte;   {  MOP version }
                                        pgmtype: byte;  {  Program type }
                                        );
                                request_load_code: (
                                        loadnum: byte;  {  Load number }
                                        error: byte;    {  Error flag }
                                        );
                        end;

                rcvmsg = packed record                  { Receive message format}
                        code: byte;                     {  Code }
                        loadnum: byte;                  {  Load number }
                            case byte of

                                memload_code: (         {  Memory load }
                                        loadaddr: ^buffer; {  Load address }
                                        image: buffer;  {  Image data }
                                        );

                                parameters_code: (      {  Parameters }
                                        param: packed array[1..255] of char;
                                        );
```

```
                          end;

                  bitmap = packed array[0..8192] of boolean; { PFN bitmap }

         function net_init(var devtype: integer; data_buffer: ^anytype):boolean;
         external;

         function net_transmit(var buff: byte_data(buff_size); buff_size: word): boolean;
         external;

         function net_receive(var buff: byte_data(buff_size); buff_size: word): boolean;
         external;

         procedure net_stop;
         external;

```

```
0        function boo$downline_load(load_address: integer;
1                                   data_buffer: ^anytype;
1                                   pfn_bitmap: ^bitmap;
1                                    var transfer_address: integer;
1                                    var node_address: datalink_address;
1                                    var node_name: varying_string(6)): integer;
1        {++
1        { boo$downline_load - Downline load a system image
1        {
1        { Inputs:
1        {
1        {       load_address - Starting physical load address
1        {       data_buffer - Pointer to an 8KB buffer for use by the load driver
1        {       pfn_bitmap - Pointer to the Page Frame Number bitmap
1        {
1        { Outputs:
1        {
1        {       transfer_address - Received transfer address
1        {       node_address - Received node address
1        {       node_name - Received node name
1        {
1        {       Function value -
1        {                       ss$_normal - Downline load completed successfully
1        {
1        {                       ss$_bufferovf - Load buffer overflow (ie. PFN invalid)
1        {                       ss$_devinact - Device initialization error
1        {                       ss$_ctrlerr - Device I/O error
1        {                       ss$_nosuchnode - No response from a load server
1        {--}
1        var
1                status: boolean;
1                retries: integer := maximum_retries;
1                request_opsys: boolean := true;
1                current_loadnum: integer := 0;
1                current_devtype: integer;
1                xmtbuff: xmtmsg;
1                rcvbuff: rcvmsg;
1                pi: integer;
1        begin
                                    0008
                                    0008  BOO$DOWNLINE_LOAD:
                          CFFC      0008          .entry  BOO$DOWNLINE_LOAD,^m<dv,iv,r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
                 5E FEE8 CE 9E      000A          movab   -0118(sp),sp
                    6E 04 AC D0     000F          movl    04(ap),(sp)
                    5B 0C AC D0     0013          movl    0C(ap),r11
                       57 08 D0     0017          movl    #8,r7
                       5A 01 90     001A          movb    #1,r10
                          58 D4     001D          clrl    r8

1
1        rcvbuff.code := 0;
                                    001F
                    FEFA CD 94      001F          clrb    -0106(fp)

1
1        { Initialize the driver }
1
```

```
  164  1        if not net_init(current_devtype, data_buffer)
                                      0023
                         08 AC DD     0023          pushl   08(ap)
                       FEF0 CD DF     0026          pushal  -0110(fp)
              00000000* EF 02 FB      002A          calls   #2,NET_INIT
                         08 50 E8     0031          blbs    r0,vcg.1

  165  1        then
  166  1               begin
                                      0034

  167  2               boo$downline_load := ss$_devinact;
                                      0034
                   59 20D4 8F 3C      0034          movzwl  #20D4,r9

  168  2               goto return;
                                      0039
                          016C 31     0039          brw     RETURN

  169  2               end;
                                      003C
                                      003C  vcg.1:

  170  1
  171  1        repeat
                                      003C

  172  2
  173  2               { Create a Request Program or Request Load message }
  174  2
  175  2               if request_opsys
                                      003C
                         18 5A E9     003C          blbc    r10,vcg.2

  176  2               then
  177  2                      begin
                                      003F

  178  3                      xmtbuff.code := request_program_code;
                                      003F
                   FEF6 CD 08 90      003F          movb    #8,-010A(fp)

  179  3                      xmtbuff.devtype := current_devtype;
                                      0044
              FEF7 CD FEF0 CD 90      0044          movb    -0110(fp),-0109(fp)

  180  3                      xmtbuff.mopver := 1;
                                      004B
                   FEF8 CD 01 90      004B          movb    #1,-0108(fp)

  181  3                      xmtbuff.pgmtype := opsys_pgmtype;
                                      0050
                   FEF9 CD 02 90      0050          movb    #2,-0107(fp)

  182  3                      end
                                      0055
                            0E 11     0055          brb     vcg.3
```

```
                                     0057  vcg.2:

 183  2             else
 184  2                   begin
                                     0057

 185  3                   xmtbuff.code := request_load_code;
                                     0057
                    FEF6 CD 0A 90    0057           movb    #A,-010A(fp)

 186  3                   xmtbuff.loadnum := current_loadnum;
                                     005C
                    FEF7 CD 58 90    005C           movb    r8,-0109(fp)

 187  3                   xmtbuff.error := 0;
                                     0061
                      FEF8 CD 94     0061           clrb    -0108(fp)

 188  3                   end;
                                     0065
                                     0065  vcg.3:

 189  2
 190  2             { Transmit the message and retry if it fails. }
 191  2
 192  2             while not net_transmit(xmtbuff, size(xmtbuff)) do
                                     0065
                           04 DD     0065           pushl   #4
                      FEF6 CD 9F     0067           pushab  -010A(fp)
                00000000* EF 02 FB   006B           calls   #2,NET_TRANSMIT
                        0B 50 E8     0072           blbs    r0,vcg.4

 193  2                   begin
                                     0075

 194  3                   retries := retries - 1;
                                     0075
                           57 D7     0075           decl    r7

 195  3                   if retries <= 0
                                     0077
                           EC 14     0077           bgtr    vcg.3

 196  3                   then
 197  3                         begin
                                     0079

 198  4                         boo$downline_load := ss$_ctrlerr;
                                     0079
                     59 54 8F 9A     0079           movzbl  #54,r9                                                          58

 199  4                         goto return;
                                     007D
                         0128 31     007D           brw     RETURN                                                          58

 200  4                         end;
```

```
                                     0080

 201  3                    end;
                                     0080
                                     0080  vcg.4:

 202  2
 203  2              { If last message received was a Parameters message, then load done }
 204  2
 205  2              if rcvbuff.code = parameters_code
                                     0080
               52 FEFA CD 9A         0080           movzbl  -0106(fp),r2
                   14 52 D1          0085           cmpl    r2,#14
                      03 12          0088           bneq    gen.1
                    0087 31          008A           brw     DONE
                                     008D  gen.1:

 206  2              then
 207  2                    goto done;
                                     008D

 208  2
 209  2              { Receive a response message }
 210  2
 211  2              status := net_receive(rcvbuff, size(rcvbuff));
                                     008D
               7E 0106 8F 3C         008D           movzwl  #106,-(sp)
                    FEFA CD 9F       0092           pushab  -0106(fp)
           00000000* EF 02 FB        0096           calls   #2,NET_RECEIVE

 212  2
 213  2              { If a message successfully received and it is either a Memory Load }
 214  2              { or a Parameters message, process it. }
 215  2
 216  2              if status and
                                     009D
                   5B 50 E9          009D           blbc    r0,vcg.9
               52 FEFA CD 9A         00A0           movzbl  -0106(fp),r2
                   02 52 D1          00A5           cmpl    r2,#2
                      05 13          00A8           beql    vcg.5
                   14 52 D1          00AA           cmpl    r2,#14
                      4C 12          00AD           bneq    vcg.9
                                     00AF  vcg.5:

 217  2                ((rcvbuff.code = memload_code) or
 218  2                 (rcvbuff.code = parameters_code))
 219  2              then
 220  2                    begin
                                     00AF

 221  3                    if (rcvbuff.loadnum = current_loadnum)
                                     00AF
               52 FEFB CD 9A         00AF           movzbl  -0105(fp),r2
                   58 52 D1          00B4           cmpl    r2,r8
                      04 13          00B7           beql    vcg.6
                      52 D5          00B9           tstl    r2
                      39 12          00BB           bneq    vcg.8
```

```
                                                   00BD  vcg.6:

 222  3                                  or (rcvbuff.loadnum = 0)
 223  3                                  then
 224  3                                          begin
                                                   00BD

 225  4                                          current_loadnum := (rcvbuff.loadnum + 1) mod 256;
                                                   00BD
                             52 FEFB CD 9A         00BD                 movzbl  -0105(fp),r2
                                   52 D6           00C2                 incl    r2
                                58 52 9A           00C4                 movzbl  r2,r8

 226  4
 227  4                                          { If memory data, then move to proper memory location }
 228  4
 229  4                                          if rcvbuff.code = memload_code
                                                   00C7
                             52 FEFA CD 9A         00C7                 movzbl  -0106(fp),r2
                                02 52 D1           00CC                 cmpl    r2,#2
                                   25 12           00CF                 bneq    vcg.8

 230  4                                          then
 231  4                                                  begin
                                                   00D1

 232  5
 233  5                                                  { Compute memory location for data and check }
 234  5                                                  { that page is valid in page frame bitmap. }
 235  5
 236  5                                                  rcvbuff.loadaddr::integer :=
                                                   00D1
                             FEFC CD 6E C0         00D1                 addl2   (sp),-0104(fp)

 237  5                                                        rcvbuff.loadaddr::integer + load_address;
 238  5
 239  5                                                  if not pfn_bitmap^
                                                   00D6
                52 FEFC CD 00000200 8F C7          00D6                 divl3   #200,-0104(fp),r2
                               08 6B 52 E0         00E0                 bbs     r2,(r11),vcg.7

 240  5                                                         [rcvbuff.loadaddr::integer div 512]
 241  5                                                  then
 242  5                                                         begin
                                                   00E4

 243  6                                                         boo$downline_load := ss$_bufferovf;
                                                   00E4
                             59 0600 8F 3C         00E4                 movzwl  #600,r9

 244  6                                                         goto return;
                                                   00E9
                                  00BC 31          00E9                 brw     RETURN

 245  6                                                         end;
                                                   00EC
                                                   00EC  vcg.7:
```

PQE
V03

```
  246  5
  247  5                                               rcvbuff.loadaddr^ := rcvbuff.image;
                                                 00EC
        FEFC DD FF00 CD 0100 8F 28               00EC          movc3   #100,-0100(fp),@-0104(fp)

  248  5                                               end;
                                                 00F6
                                                 00F6  vcg.8:

  249  4                                         end;
                                                 00F6

  250  3                                 request_opsys := false;
                                                 00F6
                                   5A 94         00F6          clrb    r10

  251  3                                 end
                                                 00F8
                                 FF41 31         00F8          brw     vcg.1
                                                 00FB  vcg.9:

  252  2                       else
  253  2                                 begin
                                                 00FB

  254  3
  255  3                                 { Decrement the retry count.  If it is zero quit. }
  256  3
  257  3                                 retries := retries - 1;
                                                 00FB
                                   57 D7         00FB          decl    r7

  258  3                                 if retries <= 0
                                                 00FD
                                   03 15         00FD          bleq    gen.2
                                 FF3A 31         00FF          brw     vcg.1
                                                 0102  gen.2:

  259  3                                 then
  260  3                                         begin
                                                 0102

  261  4
  262  4                                         { Return device error status.  If no reponse was ever }
  263  4                                         { received from a load server, return no response }
  264  4                                         { from server status. }
  265  4
  266  4                                         boo$downline_load := ss$_ctrlerr;
                                                 0102
                             59 54 8F 9A         0102          movzbl  #54,r9

  267  4                                         if request_opsys
                                                 0106
                                03 5A E8         0106          blbs    r10,gen.3
                                 009C 31         0109          brw     RETURN
                                                 010C  gen.3:

  268  4                                         then
```

```
 269  4                                                 boo$downline_load := ss$_nosuchnode;
                                           010C
                    59 028C 8F 3C          010C                movzwl  #28C,r9

 270  4                                         goto return;
                                           0111
                              0094 31      0111                brw     RETURN

 271  4                                         end;
                                           0114

 272  3                                 end
                                           0114

 273  2         until false;
                                           0114

 274  1
 275  1         done:
 276  1
                                           0114
                                           0114  DONE:

 277: 1         { Get the load parameters }
 278  1
 279  1         pi := 1;
                                           0114
                              56 01 D0     0114                movl    #1,r6

 280  1         transfer_address := -1;
                                           0117
                          5A 10 BC DE      0117                moval   @10(ap),r10
                             6A 01 CE      011B                mnegl   #1,(r10)

 281  1         node_name := '';
                                           011E
                          58 18 BC DE      011E                moval   @18(ap),r8
                                68 B4      0122                clrw    (r8)

 282  1
 283  1         with rcvbuff do
                                           0124

 284  1                 repeat
                                           0124
                          57 14 BC DE      0124                moval   @14(ap),r7
                                           0128  vcg.10:

 285  2
 286  2                 { Scan parameter list for parameters. }
 287  2
 288  2                 case ord(param[pi]) of
                                           0128
                  52 FEFB CD46 9A          0128                movzbl  -0105(fp)[r6],r2
                          02 00 52 CF      012E                casel   r2,#0,#2
                                           0132  case.288:
                                           0132
```

```
                                  0138  case.288.ow:
                          5A 11   0138           brb     vcg.14

 289  3
 290  3            node_name_parameter:
                                  013A
                                  013A  a case label:

 291  3                 node_name := substr(param, pi + 2, ord(param[pi + 1]));
                                  013A
                52 FEFC CD46 9A   013A           movzbl  -0104(fp)[r6],r2
                       68 52 F7   0140           cvtlw   r2,(r8)
                       68 06 B1   0143           cmpw    #6,(r8)
                          03 1E   0146           bgequ   vcg.11
                       68 06 B0   0148           movw    #6,(r8)
                                  014B  vcg.11:
       02 A8 FEFD CD46 68 28      014B           movc3   (r8),-0103(fp)[r6],02(r8)
                          3F 11   0153           brb     vcg.14

 292  3
 293  3            node_address_parameter:
                                  0155
                                  0155  a case label:

 294  3                 begin
                                  0155

 295  4
 296  4                 { If only a 2 byte node address sent, pad it using the }
 297  4                 { DECnet Ethernet high-order address. }
 298  4
 299  4                 if ord(param[pi + 1]) > 2
                                  0155
                52 FEFC CD46 9A   0155           movzbl  -0104(fp)[r6],r2
                       02 52 D1   015B           cmpl    r2,#2
                          09 15   015E           bleq    vcg.12

 300  4                 then
 301  4                         node_address :=
                                  0160
          67 FEFD CD46 06 28      0160           movc3   #6,-0103(fp)[r6],(r7)
                          10 11   0167           brb     vcg.13
                                  0169  vcg.12:

 302  5                                 substr(param, pi + 2, size(node_address))
 303  5                 else
 304  4                         begin
                                  0169

 305  5                         node_address    ''(170,00,04,00,00,00,00,00);
                                  0169
             67 FE92 CF 06        0169           movc3   #6,$CODE,(r7)

 306  5                         substr(node_address, 5, 2) := substr(param, pi + 2, 2);
                                  016F
                52 FEFD CD46 9E   016F           movab   -0103(fp)[r6],r2
                    04 A7 62 B0   0175           movw    (r2),04(r7)
 307  5                             end;
                                      0179
                                      0179  vcg.13:

 308  4
 309  4                     { If the node address does not have an area address, set }
 310  4                     { it to area 1. }
 311  4
 312  4                     if ord(substr(node_address, 6, 1)) < 4
                                      0179
                  52 05 A7 9A         0179          movzbl  05(r7),r2
                     04 52 D1         017D          cmpl    r2,#4
                        12 18         0180          bgeq    vcg.14

 313  4                     then
 314  4                             substr(node_address, 6, 1) :=
                                      0182
                     52 04 C0         0182          addl2   #4,r2
                  05 A7 52 90         0185          movb    r2,05(r7)

 315  5                                     chr(ord(substr(node_address, 6, 1)) + 4);
 316  4                     end;
                                      0189
                        09 11         0189          brb     vcg.14

 317  3
 318  3             transfer_parameter:
                                      018B
                                      018B  a case label:

 319  3
 320  3                     transfer_address := substr(param, pi + 1, 4)::integer;
                                      018B
              52 FEFC CD46 9E         018B          movab   -0104(fp)[r6],r2
                     6A 62 D0         0191          movl    (r2),(r10)
                                      0194  vcg.14:

 321  3             end;
 322  2
 323  2             pi := pi + ord(param[pi + 1]) + 2;
                                      0194
              52 FEFC CD46 9A         0194          movzbl  -0104(fp)[r6],r2
                     52 56 C0         019A          addl2   r6,r2
                  56 52 02 C1         019D          addl3   #2,r2,r6

 324  2
 325  2             until transfer_address >= 0;
                                      01A1
                        6A D5         01A1          tstl    (r10)
                        83 19         01A3          blss    vcg.10

 326  1
 327  1     boo$downline_load := ss$_normal;
                                      01A5
                     59 01 D0         01A5          movl    #1,r9

 328  1
 329  1         return:
                                      01A8
                                      01A8  RETURN:

 330  1
 331  1         { Make sure the device is stopped }
 332  1
 333  1         net_stop;
                                      01A8
             00000000* EF 00 FB       01A8          calls   #0,NET_STOP

 334  1
 335  1         end;
                                      01AF
                          50 59 D0    01AF          movl    r9,r0
                                04    01B2          ret
                                      01B3

 336  0
 337  0         end; .

  EPASCAL/MACH/LIS=LIS$:/OBJ=OBJ$: VMB$:NETBOOT+ELN$:RTLOBJECT/LIB
```

```
PQ
Sy

$T
BO
BO
BO
BQ
BQ
BT
CD
CM
CM
DE
DS
ER
ER
FU
GO
IN
IO
IO
IO
MO
MS
MS
MS
MS
MS
MS
MS
MS
MS
MS
MS
MS
MS
MS
MS
MS
MS
MS
MS
OW
PQ
PR
PR
RD
RE
RI
RP
RP
RP
RP
RP
RP
RS
RS
S1
S4
```

N 11

PQ
Ps

PS
---

$A
BO
BO

Ph
---
In
Co
Pa
Sy
Pa
Sy
Ps
Cr
As

Th
86
Th
52
24

Ma
---
DI
DI
SY
TO

17

Th

MA

0430 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

UV1ROM

PQBTDRIVR
LIS

QVSS
LIS

SEARCHMSG
LIS

VMBUVAX1
MAP

SETUSER
LIS

VMBUVAX1
LIS

BOOTDRUV1
LIS

FILERDUV1
LIS

BOOTIOUV1
LIS

CONIO
LIS

NETBOOT
LIS