

[illegible]

```

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
LLLLLLLLLLLLLL  IIIIII               SSSSSSSS
LLLLLLLLLLLLLL  IIIIII               SSSSSSSS

```

```
0000 1      .TITLE  BLOCK
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
0000 31     Facility:      magnetic tape acp
0000 32
0000 33     Abstract:
0000 34     this module handles the blocking and unblocking of the process
0000 35
0000 36
0000 37     Environment:
0000 38     starlet operating system, including privileged system services
0000 39     and internal exec routines.
0000 40
0000 41
0000 42     Author:  DEBORAH H. GILLESPIE,      Creation Date:      05-JUL-1977
0000 43
0000 44     Modified By:
0000 45
0000 46     V02-005 DMW00025      David Michael Walp      20-Jul-1981
0000 47     Changed free page handling to not contract P0 region
0000 48
0000 49     V02-004 DMW00010      David Michael Walp      14-Mar-1981
0000 50     Changed calculation of CCB address using GET_CCB
0000 51
0000 52     V02-003 KDM0037      Kathleen D. Morse      12-Feb-1981
0000 53     Change non-kernel mode references to SCH$GL_CURPCB
0000 54     to use CTL$GL_PCB instead.
0000 55
0000 56     V02-002 REFORMAT      D M WALP      25-JUL-1980
0000 57
```



```
0000 58 : Revision History:
0000 59 :
0000 60 :     D. H. GILLESPIE, VERSION a0001, 12-MAY-1978
0000 61 :     a0001 - change current_vcb to register
0000 62 : --
0000 63 :
0000 64 :
0000 65 : Include Files:
0000 66 :     .include mtadef.mar
0000 67 :
0000 68 :
0000 69 :
0000 70 : Macros:
0000 71 :
0000 72 :
0000 73 :     $PCBDEF
0000 74 :
0000 75 :
0000 76 :
0000 77 :
0000 78 :
0000 79 : Equated Symbols:
0000 80 :
0000 81 :
0000 82 :
00000000 0000 83 :     ASTEXIT = 0                ; ast exit change mode code
00000001 0000 84 :     EXEC_MODE = 1             ; exec mode
0000 85 :
0000 86 :
0000 87 : displacements of interesting variables on stack
0000 88 :
0000 89 :
0000000C 0000 90 :     PREVFP = 12                ; location on stack of saved
0000 91 :                                ; fp of caller
0000 92 :
0000 93 :
0000 94 : parameters for kernel_block
0000 95 :
0000 96 :
00000004 0000 97 :     REASON = 4                 ; mask in status to set indicating
0000 98 :                                ; reason for block
00000008 0000 99 :     PAGE = 8                  ; first argument is page address
0000 100 :
0000 101 :     $VCBDEF                    ; define volume control block
0000 102 :     $VVPDEF                    ; define volume virtual page
0000 103 :
0000 104 :
0000 105 : Own Storage:
0000 106 :
0000 107 :
```

```
0000 109
0000 110 :++
0000 111
0000 112 : BLOCK - this routine handles the blocking of current request
0000 113
0000 114 : Calling sequence:
0000 115 :     call    arglist,block
0000 116
0000 117 : Input Parameters:
0000 118 :     reason(ap) - mask of status bit to be set indicating reason for block
0000 119
0000 120 : Implicit Inputs:
0000 121 :     the exec stack
0000 122
0000 123 : Output Parameters:
0000 124 :     none
0000 125
0000 126 : Implicit Outputs:
0000 127 :     virtual page(s) containing stack and impure area
0000 128
0000 129 : Routine Value:
0000 130 :     none
0000 131
0000 132 : Side Effects:
0000 133 :     the request's exec stack and impure area are saved
0000 134 :--
0000 135
0000 136
0000 137 :.PSECT $CODE$,NOWRT,LONG
0000 138
0000 139 BLOCK::
OFFC 0000 140 :.WORD  ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>; save registers
0002 141
0002 142 :
0002 143 : block current process
0002 144 :
0002 145
60 50 0C AD D0 0002 146 10$: MOVL    PREVFP(FP),R0          ; get previous frame pter
00000000'8F D1 0006 147      CMPL    #EXCEPT_HNDLR,(R0)      ; does it contain the exception handler
50 50 05 13 000D 148      BEQLU    20$          ; yes,leave fp at frame before this one
56 50 EE 11 0012 149      MOVL    R0,FP          ; try next one
0000000E'8F C3 0014 150      BRB     10$          ; check it
57 56 5E C1 0018 151 20$:  SUBL3    SP,R0,R6          ; calc # of bytes of stack to save
0000000E'8F C1 0018 152      ADDL3    #IMPURE_SIZE+2+VVP$K_LENGTH,- ; save impure area, length and
57 56 001E 153      R6,R7          ; include fixed area of virtual blocks
0002 154
51 50 57 00000200 58 D4 0020 155      CLRL    R8          ; r7-r8 quotient
8F 7B 0022 156      EDIV    #512,R7,R0,R1      ; calc # of virtual pages needed
51 D5 002B 157      TSTL    R1          ; is another page needed
02 13 002D 158      BEQLU    50$          ; no
50 D6 002F 159      INCL    R0          ; yes, inc # of pages needed
7E DF 0031 160 50$:  PUSHAL   -(SP)          ; allocate space to store address of
0033 161      ; free page and push that addr
0033 162      PUSHL    R0          ; # of pages needed
00000000'EF 02 FB 0035 163      CALLS    #2,GET_FREE_PAGE      ; get virtual page(s)
0A A3 02 90 003C 164      POPR     #^M<R3>          ; get free page address off stack
003E 165      MOVB     #VVP_TYPE,VVP$B_TYPE(R3); set block type to virtual page
```



```
0042 166
0042 167
0042 168 : change to kernel mode to insert in volume virtual page queue
0042 169
0042 170
04 53 DD 0042 171 PUSHL R3 ; address of page
AC DD 0044 172 PUSHL REASON(AP) ; reason for block
02 DD 0047 173 PUSHL #2 ; one argument
5E DD 0049 174 PUSHL SP ; address of argument list
000000D0'EF 9F 004B 175 PUSHAB KERNEL_BLOCK
00000000'9F 05 FB 0051 176 CALLS #5,@#SYS$CMKRN
00000000'EF 0000'8F 28 0058 177 MOV C #IMPURE_SIZE,USER_STATUS,-
OC A3 0061 178 VVP$K_LENGTH(R3) ; move impure area to virtual page
83 56 B0 0063 179 MOVW R6,(R3)+ ; amount of stack
63 6E 56 28 0066 180 MOV C R6,(SP),(R3) ; move stack
00000000'EF D4 006A 181 CLRL IO_PACKET ; prevent completion of io
04 0070 182 RET ; return to control with all
0071 183 ; register clobbered
```

```
0071 185
0071 186 :++
0071 187 :
0071 188 : UNBLOCK - this routine locates the stack and impure area for the blocked
0071 189 : process. it restores the impure area, locates the exception
0071 190 : handler on the stack and overlays the stack beginning just
0071 191 : following the exception handler frame. it returns the virtual
0071 192 : address space to the free page list. it then returns to where
0071 193 : the blocked process left off.
0071 194 :
0071 195 : Calling sequence:
0071 196 : call unblock
0071 197 :
0071 198 : Input Parameters:
0071 199 : none
0071 200 :
0071 201 : Implicit Inputs:
0071 202 : the exec stack, current_vcb(in r11) and its associated virtual pages
0071 203 :
0071 204 : Output Parameters:
0071 205 : none
0071 206 :
0071 207 : Implicit Outputs:
0071 208 : virtual page(s) containing saved data are returned to the free pages
0071 209 : list stack is restored. the exec stack is restored to state before
0071 210 : process was blocked. if an ast triggered the unblocking of the proces,
0071 211 : an exit ast is done.
0071 212 :--
0071 213 :
0071 214 :
0071 215 : Routine Value:
0071 216 : none
0071 217 :
0071 218 : Side Effects:
0071 219 :
0071 220 :
00000071 221 : .PSECT $CODE$,NOWRT, LONG
0071 222 :
0071 223 : .EXTRN GET_CCB
0071 224 :
0071 225 UNBLOCK::
00000071 226 : .WORD ^M<> ; don't save registers
0073 227 :
0073 228 : MOVL VCB$$_VPBL(R11),R6 ; pickup tail of virtual page list,
0077 229 : ; contains saved process status
0077 230 :
0077 231 : MOVC3 #IMPURE_SIZE,-
0078 232 : VVP$K_LENGTH(R6),-
007D 233 : USER_STATUS ; restore impure area first
0082 234 :
0082 235 : ; locate exception handler on stack
0082 236 :
0082 237 :
0082 238 10$: MOVL PREVFP(FP),FP ; pickup previous fp
0086 239 : CMPL #EXCEPT_HNDLR,(FP) ; is this the exception frame?
008D 240 : BNEQU 10$ ; not found yet
008F 241 :
```



```
008F 242 :  
008F 243 : found exception handler on stack  
008F 244 :  
008F 245 :  
50 61 3C 008F 246 MOVZWL (R1),R0 ; convert word length to long length  
5D 50 C2 0092 247 SUBL2 R0,FP ; restore stack pointer by subtracting  
0095 248 ; length from exception frame address  
6D 5E 5D D0 0095 249 MOVL FP,SP ; reset stack pointer  
61 81 28 0098 250 MOVC3 (R1)+,(R1),(FP) ; restore stack  
009C 251 :  
009C 252 :  
009C 253 : now give back the pages used to store the stack  
009C 254 :  
009C 255 :  
7E D4 009C 256 CLRL -(SP) ; no parameters  
5E DD 009E 257 PUSHL SP ; address of argument list  
000000DD'EF 9F 00A0 258 PUSHAB KERNEL_UNBLOCK ; address of subroutine to execute in  
00000000'9F 03 FB 00A6 259 ; kernel mode  
00AD 260 CALLS #3,@#SYSS$CMKRNL ; change mode to kernel so can write  
00AD 261 ; to sys space  
7E D4 00AD 262 CLRL -(SP) ; do not contract the MTAACP space  
56 DD 00AF 263 PUSHL R6 ; addr of page  
00000000'EF 02 FB 00B1 264 CALLS #2,RET_FREE_PAGE ; return pages  
00B8 265 :  
00B8 266 :  
00B8 267 : if there is an active ast for exec mode, exit from it  
00B8 268 :  
00B8 269 :  
51 00000000'GF D0 00B8 270 MOVL G^CTL$GL_PCB,R1 ; address of pcb for this process  
OB OC A1 01 E1 00BF 271 BBC #EXEC_MODE,PCB$B_ASTACK(R1),20$  
00C4 272 $SETAST_S #0 ; disable ast delivery  
00 00 00CD 273 CHMK -S^#ASTEXIT ; return from ast  
00CF 274 :  
04 00CF 275 20$: RET ; return to where process blocked
```



```
00D0 277
00D0 278 :++
00D0 279
00D0 280 : KERNEL_BLOCK
00D0 281 :   This routine inserts a virtual page into to volume's virtual
00D0 282 :   page queue in the volume control block and set the reason for the block
00D0 283 :
00D0 284 : Calling sequence:
00D0 285 :   callg arglist, kernel_block
00D0 286 :   called in kernel mode
00D0 287 :
00D0 288 : Input Parameters:
00D0 289 :   reason(ap)      - mask in status to set indicating reason for block
00D0 290 :   page(ap)        - address of page to insert at tail of the virtual page
00D0 291 :                   queue
00D0 292 :
00D0 293 : Implicit Inputs:
00D0 294 :   current-vcb - address of current volume control block
00D0 295 :
00D0 296 : Output Parameters:
00D0 297 :   none
00D0 298 :
00D0 299 : Implicit Outputs:
00D0 300 :   insert complete
00D0 301 :
00D0 302 : Routine Value:
00D0 303 :   none
00D0 304 :
00D0 305 : Side Effects:
00D0 306 :   none
00D0 307 :
00D0 308 :--
00D0 309
00D0 310 KERNEL_BLOCK:
00D0 311   .WORD ^M<> ; save register one
00D2 312   BISB2 REASON(AP),VCB$B_STATUS(R11); set reason for block
00D7 313   INSQUE @PAGE(AP),@VCB$B_VPBL(R11); insert in queue
00DC 314   RET
```

0B AB 04 AC 0000 00D0 311
40 BB 08 BC 88 00D2 312
0E 00D7 313
04 00DC 314

```
00DD 316
00DD 317 :++
00DD 318 :
00DD 319 : KERNEL_UNBLOCK
00DD 320 : This routine removes the tail of the virtual page queue in the
00DD 321 : volume control block and the volume set reasons for waiting are cleared
00DD 322 : it also requests any stalled i/o
00DD 323 :
00DD 324 : Calling sequence:
00DD 325 : callg kernel_unblock
00DD 326 : called in kernel mode
00DD 327 :
00DD 328 : Input Parameters:
00DD 329 : r11 - address of volume control block
00DD 330 :
00DD 331 : Implicit Inputs:
00DD 332 : none
00DD 333 :
00DD 334 : Output Parameters:
00DD 335 : none
00DD 336 :
00DD 337 : Implicit Outputs:
00DD 338 : one item removed from tail of virtual page queue
00DD 339 : reason's for blocking process are cleared
00DD 340 :
00DD 341 : Routine Value:
00DD 342 : none
00DD 343 :
00DD 344 : Side Effects:
00DD 345 : none
00DD 346 :
00DD 347 :--
00DD 348 :
00DD 349 : KERNEL_UNBLOCK:
00DD 350 : .WORD ^M<R7,R8> ; save one register
58 3C AB 0180 00DF 351 1$: MOVL VCB$$_VPFL(R11),R8 ; get addr of virtual page for this vol
58 01A8 D8 0F 00E3 352 REMQUE @VVP$$_STALLIOBL(R8),R8 ; pickup packet at end of stalled
00E8 353 ; i/o queue
00E8 354 BVS 2$ ; packet not found
00EA 355
00EA 356 ;
00EA 357 ; requeue all stalled i/o
00EA 358 ;
00EA 359 ;
00000000'FF 68 0E 00EA 360 INSQUE (R8),@QUEUE_HEAD
EC 11 00F1 361 BRB 1$
00F3 362 2$: BICB2 #<VCB$$_WAIREWIND + VCB$$_WAIMOUVOL + VCB$$_WAIUSRLBL>,-
00F5 363 VCB$$_STATUS(R11) ; reason for blocking
58 0B AB 0F 00F7 364 REMQUE @VCB$$_VPBL(R11),R8 ; remove one entry
00FB 365
00FB 366 ;
00FB 367 ; assign channel
00FB 368 ;
00FB 369 ;
00000000'EF DD 00FB 370 PUSHL IO_CHANNEL ; calc addr of channel control block
00000000'EF 01 FB 0101 371 CALLS #1,GET_CCB ; via GET_CCB in kernel mode
60 00000000'EF DD 0108 372 MOVL CURRENT_UCB,(R0) ; stuff channel with current ucb
```


BLOCK
V04-000

H 13

16-SEP-1984 02:02:53 VAX/VMS Macro V04-00
5-SEP-1984 02:10:25 [MTAACP.SRC]BLOCK.MAR;1

Page 9
(9)

04	010F	373	
	010F	374	RET
	0110	375	
	0110	376	.END

C
S
A
C
C
O
U
N
T
S
P
I
P
I
C
S
P
S
P
C
A
T
I
N
I
T
I
O
N
M

BLOCK
Symbol table

I 13

16-SEP-1984 02:02:53 VAX/VMS Macro V04-00
5-SEP-1984 02:10:25 [MTAACP.SRC]BLOCK.MAR;1

Page 10
(9)

AQB_TYPE	= 00000005		
ASTEXIT	= 00000000		
BLOCK	00000000	RG	02
CTL\$GL_PCB	*****	X	02
CURRENT_UCB	*****	X	02
EXCEPT_RNDLR	*****	X	02
EXEC_MODE	= 00000001		
FCB_TYPE	= 00000000		
GET_CCB	*****	X	02
GET_FREE_PAGE	*****	X	02
IMPORE_SIZE	*****	X	02
IO_CHANNEL	*****	X	02
IO_PACKET	*****	X	02
KERNEL_BLOCK	000000D0	R	02
KERNEL_UNBLOCK	000000D0	R	02
MVL_TYPE	= 00000004		
PAGE	= 00000008		
PCBSB_ASTACT	= 0000000C		
PREVP	= 0000000C		
QUEUE_HEAD	*****	X	02
REASON	= 00000004		
RET_FREE_PAGE	*****	X	02
RVT_TYPE	= 00000003		
SYSSCMKRN	*****	X	02
SYSSSETAST	*****	GX	02
UNBLOCK	00000071	RG	02
USER_STATUS	*****	X	02
VCBSB_STATUS	= 0000000B		
VCBSL_VPBL	= 00000040		
VCBSL_VPFL	= 0000003C		
VCBSM_WAIMOUVOL	= 00000004		
VCBSM_WAIREWIND	= 00000008		
VCBSM_WAIUSRLBL	= 00000010		
VCB_TYPE	= 00000002		
VVP\$B_TYPE	0000000A		
VVP\$K_LENGTH	0000000C		
VVP\$L_BACKWARD	00000004		
VVP\$L_FORWARD	00000000		
VVP\$L_STALLIOBL	000001A8		
VVP\$L_STALLIOFL	000001A4		
VVP\$L_STATUS	0000019C		
VVP\$T_HDR1	0000000C		
VVP\$T_HDR2	0000005C		
VVP\$T_HDR3	000000AC		
VVP\$T_HDR4	000000FC		
VVP\$T_SCRATCH	0000014C		
VVP\$W_SIZE	00000008		
VVP_TYPE	= 00000002		
WCB_TYPE	= 00000001		

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	000001AC (428.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
\$CODE\$	00000110 (272.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC LONG

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	41	00:00:00.07	00:00:01.61
Command processing	145	00:00:00.65	00:00:04.84
Pass 1	194	00:00:04.00	00:00:17.65
Symbol table sort	0	00:00:00.43	00:00:01.19
Pass 2	77	00:00:01.25	00:00:05.37
Symbol table output	7	00:00:00.06	00:00:00.47
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	469	00:00:06.49	00:00:31.17

The working set limit was 1050 pages.
21841 bytes (43 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 358 non-local and 7 local symbols.
559 source lines were read in Pass 1, producing 14 object records in Pass 2.
16 pages of virtual memory were used to define 15 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	2
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	4
TOTALS (all libraries)	6

384 GETS were required to define 6 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:BLOCK/OBJ=OBJ\$:BLOCK MSRC\$:MTADEF1/UPDATE=(ENH\$:MTADEF1)+MSRC\$:BLOCK/UPDATE=(ENH\$:BLOCK)+EXECML\$/LIB

0253

AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY