

ECVMB-1.0

VAX 11750
BOOT BLOCKS

EP-ECVMB-DL-1.0
FICHE 1 OF 1

APR 1981
COPYRIGHT © 1981
MADE IN USA



IDENTIFICATION

PRODUCT CODE: ZZ-ECVMB-1.0
PRODUCT TITLE: BOOT BLOCK LISTING
PRODUCT DATE: APRIL, 1981
DEPARTMENT: VAX ENGINEERING

COPYRIGHT (C) 1981
DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A
SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLU-
SION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY
OTHER COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE MADE
AVAILABLE TO ANY OTHER PERSON EXCEPT FOR USE ON SUCH SYSTEM
AND TO ONE WHO AGREES TO THESE LICENSE TERMS. TITLE TO AND
OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.
THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT
NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL
EQUIPMENT CORPORATION.
DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF
ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.

ZZ-ECVIB-1.0
BOOTBLOCK
Table of contents
(2) 48
(3) 77

Declarations
BOOSBLOCK - reads in and starts boot cod

C 1
9-JAN-1981

Fiche 1 Frame C1
9-JAN-1981 05:45:42 VAX-11 Macro V02.46

Sequence 2

Page 0

```
0000 1 .TITLE BOOTBLOCK
0000 2 .IDENT 'V02-001'
0000 3
0000 4
0000 5 :*****
0000 6 :*
0000 7 :* COPYRIGHT (c) 1979, 1980
0000 8 :* BY DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
0000 9 :*
0000 10 :* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 :* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 :* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 :* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 :* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 :* TRANSFERRED.
0000 16 :*
0000 17 :* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 :* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 :* CORPORATION.
0000 20 :*
0000 21 :* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 :* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23 :*
0000 24 :*****
0000 25 :
0000 26 :++
0000 27 :
0000 28 : FACILITY:
0000 29 :
0000 30 : Device-independent boot block for VAX
0000 31 :
0000 32 : ABSTRACT:
0000 33 :
0000 34 : Reads a file (usually VMB.EXE) off the booting medium into
0000 35 : memory and transfers control to the VMB code.
0000 36 :
0000 37 : AUTHOR:
0000 38 :
0000 39 : Carol Peters 23 August 1979
0000 40 :
0000 41 : REVISION HISTORY:
0000 42 :
0000 43 : Robert Rappaport 13 Sept 1979
0000 44 : Simplified references to local data items.
0000 45 :
0000 46 :--
```

```
Declarations
0000 48      .SBTTL Declarations
0000 49
0000 50 ;
0000 51 ; Own storage.
0000 52 ;
0000 53
0000 54 FILE_STATS:           ; Reserve space to contain
0000 55 FILE_SIZE:           ;
00000000 0000 56          .LONG 0           ; # blocks in primary boot.
00000000 0004 57 START_LBN:           ;
00000000 0004 58          .LONG 0           ; Swapped words of start LBN.
00000000 0008 59 LOAD_ADDR:           ;
00000000 0008 60          .LONG 0           ; Load address for primary boot.
000C 61          ; NOTE - the load address here is
000C 62          ; relative to the base of the 64KB
000C 63          ; of physical memory in which we are
000C 64          ; currently running. As explained
000C 65          ; below, the UNIBUS (or MASSBUS) map
000C 66          ; registers numbered 0-127 will be
000C 67          ; mapped to this same 64KB. Therefore,
000C 68          ; this address, as is, can be used by
000C 69          ; UNIBUS (or MASSBUS) devices to
000C 70          ; pinpoint where to load the primary
000C 71          ; bootstrap program. However, to
000C 72          ; calculate the physical memory address
000C 73          ; corresponding to this relative
000C 74          ; address, we must add in the physical
000C 75          ; offset of the base of the 64KB.
```

000C 77
000C 78
000C 79
000C 80
000C 81
000C 82
000C 83
000C 84
000C 85
000C 86
000C 87
000C 88
000C 89
000C 90
000C 91
000C 92
000C 93
000C 94
000C 95
000C 96
000C 97
000C 98
000C 99
000C 100
000C 101
000C 102
000C 103
000C 104
000C 105
000C 106
000C 107
000C 108
000C 109
000C 110
000C 111
000C 112
000C 113
000C 114
000C 115
000C 116
000C 117
000C 118
000C 119
000C 120
000C 121
000C 122
000C 123
000C 124
000C 125
000C 126
000C 127
000C 128
000C 129
000C 130
000C 131
000C 132
000C 133

++
Functional description:

The boot block code reads the primary bootstrap file into physical memory a block at a time. The code calls the device-dependent ROM subroutine once for each block in the bootstrap file. Then the routine jumps to byte 0 of the loaded code.

Inputs:

- R0 - type of boot device
- R1 - (UNIBUS) address of the I/O page for the boot device's UNIBUS
- R2 - (UNIBUS) address of the device's MASSBUS adapter (MASSBUS) 32-bit physical address of the boot device's CSR (bits <31:24> must be zero)
- R3 - (MASSBUS) adapter's controller/formatter number
- R5 - unit number of the boot device
- R6 - software boot control flags
- R6 - physical address of the device-dependent ROM routine that reads an arbitrary LBN into memory
- SP - <base_address + ^X200> of 64kb of good memory

Implicit inputs:

UNIBUS adapter map registers 0-127 are mapped to the 64kb of good memory. MR 0 maps to first page of memory, etc.

The boot block is loaded into the 1st page of the 64KB of memory, i.e. the page which corresponds to MR 0.

The first longword (bytes 0-3) of the boot block contains the size of the primary bootstrap.

The second longword (bytes 4-7) contains the starting LBN of the bootstrap file, expressed as swapped words.

The third longword (bytes 8-11) contains the relative offset from the base of the 64KB of memory into which we should load the primary bootstrap program. This must be a positive number less than or equal to 64KB-(size*512) where size is the size of the primary bootstrap.

The starting LBN format is defined by DSC and cannot be changed. The load address is defined by WRITEBOOT and cannot be changed.

Outputs:

- R0 - type of boot device
- R1 - (UNIBUS) address of the I/O page for the boot device's UNIBUS
- R2 - (MASSBUS) address of the device's MASSBUS adapter (UNIBUS) 18-bit UNIBUS address of the boot device's

BOOSBLOCK - reads in and starts boot cod

1-APR-1980 09:26:36 _DBB1:[BOOTS.SRC]BOOTBLOCK.MAR;1

```

000C 134 :
000C 135 :
000C 136 : R3 - unit number of the boot device
000C 137 : R5 - software boot control flags
000C 138 : R6 - physical address of the device-dependent ROM routine
000C 139 : that reads an arbitrary LBN into memory
000C 140 :
000C 141 : SP - <base_address + ^X200> of 64kb of good memory
000C 142 :
000C 143 : Implicit outputs:
000C 144 :
000C 145 : The routine preserves R0-R1, R3, R4, R5-R6, R8, R10-R11, AP, and SP.
000C 146 :
000C 147 : Transfers control to the 0th byte of the primary bootstrap
000C 148 : program.
000C 149 :
000C 150 : --
000C 151 :
000C 152 BOOSBLOCK CODE:
000C 153 PUSHAB FILE_STATS ; Start of device independent code.
000F 154 ; Move physical address of base of
6E F6 AF C0 000F 155 ADDL LOAD_ADDR,(SP) ; 64KB of memory.
0013 156 ; Add in relative load address. Result
0013 157 ; is physical address of load point.
0013 158 ; Leave on stack for final JMP inst.
0131 8F BB 0013 159 PUSHR #^M<R0,R4,R5,R8> ; Save 4 registers for temps.
54 E6 AF D0 0017 160 MOVL FILE_SIZE,R4 ; Get # of blocks in VMB.
58 E6 AF B0 001B 161 MOVW START_LBN,R8 ; Get upper word value of LBN.
EO AF E4 AF B0 001F 162 MOVW START_LBN+2,START_LBN ; Move lower word value of LBN into
0024 163 ; lower word position.
DE AF 58 B0 0024 164 MOVW R8,START_LBN+2 ; Move upper word value to upper
0028 165 ; word position.
58 D9 AF D0 0028 166 MOVL START_LBN,R8 ; Pickup the swapped LBN.
55 D9 AF D0 002C 167 MOVL LOAD_ADDR,R5 ; Get primary boot relative load addr.
10 AE DD 0030 168 PUSHL 16(SP) ; Copy physical transfer address to
0033 169 ; top of stack for those devices such
0033 170 ; as the TUSB which need physical
0033 171 ; rather than UNIBUS virtual addresses.
0033 172
0033 173 READ_BLOCK:
0033 174 JSB (R6) ; VMB read loop.
01 50 E8 0035 175 BLBS R0,NEXT_BLOCK ; Call ROM read LBN routine.
0038 176 HALT ; Branch on successful read.
0039 177 ; Halt on failure to read.
0039 178 NEXT_BLOCK:
55 00000200 8F C0 0039 179 ADDL #^X200,R5 ; Read next block.
6E 00000200 8F C0 0040 180 ADDL #^X200,(SP) ; Increment relative address 512 bytes.
BA AF D6 0047 181 INCL START_LBN ; Increment physical address one page.
58 B7 AF D0 004A 182 MOVL START_LBN,R8 ; Increment LBN number.
E2 54 F5 004E 183 SOBGTR R4,READ_BLOCK ; Next LBN is LBN+1.
8E D5 0051 184 TSTL (SP)+ ; If more blocks, loop.
0053 185 ; Pop now useless data from stack.
0053 186 :
0053 187 : The primary bootstrap program is now in physical memory starting at
0053 188 : the specified load address. Restore the saved registers, convert the
0053 189 : CSR address to an 18-bit UNIBUS address, and transfer control to the
0053 190 : program.

```

ZZ-ECVMB-1.0
BOOTBLOCK
V02-001

BOOSBLOCK - reads in and starts boot cod

H 1
9-JAN-1981

Fiche 1 Frame H1

Sequence 7

9-JAN-1981 05:45:42 VAX-11 Macro V02.46 Page 5
1-APR-1980 09:26:36 _DBB1:[BOOTS.SRC]BOOTBLOCK.MAR;1 (3)

```
BOOSBLOCK - reads in and starts boot cod
0053 191 ;
0053 192 ;
52 0131 8F BA 0053 193 POPR #^M<R0,R4,R5,R8> ; Restore registers.
FFFC0000 8F CA 0057 194 BICL #^XFFFC0000,R2 ; Reduce 32-bit CSR to 18-bit
9E 17 005E 195 ; CSR that VMB expects.
005E 196 JMP @ (SP)+ ; Jump to primary bootstrap program.
0060 197
0060 198 .END
```

ZZ-ECVMB-1.0 Symbol table
 BOOTBLOCK
 Symbol table
 BOOTBLOCK_CODE 0000000C R 01
 FILE_SIZE 00000000 R 01
 FILE_STATS 00000000 R 01
 LOAD_ADDR 00000008 R 01
 NEXT_BLOCK 00000039 R 01
 READ_BLOCK 00000033 R 01
 START_LBN 00000004 R 01

I 1
 9-JAN-1981

Fiche 1 Frame I1

Sequence 8

9-JAN-1981 05:45:42 VAX-11 Macro V02.46 Page 6
 1-APR-1980 09:26:36 _DBB1:[BOOTS.SRC]BOOTBLOCK.MAR;1 (3)

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
: ABS :	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
: BLANK :	00000060 (96.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	17	00:00:00.04	00:00:00.84
Command processing	24	00:00:00.25	00:00:01.69
Pass 1	34	00:00:00.40	00:00:02.45
Symbol table sort	0	00:00:00.00	00:00:00.00
Pass 2	57	00:00:00.36	00:00:02.02
Symbol table output	1	00:00:00.03	00:00:00.40
Psect synopsis output	1	00:00:00.00	00:00:00.01
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	134	00:00:01.08	00:00:07.42

The working set limit was 500 pages.

2142 bytes (5 pages) of virtual memory were used to buffer the intermediate code.

There were 10 pages of symbol table space allocated to hold 7 non-local and 0 local symbols.

198 source lines were read in Pass 1, producing 9 object records in Pass 2.

0 pages of virtual memory were used to define 0 macros.

! Macro library statistics !

Macro library name	Macros defined
_DRA7:[BOOTS.OBJ]BOOTS.MLB;1	0
_DRA7:[SYS.OBJ]LIB.MLB;1	0
_DRA7:[SYSLIB]STARLET.MLB;1	0
TOTALS (all libraries)	0

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

/LIS=LISS:BOOTBLOCK/OBJ=OBS:BOOTBLOCK MSRC\$:BOOTBLOCK/UPDATE=(ENHS:BOOTBLOCK)+EXECMLS/LIB+LIBS:BOOTS.MLB/LIB