

BBBBBBBBBBBB		00000000		00000000		TTTTTTTTTTTT		SSSSSSSSSS
BBBBBBBBBBBB		00000000		00000000		TTTTTTTTTTTT		SSSSSSSSSS
BBBBBBBBBBBB		00000000		00000000		TTTTTTTTTTTT		SSSSSSSSSS
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBB	BBB	000	000	000	000	TTT	SSS	
BBBBBBBBBBBB		00000000		00000000		TTT	SSSSSSSSSS	
BBBBBBBBBBBB		00000000		00000000		TTT	SSSSSSSSSS	
BBBBBBBBBBBB		00000000		00000000		TTT	SSSSSSSSSS	

[illegible]



BOOTBLOCK  
Table of contents

B 3

15-SEP-1984 23:40:18 VAX/VMS Macro V04-00

Page 0

(2) 50  
(3) 79

Declarations  
BOOTBLOCK - reads in and starts boot code

```

0000 1      .TITLE BOOTBLOCK
0000 2      .IDENT 'V04-000'
0000 3
0000 4
0000 5 :*****
0000 6 :*
0000 7 :*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 :*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 :*  ALL RIGHTS RESERVED.
0000 10 :*
0000 11 :*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 :*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 :*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 :*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 :*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 :*  TRANSFERRED.
0000 17 :*
0000 18 :*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 :*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 :*  CORPORATION.
0000 21 :*
0000 22 :*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 :*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 :*
0000 25 :*
0000 26 :*****
0000 27
0000 28 :++
0000 29
0000 30 : FACILITY:
0000 31
0000 32 :     Device-independent boot block for VAX
0000 33
0000 34 : ABSTRACT:
0000 35
0000 36 :     Reads a file (usually VMB.EXE) off the booting medium into
0000 37 :     memory and transfers control to the VMB code.
0000 38
0000 39 : AUTHOR:
0000 40
0000 41 :     Carol Peters      23 August 1979
0000 42
0000 43 : REVISION HISTORY:
0000 44
0000 45 :     Robert Rappaport      13 Sept 1979
0000 46 :     Simplified references to local data items.
0000 47
0000 48 :--

```



Declarations

```

0000 50      .SBTTL Declarations
0000 51
0000 52      ;
0000 53      ; Own storage.
0000 54      ;
0000 55
0000 56 FILE_STATS:
0000 57 FILE_SIZE:
00000000 0000 58      .LONG      0
0000 59 START_LBN:
00000000 0004 60      .LONG      0
00000000 0008 61 LOAD_ADDR:
00000000 0008 62      .LONG      0
000C 63
000C 64
000C 65
000C 66
000C 67
000C 68
000C 69
000C 70
000C 71
000C 72
000C 73
000C 74
000C 75
000C 76
000C 77

```

```

; Reserve space to contain
; # blocks in primary boot.
; Swapped words of start LBN.
; Load address for primary boot.
; NOTE - the load address here is
; relative to the base of the 64KB
; of physical memory in which we are
; currently running. As explained
; below, the UNIBUS (or MASSBUS) map
; registers numbered 0-127 will be
; mapped to this same 64KB. Therefore,
; this address, as is, can be used by
; UNIBUS (or MASSBUS) devices to
; pinpoint where to load the primary
; bootstrap program. However, to
; calculate the physical memory address
; corresponding to this relative
; address, we must add in the physical
; offset of the base of the 64KB.

```

BOOSBLOCK - reads in and starts boot cod

```

000C 79      .SBTTL BOOSBLOCK - reads in and starts boot code
000C 80
000C 81      :++
000C 82      : Functional description:
000C 83      :
000C 84      : The boot block code reads the primary bootstrap file into
000C 85      : physical memory a block at a time. The code calls the device-
000C 86      : dependent ROM subroutine once for each block in the bootstrap
000C 87      : file. Then the routine jumps to byte 0 of the loaded code.
000C 88
000C 89      Inputs:
000C 90
000C 91      R0      - type of boot device
000C 92      R1      - (UNIBUS) address of the I/O page for the boot device's
000C 93      UNIBUS
000C 94      (MASSBUS) address of the device's MASSBUS adapter
000C 95      R2      - (UNIBUS) 32-bit physical address of the boot device's
000C 96      CSR (bits <31:24> must be zero)
000C 97      (MASSBUS) adapter's controller/formatter number
000C 98      R3      - unit number of the boot device
000C 99      R5      - software boot control flags
000C 100     R6      - physical address of the device-dependent ROM routine
000C 101     that reads an arbitrary LBN into memory
000C 102
000C 103     SP      - <base_address + ^X200> of 64kb of good memory
000C 104
000C 105     Implicit inputs:
000C 106
000C 107     UNIBUS adapter map registers 0-127 are mapped to the 64kb of
000C 108     good memory. MR 0 maps to first page of memory, etc.
000C 109
000C 110     The boot block is loaded into the 1st page of the 64KB of
000C 111     memory, i.e. the page which corresponds to MR 0.
000C 112
000C 113     The first longword (bytes 0-3) of the boot block contains
000C 114     the size of the primary bootstrap.
000C 115
000C 116     The second longword (bytes 4-7) contains the starting LBN of
000C 117     the bootstrap file, expressed as swapped words.
000C 118
000C 119     The third longword (bytes 8-11) contains the relative offset
000C 120     from the base of the 64KB of memory into which we should load
000C 121     the primary bootstrap program. This must be a positive
000C 122     number less than or equal to 64KB-(size*512) where size is the
000C 123     size of the primary bootstrap.
000C 124
000C 125     The starting LBN format is defined by DSC and cannot be
000C 126     changed. The load address is defined by WRITEBOOT and cannot
000C 127     be changed.
000C 128
000C 129     Outputs:
000C 130
000C 131     R0      - type of boot device
000C 132     R1      - (UNIBUS) address of the I/O page for the boot device's
000C 133     UNIBUS
000C 134     (MASSBUS) address of the device's MASSBUS adapter
000C 135     R2      - (UNIBUS) 18-bit UNIBUS address of the boot device's

```



BOOSBLOCK - reads in and starts boot cod

```

000C 136 :
000C 137 :
000C 138 : CSR
000C 139 : (MASSBUS) adapter's controller/formatter number
000C 140 : R3 - unit number of the boot device
000C 141 : R5 - software boot control flags
000C 142 : R6 - physical address of the device-dependent ROM routine
000C 143 : that reads an arbitrary LBN into memory
000C 144 : SP - <base_address + ^X200> of 64kb of good memory
000C 145 :
000C 146 : Implicit outputs:
000C 147 : The routine preserves R0-R1, R3, R4, R5-R6, R8, R10-R11, AP, and SP.
000C 148 :
000C 149 : Transfers control to the 0th byte of the primary bootstrap
000C 150 : program.
000C 151 :
000C 152 : --
000C 153 :
000C 154 : BOOSBLOCK CODE:
000C 155 : PUSHAB FILE_STATS : Start of device independent code.
000F 156 : : Move physical address of base of
000F 157 : ADDL LOAD_ADDR,(SP) : 64KB of memory.
0013 158 : : Add in relative load address. Result
0013 159 : : is physical address of load point.
0013 160 : : Leave on stack for final JMP inst.
0013 161 :
0013 162 : PUSHHR #^M<R0,R4,R5,R8> : Save 4 registers for temps.
54 E6 AF D0 0017 162 : MOVL FILE_SIZE,R4 : Get # of blocks in VMB.
58 E6 AF B0 001B 163 : MOVW START_LBN,R8 : Get upper word value of LBN.
EO AF E4 AF B0 001F 164 : MOVW START_LBN+2,START_LBN : Move lower word value of LBN into
0024 165 : : lower word position.
DE AF 58 B0 0024 166 : MOVW R8,START_LBN+2 : Move upper word value to upper
0028 167 : : word position.
58 D9 AF D0 0028 168 : MOVL START_LBN,R8 : Pickup the swapped LBN.
55 D9 AF D0 002C 169 : MOVL LOAD_ADDR,R5 : Get primary boot relative load addr.
10 AE DD 0030 170 : PUSHL 16(SP) : Copy physical transfer address to
0033 171 : : top of stack for those devices such
0033 172 : : as the TU58 which need physical
0033 173 : : rather than UNIBUS virtual addresses.
0033 174 :
0033 175 : READ_BLOCK: : VMB read loop.
0033 176 : JSB (R6) : Call ROM read LBN routine.
01 66 16 0033 177 : BLBS R0,NEXT_BLOCK : Branch on successful read.
0035 178 : : Halt on failure to read.
0038 179 :
0039 180 : NEXT_BLOCK: : Read next block.
0039 181 : ADDL #^X200,R5 : Increment relative address 512 bytes.
0040 182 : ADDL #^X200,(SP) : Increment physical address one page.
0047 183 : INCL START_LBN : Increment LBN number.
58 B7 AF D0 004A 184 : MOVL START_LBN,R8 : Next LBN is LBN+1.
E2 54 F5 004E 185 : SOBGTR R4,READ_BLOCK : If more blocks, loop.
0051 186 : TSTL (SP)+ : Pop now useless data from stack.
0053 187 :
0053 188 :
0053 189 : The primary bootstrap program is now in physical memory starting at
0053 190 : the specified load address. Restore the saved registers, convert the
0053 191 : CSR address to an 18-bit UNIBUS address, and transfer control to the
0053 192 : program.

```

BOOTS\$BLOCK - reads in and starts boot cod

			0053	193 ;		
			0053	194		
	0131 8F	BA	0053	195	POPR	#^M<R0,R4,R5,R8> ; Restore registers.
52	FFFC0000 8F	CA	0057	196	BICL	#^XFFFC0000,R2 ; Reduce 32-bit CSR to 18-bit
			005E	197		; CSR that VMB expects.
	9E 17		005E	198	JMP	a(SP)+ ; Jump to primary bootstrap program.
			0060	199		
			0060	200	.END	



BOOTBLOCK  
Symbol table

BOOSBLOCK_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

-----  
! 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	35	00:00:00.08	00:00:00.29
Command processing	141	00:00:00.60	00:00:02.09
Pass 1	68	00:00:00.55	00:00:01.56
Symbol table sort	0	00:00:00.00	00:00:00.00
Pass 2	52	00:00:00.39	00:00:01.01
Symbol table output	2	00:00:00.01	00:00:00.01
Psect synopsis output	1	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	301	00:00:01.65	00:00:04.98

The working set limit was 900 pages.  
2158 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.  
200 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
_\$255\$DUA28:[BOOTS.OBJ]BOOTS.MLB;1	0
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	0
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	0
TOTALS (all libraries)	0

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:BOOTBLOCK/OBJ=OBJ\$:BOOTBLOCK MSRC\$:BOOTBLOCK/UPDATE=(ENH\$:BOOTBLOCK)+EXECML\$/LIB+LIB\$:BOOTS.MLB/LIB



0037 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

