

[illegible]

```

SSSSSSSS HH HH 000000 WW WW FFFFFFFF IIIIII LL EEEEEEEEE SSSSSSS
SSSSSSSS HH HH 000000 WW WW FFFFFFFF IIIIII LL EEEEEEEEE SSSSSSS
SS HH HH 00 00 WW WW FF IIII LL EEEEEEEEE SS SSSSSSS
SS HH HH 00 00 WW WW FF IIII LL EEEEEEEEE SS SSSSSSS
SS HH HH 00 00 WW WW FF IIII LL EEEEEEEEE SS SSSSSSS
SSSSSS HH HH 00 00 WW WW FFFFFFFF IIII LL EEEEEEEEE SSSSSS
SSSSSS HH HH 00 00 WW WW FFFFFFFF IIII LL EEEEEEEEE SSSSSS
SS HH HH 00 00 WW WW FF IIII LL EEEEEEEEE SS SSSSSS
SS HH HH 00 00 WW WW FF IIII LL EEEEEEEEE SS SSSSSS
SS HH HH 00 00 WW WW FF IIII LL EEEEEEEEE SS SSSSSS
SSSSSS HH HH 000000 WW WW FF IIIIII LLLLLLLLLL EEEEEEEEE SSSSSSS
SSSSSS HH HH 000000 WW WW FF IIIIII LLLLLLLLLL EEEEEEEEE SSSSSSS

LL IIIIII SSSSSSS
LL IIIIII SSSSSSS
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
LL IIIIII SSSSSSS
LLLLLLLLLL IIIIII SSSSSSS
LLLLLLLLLL IIIIII SSSSSSS

```


SHOW\$FILES
Table of contents

- list open files on a disk device L 12

15-SEP-1984 23:46:26 VAX/VMS Macro V04-00

Page 0

(1)	2	copyright notice
(1)	29	Program description
(2)	90	declarations
(3)	122	storage definitions
(4)	208	show\$files -- entry point for this module
(5)	289	get_device, get device spec and assign a channel
(6)	317	count_fcbwcb -- count the number of FCB's and WCB's
(7)	410	save_dev_info -- save device information
(8)	470	save_all_fcbs Save all FCB's for a volume
(9)	559	display_info, Display all files from FCB list
(10)	615	display_file, Display information on a single file
(11)	707	get_filename, Get file name from FID

SHOW\$FILES
V04-000

- list open files on a disk device M 12

15-SEP-1984 23:46:26 VAX/VMS Macro V04-00 Page 1
4-SEP-1984 23:22:34 [CLIUTL.SRC]SHOWFILES.MAR;1 (1)

```
0000 1 .title show$files - list open files on a disk device
0000 2 .sbttl copyright notice
0000 3 .ident 'V04-000'
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 29 .sbtll Program description
0000 30 :++
0000 31 Facility
0000 32
0000 33 Utility to list open files on a disk
0000 34
0000 35 Abstract
0000 36
0000 37 This program lists all open files on a disk device
0000 38 in order to find out who is accessing the device.
0000 39
0000 40 Environment
0000 41
0000 42 Native mode, User mode
0000 43
0000 44 Author
0000 45
0000 46 Tim Halvorsen, Jan 1980
0000 47
0000 48 Modified by:
0000 49
0000 50 V03-012 AEW0002 Anne E. Warner 26-Apr-1984
0000 51 Made fix to clear the variable containing the PID of
0000 52 a system file at output. This variable was not being
0000 53 reset after a file with a non-zero PID was processed.
0000 54
0000 55 V03-011 AEW0001 Anne E. Warner 18-Apr-1984
0000 56 Changed the output processing from LIB$PUT_OUTPUT to
0000 57 SHOW$WRITE_LINE in [CLIUTL.SRC]SHOWIO.B32. This change
0000 58 was made for continuity in the SHOW Utility and because
0000 59 LIB$PUT_OUTPUT does not work when SYS$OUTPUT is a user
0000 60 defined file.
0000 61
0000 62 V03-010 CWH3010 CW Hobbs 28-Feb-1984
0000 63 Shorten the heading so that long device names do not cause
0000 64 screen wraps.
0000 65
0000 66 V03-009 TCM0001 Trudy C. Matthews 10-Jun-1983
0000 67 Change sense of input flag to IOC$CVT_DEVNAM to match
0000 68 change in that routine.
0000 69
0000 70 V03-008 GAS0136 Gerry smith 19-May-1983
0000 71 Fix the flags longword so that it gets propagated
0000 72 correctly. Also make sure that IOC$CVT_DEVNAM returns
0000 73 the full device name.
0000 74
0000 75 V03-007 CWH1002 CW Hobbs 25-Feb-1983
0000 76 Use extended pid for getjpi and display.
0000 77
0000 78 V03-006 GAS0104 Gerry Smith 31-Jan-1983
0000 79 Remove the pre-parsing of the device before assigning
0000 80 a channel to it, since $ASSIGN does that anyway.
0000 81
0000 82 V03-005 GAS0099 Gerry Smith 7-Jan-1983
0000 83 Minor modification to run with new SHOW.
0000 84
0000 85 V03-004 KDM0002 Kathleen D. Morse 28-Jun-1982

```


SHOWFILES
V04-000

- list open files on a disk device B 13
Program description

15-SEP-1984 23:46:26 VAX/VMS Macro V04-00 Page 3
4-SEP-1984 23:22:34 [CLIUTL.SRC]SHOWFILES.MAR;1 (1)

0000 86 ;
0000 87 ;
0000 88 ;--

Added \$PRDEF.


```

0000 90      .sbtll declarations
0000 91      :
0000 92      :
0000 93      :
0000 94      $dscdef      : Descriptor definitions
0000 95      $jpiodef     : GETJPI definitions
0000 96      $ddbdef      : device data block
0000 97      $ucbdef      : unit control block
0000 98      $vcbdef      : Volume control block
0000 99      $rvtdef      : Relative volume table
0000 100     $fcbdef      : File control block
0000 101     $wcbdef      : Window control block
0000 102     $dyndef      : Type codes for VMS blocks
0000 103     $ssdef       : System error messages
0000 104     $ccbdef      : CCB definitions
0000 105     $ipldef      : IPL definitions
0000 106     $prdef       : Processor register definitions
0000 107     $prvdef      : Privilege bit definitions
0000 108     $dcdef       : Device class definitions
0000 109
0000 110     :
0000 111     : Define the position of the options bits in DEVISL_BITLIS
0000 112     :
0000 113     $vield devi,3,<-
0000 114         <file,,m>, -      : /FILES bit
0000 115         <syst,,m>, -      : /SYSTEM bit
0000 116         <user,,m>, -      : /NOSYSTEM bit
0000 117         <wind,,m>,-        : /WINDOWS bit
0000 118         >
0000 119

```

```
0000 122 .sbttl storage definitions
0000 123 :
0000 124 : storage definitions
0000 125 :
0000 126 :
00000000 127 .psect show$rwdata,noexe,wrt
0000 128
0000 129 devi$l_bitlis:
00000004 0000 130 .blk1 1 ; Options longword
0004 131
0004 132 ;**** the following block of code must be together:
0004 133 fcb_length:
00000008 0004 134 .blk1 1 ; Length of buffer in use
0008 135
0008 136 fcb_buffer:
0000000C 0008 137 .blk1 1 ; Address of the buffer
000C 138
000C 139 fcb_count:
00000010 000C 140 .blk1 1 ; Number of FCB's to display
0010 141
0010 142 wcb_length:
00000014 0010 143 .blk1 1 ; Length of the buffer in use
0014 144
0014 145 wcb_buffer:
00000018 0014 146 .blk1 1 ; Address of the buffer
0018 147 ; The 16 byte entry consists of the PID,
0018 148 ; the number of window segments, the
0018 149 ; total size of all the windows, and the
0018 150 ; access status byte.
0018 151
0018 152 wcb_count:
0000001C 0018 153 .blk1 1 ; Number of WCB's to display
001C 154
001C 155 def_disk:
49 44 24 53 59 53 00000024'010E0000' 001C 156 .ascid /SYS$DISK:/ ; Descriptor for default disk
3A 4B 53 002A
002D 157 ;****
002D 158
002D 159 procnambuf:
0000003C 002D 160 .blk1 15 ; Process name buffer
003C 161 process_name:
00000000 003C 162 .long 0
0000002D' 0040 163 .address procnambuf
0044 164
0044 165 jpi_pid:
00000000 0044 166 .long 0 ; Epid for the getjpi and fao
0048 167
0048 168 newbuflim:
00000000 00000000 0048 169 .long 0,0 ; descriptor for locking dynamic buffers
0050 170
0050 171 null_wcblock:
00000000 0050 172 .long 0 ; PID
00000000 0054 173 .long 0 ; window count
00000000 0058 174 .long 0 ; window size
00000000 005C 175 .long 0 ; access flags
0060 176
00000000 177 .psect show$rodata,noexe,nowrt
```



```
0000 178
0000 179 jpi_procnam:
000F 0000 180 .word 15 ; Length of output buffer
031C 0002 181 .word jpi$prcnam ; Request type
0000002D 0004 182 .address procnambuf ; Address of output buffer
0000003C 0008 183 .address process_name ; Address to receive length
00000000 000C 184 .long 0
0010 185
0010 186 ;**** buflim depends on having fcb_length, fcb_buffer, and def_disk all together
0010 187 buflim:
00000004 0010 188 .address fcb_length ; descriptor showing address range
0000001C 0014 189 .address def_disk ; for data that can't page
0018 190
0018 191 codelim:
0000016D 0018 192 .address count_fcwcb ; descriptor showing address range
000003F5 001C 193 .address display_file ; for code that can't page
0020 194
0020 195 ;
0020 196 ; FAO control strings and the header for output
0020 197 ;
20 53 41 35 31 21 00000028 010E0000 0020 198 faowin: .ascid '!15AS !XL !8UL!AS !8UL !AS'
20 53 41 21 4C 55 38 21 20 4C 58 21 002E
53 41 21 20 20 4C 55 38 21 003A
20 53 41 35 31 21 0000004B 010E0000 0043 199 faofil: .ascid '!15AS !XL !AS'
53 41 21 20 20 4C 58 21 0051
43 00000061 010E0000 0059 200 cath: .ascid 'C'
20 0000006A 010E0000 0062 201 nocath: .ascid ''
65 6C 69 46 2F 21 00000073 010E0000 006B 202 disk: .ascid '!/Files accessed on device !AS on !%D!/'
6F 20 64 65 73 73 65 63 63 61 20 73 0079
53 41 21 20 65 63 69 76 65 64 20 6E 0085
2F 21 44 25 21 20 6E 6F 20 0091
73 65 63 6F 72 50 000000A2 010E0000 009A 203 hedwin: .ascid 'Process name PID Win Cnt Win Size File name'
20 20 20 20 20 20 65 6D 61 6E 20 73 00A8
43 20 6E 69 57 20 20 20 20 44 49 50 00B4
7A 69 53 20 6E 69 57 20 20 20 74 6E 00C0
65 6D 61 6E 20 65 6C 69 46 20 20 65 00CC
73 65 63 6F 72 50 000000E0 010E0000 00DB 204 hedfil: .ascid 'Process name PID File name'
20 20 20 20 20 20 65 6D 61 6E 20 73 00E6
65 6C 69 46 20 20 20 20 44 49 50 00F2
65 6D 61 6E 20 00FE
0103 205
00000000 206 .psect show$code,exe,nowrt
```

```
0000 208 .sbtll show$files -- entry point for this module
0000 209
0000 210
0000 211 This is the main routine of the program. Based on the
0000 212 options in the command line, subroutines are called to
0000 213 display open files on a particular device.
0000 214
0000 215 Inputs:
0000 216
0000 217 4(AP) = address of translated device descriptor
0000 218 8(AP) = flags longword
0000 219
0000 220 Outputs:
0000 221
0000 222 The open files are printed on SYS$OUTPUT
0000 223
0000 224
003C 0000 225 .entry show$files,^m<r2,r3,r4,r5>
0002 226
00000000'EF 08 BC D0 0002 227 movl a8(ap),devi$l_bitlis ; get the flags
00000000'EF 30 D3 000A 228 bitl #devi$m_user- ; check to see if either option
0011 230 !devi$m_syst, - ; was set. If not, then set both
0011 231 devi$l_bitlis
00000000'EF 07 12 0011 232 bneq 10$
00000000'EF 30 C8 0013 233 bisl #devi$m_user- ; set both bits, saying we want
001A 234 !devi$m_syst, - ; both system and user files
001A 235 devi$l_bitlis
001A 236
00000153'EF 16 001A 237 10$: jsb get_device ; assign a channel and put it in r3
0020 238 ; note that r2-r5 are changed
28 50 E9 0020 239 blbc r0,15$ ; branch on error
0023 240 $lkwsel_s inadr=buflim ; lock data buffer in working set
14 50 E9 0034 241 blbc r0,15$ ; branch on error
0037 242 $lkwsel_s inadr=codelim ; lock code pages in working set
01 50 E8 0048 243 blbs r0,20$ ; xfer if all ok
04 004B 244 15$: ret ; return with status
53 DD 004C 245 20$: pushl r3 ; channel number on the stack
01 DD 004E 246 pushl #1
5E DD 0050 247 pushl sp
016D'CF 9F 0052 248 pushab w^count_fcbwcb
00000000'GF 02 FB 0056 249 calls #2,g^sys$cmkrnl ; count the number of FCB's and WCB's
5E 08 C0 005D 250 addl2 #8,sp ; clean up the stack
50 0000000C'EF 0A C1 0060 251 addl3 #10,fcb_count,r0 ; set minimum
50 50 0A C7 0068 252 divl3 #10,r0,r0 ; 10% overdraft
0000000C'EF 50 000007BC 8F C5 006C 253 mull3 #11*fcb$length,r0,fcb_count ; amount needed
50 00000018'EF 0A C1 0078 254 addl3 #10,wcb_count,r0 ; set minimum
50 50 0A C7 0080 255 divl3 #10,r0,r0 ; 10% overdraft
00000018'EF 50 000000B0 8F C5 0084 256 mull3 #11*16,r0,wcb_count ; amount needed
7E 00000018'EF 0000000C'EF C1 0090 257 addl3 fcb_count,wcb_count,-(sp) ; calculate total needed
0000004C'EF 6E D0 009C 258 movl (sp),newbuflim+4 ; save size for later
50 5E D0 00A3 259 movl sp,r0 ; save address for later
00000008'EF 9F 00A6 260 pushab fcb_buffer ; get some memory
50 DD 00AC 261 pushl r0
00000000'GF 02 FB 00AE 262 calls #2,g^lib$get_vm
01 50 E8 00B5 263 blbs r0,30$ ; xfer if all ok
04 00B8 264 ret ; else return with status posted
```


00000048'EF	00000008'EF	D5	00B9	265	30\$:	tstl	(sp)+	; clean the stack
0000004C'EF	00000008'EF	D0	00BB	266		movl	fcf_buffer,newbuflim	; set starting address to lock
		C0	00C6	267		addl2	fcf_buffer,newbuflim+4	; set ending address to lock
			00D1	268		\$lkws	inadr=newbuflim	; lock new buffers into memory
	01 50	E8	00E2	269		blbs	r0,40\$	; xfer if all ok
		04	00E5	270		ret		; else exit stage right
00000008'EF	0000000C'EF	C1	00E6	271	40\$:	addl3	fcf_count,fcf_buffer,wcb_buffer	; set address of WCB buffer area
	00000014'EF		00F1					
	53	DD	00F6	272		pushl	r3	; channel number on stack
	01	DD	00F8	273		pushl	#1	
	5E	DD	00FA	274		pushl	sp	
	0227'CF	9F	00FC	275		pushab	w^save_dev_info	
00000000'GF	02	FB	0100	276		calls	#2,g^sys\$cmkrnl	; Save device information
5E	08	C0	0107	277		addl	#8,sp	; Pop cmrnl arguments off stack
45 50		E9	010A	278		blbc	r0,90\$	; branch if error
			010D	279		\$ulwset_s	inadr=buflim	; unlock data pages
31 50		E9	011E	280		blbc	r0,90\$	; branch if error
			0121	281		\$ulwset_s	inadr=codelim	; unlock code pages
1D 50		E9	0132	282		blbc	r0,90\$	; branch if error
			0135	283		\$ulwset_s	inadr=newbuflim	; unlock dynamic buffer area
09 50		E9	0146	284		blbc	r0,90\$	; xfer if in error
	52	DD	0149	285		pushl	r2	; Address of device descriptor
0000038F'EF	01	FB	014B	286		calls	#1,display_info	; Display the saved info
		04	0152	287	90\$:	ret		

```

0153 289 .sbttl get_device, get device spec and assign a channel
0153 290 :---
0153 291
0153 292 This routine takes the device specification, if any, and assigns
0153 293 a channel to it.
0153 294
0153 295 Inputs:
0153 296
0153 297 4(ap) = address of device string descriptor
0153 298
0153 299 Outputs:
0153 300
0153 301 r3 = channel number from assign
0153 302
0153 303 r4,r5 are destroyed
0153 304
0153 305 :---
0153 306
0153 307 get_device:
0153 308
52 04 AC D0 0153 309 movl 4(ap),r2 ; get device string descriptor
53 7E D4 0157 310 80$: clrl -(sp) ; make space on stack for channel
53 5E D0 0159 311 movl sp,r3 ; and save its address
015C 312 $assign_s devnam=(r2),- ; assign a channel to the device
015C 313 chan=(r3)
53 8ED0 0169 314 popl r3 ; put channel number in r3
05 016C 315 90$: rsb

```



```
016D 317 .sbttl count_fcbwcb -- count the number of FCB's and WCB's
016D 318
016D 319 :---
016D 320
016D 321 This kernel mode routine counts the number of FCB's and WCB's
016D 322 associated with a particular device or volume set. This is necessary
016D 323 to allow a variable number of files to be displayed.
016D 324
016D 325 Inputs:
016D 326
016D 327 4(ap) = Channel number of the assigned device
016D 328
016D 329 Outputs:
016D 330
016D 331 fcb_count = the number of FCB's actually active
016D 332 wcb_count = the number of primary windows associated with the FCB's
016D 333 :---
016D 334
016D 335
016D 336 count_fcbwcb:
016D 337 .word ^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
016F 338
0000000C'EF D4 016F 339 clrl fcb_count ; nothing so far
00000018'EF D4 0175 340 clrl wcb_count
50 04 AC 3C 017B 341 movzwl 4(ap),r0 ; get the channel number
00000000'GF 16 017F 342 jsb g^ioc$verifychan ; verify the channel number
55 50 E9 0185 343 blbc r0,90$ ; xfer if in error
01 51 61 D0 0188 344 movl ccb$l_ucb(r1),r1 ; get the UCB address
01 40 A1 91 018B 345 cmpb ucb$b_devclass(r1),#dc$_disk ; is this a disk?
47 12 018F 346 bneg 80$ ; exit if not
0191 347 setipl #ipl$_synch ; lock the FCB's into place
0194 348 :
0194 349 :
0194 350 :
56 34 A1 D0 0194 351 movl ucb$l_vcb(r1),r6 ; get the address of the first VCB
3E 13 0198 352 beql 80$ ; error if none there
50 51 D0 019A 353 movl r1,r0 ; put UCB address if useful place
0E A6 B5 019D 354 tstw vcb$w_rvn(r6) ; is this part of a volume set?
2F 13 01A0 355 beql 50$ ; xfer if not
59 20 A6 D0 01A2 356 movl vcb$l_rvt(r6),r9 ; else get address of RVT
30 13 01A6 357 beql 80$ ; error if none
0E 0A A9 91 01A8 358 cmpb rvt$b_type(r9),#dyn$_rvt ; right type of block?
2A 12 01AC 359 bneg 80$ ; error if not
5A 0B A9 9A 01AE 360 movzbl rvt$b_nvols(r9),r10 ; get the number of volumes in the set
59 44 A9 9E 01B2 361 movab rvt$l_ucblst(r9),r9 ; get address of the UCB list
56 89 D0 01B6 362 10$: movl (r9)+,r6 ; get the next UCB
11 13 01B9 363 beql 20$ ; skip if not mounted
10 0A A6 91 01BB 364 cmpb ucb$b_type(r6),#dyn$_ucb ; really a UCB?
17 12 01BF 365 bneg 80$ ; xfer if not
50 56 D0 01C1 366 movl r6,r0 ; else move UCB address into useful place
56 34 A6 D0 01C4 367 movl ucb$l_vcb(r6),r6 ; get VCB address
0E 13 01C8 368 beql 80$ ; xfer if none
15 10 01CA 369 bsbb 100$ ; go do the actual counting
E7 5A F5 01CC 370 20$: sobgtr r10,10$ ; continue till all UCB's processed
02 11 01CF 371 brb 70$ ; go finish up
0E 10 01D1 372
373 50$: bsbb 100$ ; go count for a single volume
```

```

50 01 D0 01D3 374
05 11 01D6 375 70$: movl #1,r0 ; set success
01D8 376 brb 90$ ; go return
50 01CC 8F 3C 01D8 377
01DD 378 80$: movzwl #ss$_notfiledev,r0 ; set error status
04 01E0 379 90$: setipl #0 ; lower IPL
01E1 380 ret ; and return to caler
01E1 381
01E1 382 ; count the FCB's and WCB's
01E1 383
58 66 D0 01E1 384 100$: movl vcb$_fcbfl(r6),r8 ; get address of the first FCB
01 12 01E4 385 bneq 110$ ; xfer if initialized
05 01E6 386 rsb ; else assume not initialized
01E7 387
56 58 D1 01E7 388 110$: cmpl r8,r6 ; all FCB's processed?
3A 13 01EA 389 beql 150$ ; xfer if all processed
1A A8 B5 01EC 390 tstw fcb$_acnt(r8) ; active?
2F 13 01EF 391 beql 140$ ; xfer if inactive
2A A8 B5 01F1 392 tstw fcb$_segn(r8) ; extension FCB?
2A 12 01F4 393 bneq 140$ ; ignore if so
0000000C'EF D6 01F6 394 incl fcb_count ; else count one more
50 10 A8 9E 01FC 395 movab fcb$_wlfl(r8),r0 ; get WCB listhead
51 60 D0 0200 396 movl (r0),r1 ; get address of the first WCB
1B 13 0203 397 beql 140$ ; xfer if none
50 51 D1 0205 398 115$: cmpl r1,r0 ; more to do?
16 13 0208 399 beql 140$ ; xfer if not
06 0B A1 06 E1 020A 400 bbc #wcb$_cathedral,wcb$_access(r1),120$ ; xfer if not Cathedral
01 2C A1 D1 020F 401 cmpl wcb$_stvb(r1),#1 ; else check for primary window
06 12 0213 402 bneq 130$ ; xfer if not primary
00000018'EF D6 0215 403 120$: incl wcb_count ; else count window
51 61 D0 0218 404 130$: movl wcb$_wlfl(r1),r1 ; get address of the next window
E5 11 021E 405 brb 115$ ; go check for the end
58 68 D0 0220 406 140$: movl fcb$_fcbfl(r8),r8 ; get address of the next FCB
FFC1 31 0223 407 brw 110$ ; go check for the end
05 0226 408 150$: rsb ; return with updated counts
```



```
0227 410 .sbttl save_dev_info -- save device information
0227 411 :---
0227 412 :
0227 413 This kernel mode routine saves the FIB's corresponding to a
0227 414 specified device in a dynamic buffer for later display.
0227 415 :
0227 416 Inputs:
0227 417 :
0227 418 4(ap) = Channel number of assigned device
0227 419 :
0227 420 Outputs:
0227 421 :
0227 422 fcb_buffer is filled with FCB's for device
0227 423 fcb_length = Length of fcb_buffer filled
0227 424 :---
0227 425
0227 426 save_dev_info:
0227 427 .word *m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
0229 428
50 04 AC 3C 0229 429 movzwl 4(ap),r0 ; get channel number
00000000 GF 16 022D 430 jsb g^ioc$verifychan ; verify the channel number
55 50 E9 0233 431 blbc r0,90$ ; branch on error
01 51 61 D0 0236 432 movl ccb$l_ucb(r1),r1 ; get UCB address
40 A1 91 0239 433 cmpb ucb$b_devclass(r1),#dc$_disk ; is this a disk?
47 12 023D 434 bneg 80$ ; branch if not
023F 435 setipl #ipl$_synch ; and lock FCB's in place
0242 436 :
0242 437 : Search all UCB's in the relative volume list
0242 438 :
56 34 A1 D0 0242 439 movl ucb$l_vcb(r1),r6 ; Address of first VCB
3E 13 0246 440 beql 80$ ; error if none
50 51 D0 0248 441 movl r1,r0 ; Put UCB address in place
0E A6 B5 024B 442 tstw vcb$w_rvn(r6) ; Is this a multiple disk volume?
2F 13 024E 443 beql 50$ ; branch if not
59 20 A6 D0 0250 444 movl vcb$l_rvt(r6),r9 ; Get address of RVT
30 13 0254 445 beql 80$ ; error if none
0E 0A A9 91 0256 446 cmpb rvt$b_type(r9),#dyn$_rvt ; Is it really an RVT?
2A 12 025A 447 bneg 80$ ; Error if not
5A 0B A9 9A 025C 448 movzbl rvt$b_nvols(r9),r10 ; Get number of volumes
59 44 A9 9E 0260 449 movab rvt$l_ucblst(r9),r9 ; Address of list of UCB addresses
56 89 D0 0264 450 10$: movl (r9)+,r6 ; Get next UCB in list
11 13 0267 451 beql 20$ ; Skip if this RVN is dismounted
10 0A A6 91 0269 452 cmpb ucb$b_type(r6),#dyn$_ucb ; Is it really a UCB?
17 12 026D 453 bneg 80$ ; Error if not
50 56 D0 026F 454 movl r6,r0 ; save the UCB address
56 34 A6 D0 0272 455 movl ucb$l_vcb(r6),r6 ; Get address of VCB
0E 13 0276 456 beql 80$ ; Error if none
15 10 0278 457 bsbb save_all_fcbs ; Save all FCB's for this volume
E7 5A F5 027A 458 20$: sobgtr r10,T0$ ; Loop until we reach last volume
02 11 027D 459 brb 70$ ; Exit with success
027F 460
0E 10 027F 461 50$: bsbb save_all_fcbs ; Save all FCB's on single volume
0281 462
50 01 D0 0281 463 70$: movl #1,r0 ; success
05 11 0284 464 brb 90$
0286 465
50 01CC 8F 3C 0286 466 80$: movzwl #ss$_notfiledev,r0 ; Not a disk device
```

SHOWFILES
V04-000

- list open files on a disk device L 13
save_dev_info -- save device information 15-SEP-1984 23:46:26 VAX/VMS Macro V04-00 Page 13
4-SEP-1984 23:22:34 [CLIUTL.SRC]SHOWFILES.MAR;1 (7)

04	028B	467	90\$:	setipl	#0	
	028E	468		ret		; lower IPL ; and exit


```
028F 470 .sbtcl save_all_fcbs Save all FCB's for a volume
028F 471 :---
028F 472 :
028F 473 Save all the FCB's for this device whose access count NE 0
028F 474 :
028F 475 Inputs:
028F 476 :
028F 477 R6 = Address of VCB
028F 478 R0 = Address of UCB
028F 479 :
028F 480 Outputs:
028F 481 :
028F 482 fcb_length = Updated length of stuff stored in fcb_buffer
028F 483 :
028F 484 r0-r5, r7-r8 destroyed.
028F 485 :---
028F 486
028F 487 save_all_fcbs:
57 00000004'EF D0 028F 488 movl fcb_length,r7 ; Pickup length already stored
53 00000008'FF47 9E 0296 489 movab @fcb_buffer[r7],r3 ; Address to store FCB's
58 66 D0 029E 490 movl vcb$_fcbfl(r6),r8 ; Address of first FCB
03 12 02A1 491 bneq 10$ ; if non-zero, VCB is initialized
00E1 31 02A3 492 brw 120$ ; else return now, not initialized
02A6 493 :
02A6 494 : The first fcb entry for each disk is a dummy fcb, containing data
02A6 495 : which will later be used to construct a device string.
02A6 496 :
55 50 D0 02A6 497 10$: movl r0,r5 ; put UCB address in place
50 00AC 8F 3C 02A9 498 movzwl #fcb$_length-8,r0 ; give length
51 08 A3 DE 02AE 499 movab 8(r3),r1 ; and address of output buffer
04 A3 51 D0 02B2 500 movl r1,4(r3) ; also put this into the descriptor
54 54 DD 02B6 501 pushl r4 ; Save R4
54 01 CE 02B8 502 mnegl #1,r4 ; R4<0 ==> give node if not local
00000000'EF 16 02BB 503 jsb ioc$cvl_devnam ; then call routine to return dev name
54 51 8ED0 02C1 504 popl r4 ; Restore R4
63 51 D0 02C4 505 movl r1,(r3) ; put actual length in place
57 000000B4 8F C0 02C7 506 addl2 #fcb$_length,r7 ; skip to next fcb cell
53 00000008'FF47 9E 02CE 507 movab @fcb_buffer[r7],r3 ; and recompute address to store FCB
02D6 508 :
56 58 D1 02D6 509 20$: cmpl r8,r6 ; end of list?
03 12 02D9 510 bneq 30$ ; xfer if more to come
00A9 31 02DB 511 brw 120$ ; else i'm done with this volume
1A A8 B5 02DE 512 30$: tstw fcb$_acnt(r8) ; Anybody accessing this file?
03 12 02E1 513 bneq 40$ ; xfer if not active
0092 31 02E3 514 brw 110$ ; else go get the next one
2A A8 B5 02E6 515 40$: tstw fcb$_segn(r8) ; Is this an extension header FCB?
03 13 02E9 516 beql 45$ ; skip if not
008A 31 02EB 517 brw 110$ ; else ignore it
63 68 00B4 8F 28 02EE 518 45$: movc #fcb$_length,(r8),(r3) ; Add to accum. FCB list
57 000000B4 8F C0 02F4 519 addl #fcb$_length,r7 ; Increment length in buffer
FF5C C3 7C 02FB 520 clrq fcb$_wfl-fcb$_length(r3) ; Preset WCB list to zero
50 10 A8 9E 02FF 521 movab fcb$_wfl(r8),r0 ; Get address of WCB listhead
51 60 D0 0303 522 movl (r0),r1 ; Get address of first WCB
70 13 0306 523 beql 110$ ; branch if zero
50 51 D1 0308 524 50$: cmpl r1,r0 ; Empty WCB list?
6B 13 030B 525 beql 110$ ; skip if so
52 00000010'EF D0 030D 526 movl wcb_length,r2 ; else get current length of list
```



```
54 00000014'FF42 9E 0314 527 movab @wcb buffer[r2],r4 ; calculate actual buffer address
06 0B A1 06 E1 031C 528 60$: bbc #wcb$ cathedra,wcb$b_access(r1),70$ ; xfer if not cathedral
01 2C A1 D1 0321 529 cmpl wcb$l_stvbn(r1),#1 ; else check for a primary window
64 0C A1 12 0325 530 bneq 100$ ; xfer if not primary
04 A4 01 D0 0327 531 70$: movl wcb$l_pid(r1),(r4) ; else save internal PID
08 A4 08 A1 D0 032B 532 movl #1,4(r4) ; save the window count
OC A4 0B A1 3C 032F 533 movzwl wcb$w_size(r1),8(r4) ; save the window size
52 52 51 D0 0339 534 movzbl wcb$b_access(r1),12(r4) ; save access status
52 20 A2 D0 033C 535 movl r1,r2 ; copy address for later
04 A4 D6 0342 536 80$: movl wcb$l_link(r2),r2 ; get the address of the next segment
55 08 A2 3C 0345 537 beql 90$ ; xfer if the last
08 A4 55 C0 0349 538 incl 4(r4) ; else count one more window segment
55 FF60 C3 9E 034F 539 movzwl wcb$w_size(r2),r5 ; get the size of the window
01 65 D1 0356 540 addl2 r5,8(r4) ; total it up
FC A5 54 D0 035B 541 brb 80$ ; go get the next segment (if any)
00000018'EF 00000010'EF 10 C0 035F 542 90$: movab fcb$l_wlbl-fcb$e_length(r3),r5 ; set address of count
51 61 D0 0373 543 incl (r5) ; one more block
58 68 D0 0378 544 cmpl (r5),#1 ; is it the first?
0000000C'EF 03 D1 037B 545 bneq 95$ ; xfer if not
FF4F 31 0384 546 movl r4,-4(r5) ; else save address of first block
00000004'EF 57 D0 0387 547 95$: addl2 #16,wcb_length ; advance to the next block
57 05 038E 548 cmpl wcb_length,wcb_count ; too much?
57 05 038E 549 bgeq 120$ ; xfer if so...finish up
57 05 038E 550 100$: movl wcb$l_wlfl(r1),r1 ; else get the next in line
57 05 038E 551 110$: brb 50$ ; go check for more
57 05 038E 552 movl fcb$l_fcbfl(r8),r8 ; Skip to next FCB
57 05 038E 553 cmpl r7,fcb_count ; reached the upper limit?
57 05 038E 554 bgeq 120$ ; stop if so
57 05 038E 555 brw 20$
57 05 038E 556 120$: movl r7,fcb_length ; Store length actually in buffer
57 05 038E 557 rsb
```



```
038F 559 .sbttl display_info, Display all files from FCB list
038F 560 :---
038F 561 :
038F 562 This routine displays a list of file specifications
038F 563 given a list of FCB's
038F 564 :
038F 565 Inputs:
038F 566 :
038F 567 fcb_buffer = Buffer holding FCB's for device
038F 568 fcb_length = Number of bytes actually filled in fcb_buffer
038F 569 :
038F 570 Outputs:
038F 571 :
038F 572 None
038F 573 :---
038F 574 :
038F 575 display_info:
038F 576 .word ^m<r2,r6,r7>
0391 577 :
57 00000008'EF D0 0391 578 movl fcb_buffer,r7 ; Address of first FCB
56 00000004'EF D0 0398 579 movl fcb_length,r6 ; Length of buffer left to process
50 13 039F 580 beql 80$ ; Branch if none found
03A1 581 :
03A1 582 : Check here to see if this "fcb" is actually a dummy, containing
03A1 583 the device descriptor and device string for the next unit.
03A1 584 In that case, the FCB forward pointer, instead of being a system
03A1 585 address (and therefore negative) will contain a small positive
03A1 586 number, which is the length of the device string.
03A1 587 :
67 D5 03A1 588 5$: tstl (r7) ; check to see if system address
31 19 03A3 589 blss 10$ ; if negative, it's an FCB
57 D0 03A5 590 6$: movl r7,r2 ; use this descriptor for device name
57 000000B4 8F C0 03A8 591 addl #fcb$c_length,r7 ; point to next FCB
67 D5 03AF 592 tstl (r7) ; check to see if system address
F2 18 03B1 593 bgeq 6$ ; if GEQ, then this is another
03B3 594 ; dummy fcb, so skip the present one
03B3 595 ; otherwise build disk header message
00 DD 03B3 596 pushl #0 ; current date and time
52 DD 03B5 597 pushl r2 ; disk name descriptor
5E DD 03B7 598 pushl sp ; address of arglist for call
0000006B'EF DF 03B9 599 pushal disk ; format for arglist
00000000'GF 02 FB 03BF 600 calls #2,g^SHOW$WRITE_LINE ; print line from [cliutl.src]showio.b32
00000000'EF 08 C8 03C6 601 bisl2 #devi$m_file,devi$L_bitlis ; force a new header line
03CD 602 :
56 000000B4 8F C2 03CD 603 subl #fcb$c_length,r6 ; adjust length of FCB buffer
1B 13 03D4 604 beql 80$ ; if zero, finished
03D6 605 :
57 DD 03D6 606 10$: pushl r7 ; Address of local FCB
52 DD 03D8 607 pushl r2 ; Address of device descriptor
000003F5'EF 02 FB 03DA 608 calls #2,display_file ; Display file information
57 000000B4 8F C0 03E1 609 addl #fcb$c_length,r7 ; Skip to next FCB
56 000000B4 8F C2 03E8 610 subl #fcb$c_length,r6 ; Decrement length left to do
B0 12 03EF 611 bneq 5$
50 01 D0 03F1 612 80$: movl #1,r0
04 03F4 613 90$: ret
```



```
03F5 615 .sbtt display_file, Display information on a single file
03F5 616 :---
03F5 617 :
03F5 618 This routine is called to display a single FCB
03F5 619 :
03F5 620 Inputs:
03F5 621 :
03F5 622 4(ap) = Address of device name descriptor
03F5 623 8(ap) = Address of FCB in local memory
03F5 624 :
03F5 625 Outputs:
03F5 626 :
03F5 627 None
03F5 628 :---
03F5 629
03F5 630 display_file:
04BC 03F5 631 .word ^m<r2,r3,r4,r5,r7,r10>
03F7 632
000004ED'EF 16 03F7 633 jsb get_filename ; get the filename
03 50 E8 03FD 634 blbs r0,T0$ ; if success, continue
00E6 31 0400 635 brw 110$ ; else exit
54 10 A7 7D 0403 636 10$: movq fcb$l_wlfl(r7),r4 ; r4 = address of WCB info blocks
0407 637 ; r5 = number of WCB info blocks
0A 12 0407 638 bneq 20$ ; xfer if one there
54 00000050'EF 9E 0409 639 movab null_wcbblock,r4 ; else set as a null block
55 01 D0 0410 640 movl #1,r5 ; only one block
64 B5 0413 641 20$: tstw (r4) ; any PID obtained?
0415 642 ; (installed files may set upper word
0415 643 ; to shared WCB reference count)
1B 12 0415 644 bneq 40$ ; if so, then process normally
0417 645
00000000'EF 04 E0 0417 646 bbs #devi$u_syst, - ; check if system files wanted
03 041E 647 devi$l_bitlis,30$
00BC 31 041F 648 brw 100$ ; branch if not wanted
0422 649
0000003C'EF D4 0422 650 30$: clrl process_name ; preset process name to null
64 D4 0428 651 clrl (r4) ; zero PID
00000044'EF D4 042A 652 clrl jpi_pid ; initialize for output
3C 11 0430 653 brb 60$ ; and print it
0432 654
0432 655 ; handle normal files here
00000000'EF 05 E0 0432 656 40$: bbs #devi$u_user, - ; check to see if user files wanted
03 0439 657 devi$l_bitlis,50$
00A1 31 043A 658 brw 100$ ; branch if not
043D 659
043D 660 : Now to attempt a $GETJPI on the process which has the file open.
043D 661 : If the request fails, then the assumption is that the current
043D 662 : process did not have the right privilege (GROUP or WORLD) to
043D 663 : get information about the file's process -- and therefore should
043D 664 : not be allowed to see what files that process is accessing.
043D 665 :
50 64 D0 043D 666 50$: movl (r4),r0 ; Put the internal pid in r0
00000000'GF 16 0440 667 jsb g^exe$ipid_to_epid ; Convert to extended pid
00000044'EF 50 D0 0446 668 movl r0,jpi_pid ; Store the extended pid, we need it again
044D 669 $getjpi_s_pida=r=jpi_pid,- ; Get process name
044D 670 itmlst=jpi_procnam
03 50 E8 0468 671 blbs r0,60$ ; xfer if allowed
```



```
0070 31 046B 672 brw 100$ ; else skip this one
      046E 673 ; make output line for normal files
53 00000043'EF 52 DD 046E 674 60$: pushl r2 ; set addr of file name descriptor
00000000'EF 06 9E 0470 675 movab faofil,r3 ; set control string address
      1F 06 E1 0477 676 bbc #devi$wind, - ; xfer if no window information
53 00000020'EF 9E 047E 677 devisl_bitlis,75$
      08 A4 DD 047F 678 movab faowin,r3 ; else set new control string
      00000062'EF DD 0486 679 pushl 8(r4) ; total size of windows
07 0C A4 06 9F 0489 680 pushab nocath ; assume no cathedral windows
6E 00000059'EF 9E 048F 681 bbc #wcb$ cathedral,12(r4),70$ ; xfer if i got it right
      04 A4 DD 049B 682 movab cath,(sp) ; else set cathedral indicator
      00000044'EF DD 049E 683 70$: pushl 4(r4) ; total number of windows
      0000003C'EF 9F 04A4 684 75$: pushl jpi_pid ; extended PID of accessor
      5A 5E D0 04AA 685 pushab process_name ; address of the owner process name
      04AD 686 movl sp,r10 ; save the address of this arglist
00000000'EF 03 E5 04AD 687 ; output the line
      1E 04B4 688 bbcc #devi$wind, - ; branch if header already written
      00 DD 04B5 689 devisl_bitlis,85$ ; (notice re-use of devi$wind)
00000000'EF 06 DF 04B7 690 pushl #0 ; indicate there is no arglist for heading
      07 06 E1 04BD 691 pushal hedfil ; else output the header
6E 0000009A'EF 02 DE 04C4 692 bbc #devi$wind, - ; get correct header string
00000000'GF 5A DD 04C5 693 devisl_bitlis,80$
      53 DD 04D3 694 movl hedwin,(sp)
      02 FB 04CC 695 80$: calls #2,g^show$write_line ; print from [cliutl.src]showio.b32
      54 DD 04D5 696 85$: pushl r10 ; the arglist
00000000'GF 02 FB 04D7 697 pushl r3 ; the format
      54 10 C0 04DE 698 calls #2,g^show$write_line ; Output the expanded name string
      02 55 F5 04E1 699 100$: addl2 #16,r4 ; point to the next WCB block
      03 11 04E4 700 sobgtr r5,105$ ; skip if more
      FF2A 31 04E6 701 brb 110$ ; else go finish up
      50 01 D0 04E9 702 105$: brw 20$ ; go analyze this info block
      04 04EC 703 movl #1,r0
      04 04EC 704 110$: ret
      04 04EC 705
```



```
04ED 707 .sbtll get_filename, Get file name from FID
04ED 708 :---
04ED 709
04ED 710 This routine is called for each FCB, to extract the name of
04ED 711 the file.
04ED 712
04ED 713 Inputs:
04ED 714
04ED 715 4(ap) = Address of device name descriptor
04ED 716 8(ap) = Address of FCB in local memory
04ED 717
04ED 718 Outputs:
04ED 719
04ED 720 R0 = success/failure in finding file
04ED 721 :---
04ED 722
04ED 723
04ED 724 get_filename:
04ED 725     popl     r5                ; save return address
04ED 726     movl     8(ap),r7          ; get address of local FCB
04ED 727     subl     #4,sp            ; Allocate space for error return
04ED 728     movl     sp,r3            ; r3 = Address of error
04ED 729     subl     #nam$c_maxrss,sp ; Allocate filespec buffer
04ED 730     pushl    sp              ; and descriptor of buffer
04ED 731     movzwl   #nam$c_maxrss,-(sp)
04ED 732     movl     sp,r2            ; r2 = Address of filespec descriptor
04ED 733     pushl    r3              ; address for secondary error return
04ED 734     pushl    #0              ; don't know directory ID
04ED 735     pushl    r2              ; return length of filespec
04ED 736     pushl    r2              ; filespec descriptor
04ED 737     pushab   fcb$w_fid(r7)    ; Address of FID
04ED 738     pushl    4(ap)           ; Address of device name descriptor
04ED 739     calls    #6,g^lib$fid_to_name ; Convert DID/FID to filespec
04ED 740     blbc     r0,5$            ; branch if in error
04ED 741     movl     (r3),r0          ; Check for secondary error
04ED 742     blbc     r0,5$            ; branch if secondary error
04ED 743     pushr    #^m<r0,r1>      ; save LOCC regs
04ED 744     movq     (r2),r0          ; get file name descriptor
04ED 745     locc     #^a/;/,r0,(r1)  ; get the end of the device name
04ED 746     decl     r0              ; one less byte
04ED 747     incl     r1              ; skip over colon
04ED 748     movq     r0,(r2)          ; save new descriptor
04ED 749     popr     #^m<r0,r1>      ; restore LOCC regs
04ED 750     brb      20$             ; go finish up
04ED 751 5$:      cmpw     r0,#ss$_nosuchfile ; check for "no such file"
04ED 752     bneq     10$            ; if not, process normally
04ED 753     clrl     (r2)            ; if no such file, leave name blank
04ED 754     movzbl   #1,r0          ; show success
04ED 755     brb      20$            ; and exit
04ED 756 10$:     movl     #nam$c_maxrss,(r2) ; Restore length of buffer
04ED 757     $getmsg_s msgid=r0,msglen=(r2),bufadr=(r2),flags=#1 ; and get message
04ED 758 20$:     pushl    r5                ; restore return address
04ED 759     rsb
04ED 760
04ED 761 .end
```

57 08 AC D0 04F0 726
5E 04 C2 04F4 727
53 5E D0 04F7 728
5E 00000000'8F C2 04FA 729
7E 0000'8F 3C 0501 730
52 5E DD 0503 731
53 DD 0508 732
00 DD 050B 733
52 DD 050D 734
52 DD 050F 735
24 A7 9F 0511 736
04 AC DD 0513 737
00000000'GF 06 FB 0516 738
1A 50 E9 0519 739
50 63 D0 0520 740
14 50 E9 0523 741
03 BB 0526 742
50 62 7D 0529 743
61 50 3A 3A 052B 744
50 D7 052E 745
51 D6 0532 746
62 50 7D 0534 747
03 BA 0536 748
26 11 0539 749
0910 8F 50 B1 053B 750
07 12 053D 751
62 D4 0542 752
50 01 9A 0544 753
18 11 0546 754
62 00000000'8F D0 0549 755
55 DD 054B 756
05 0563 757
0565 758
0566 759
0566 760
0566 761

SHOWFILES
Symbol table

- list open files on a disk device

F 14

15-SEP-1984 23:46:26 VAX/VMS Macro V04-00
4-SEP-1984 23:22:34 [CLIUTL.SRC]SHOWFILES.MAR;1

Page 20
(11)

\$\$T1	=	00000001		
BIT...	=	00000007		
BUFLIM		00000010	R	03
CATH		00000059	R	03
CCBSL_UCB	=	00000000		
CODELIM		00000018	R	03
COUNT_FCBWCB		0000016D	R	04
DCS_DISK	=	00000001		
DEF_DISK		0000001C	R	02
DEVISL_BITLIS		00000000	R	02
DEVISM_FILE	=	00000008		
DEVISM_SYST	=	00000010		
DEVISM_USER	=	00000020		
DEVISM_WIND	=	00000040		
DEVISV_FILE	=	00000003		
DEVISV_SYST	=	00000004		
DEVISV_USER	=	00000005		
DEVISV_WIND	=	00000006		
DISK		0000006B	R	03
DISPLAY_FILE		000003F5	R	04
DISPLAY_INFO		0000038F	R	04
DYNIS_RVT	=	0000000E		
DYNIS_UCB	=	00000010		
EXESIPID_TO_EPID		*****	X	04
FAOFIL		00000043	R	03
FAOWIN		00000020	R	03
FCBSC_LENGTH	=	000000B4		
FCBSL_FCBFL	=	00000000		
FCBSL_WLBL	=	00000014		
FCBSL_WLFL	=	00000010		
FCBSW_ACNT	=	0000001A		
FCBSW_FID	=	00000024		
FCBSW_SEGN	=	0000002A		
FCB_BUFFER		00000008	R	02
FCB_COUNT		0000000C	R	02
FCB_LENGTH		00000004	R	02
GET_DEVICE		00000153	R	04
GET_FILENAME		000004ED	R	04
HEDFIL		000000D8	R	03
HEDWIN		0000009A	R	03
IOCSVT_DEVNAM		*****	X	04
IOCSVTIFYCHAN		*****	X	04
IPLS_SYNCH	=	00000008		
JPI\$-PRCNAM	=	0000031C		
JPI_PID		00000044	R	02
JPI-PRCNAM		00000000	R	03
LIB\$FID_TO_NAME		*****	X	04
LIB\$GET-VM		*****	X	04
NAMSC_MAXRSS		*****	X	04
NEWBUFLIM		00000048	R	02
NOCATH		00000062	R	03
NULL_WCBLOCK		00000050	R	02
PR\$ IPL	=	00000012		
PROCESS_NAME		0000003C	R	02
PROCNAMBUF		0000002D	R	02
RVT\$B_NVOLS	=	0000000B		
RVT\$B_TYPE	=	0000000A		

RVT\$L_UCBLST	=	00000044		
SAVE_ALL_FCBS		0000028F	R	04
SAVE_DEV_INFO		00000227	R	04
SHOWFILES		00000000	RG	04
SHOWWRITE_LINE		*****	X	04
SIZ...	=	00000001		
SS\$_NOSUCHFILE	=	00000910		
SS\$_NOTFILEDEV	=	000001CC		
SYSS\$ASSIGN		*****	GX	04
SYSS\$CMKRN		*****	X	04
SYSS\$GETJPI		*****	GX	04
SYSS\$GETMSG		*****	GX	04
SYSS\$LKWSET		*****	GX	04
SYSS\$ULWSET		*****	GX	04
UCBSB_DEVCLASS	=	00000040		
UCBSB_TYPE	=	0000000A		
UCBSL_VCB	=	00000034		
VCBSL_FCBFL	=	00000000		
VCBSL_RVT	=	00000020		
VCBSW_RVN	=	0000000E		
WCB\$B_ACCESS	=	0000000B		
WCB\$B_LINK	=	00000020		
WCB\$B_PID	=	0000000C		
WCB\$B_STVBN	=	0000002C		
WCB\$B_WLFL	=	00000000		
WCB\$V_CATHEDRAL	=	00000006		
WCB\$W_SIZE	=	00000008		
WCB_BUFFER		00000014	R	02
WCB_COUNT		00000018	R	02
WCB_LENGTH		00000010	R	02

SHOW\$FILES
Psect synopsis

- list open files on a disk device G 14

15-SEP-1984 23:46:26 VAX/VMS Macro V04-00 Page 21
4-SEP-1984 23:22:34 [CLIUTL.SRC]SHOWFILES.MAR;1 (11)

+-----+
! 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	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SHOW\$RWDATA	00000060 (96.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC BYTE
SHOW\$RODATA	00000103 (259.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC BYTE
SHOW\$CODE	00000566 (1382.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	11	00:00:00.07	00:00:01.32
Command processing	77	00:00:00.92	00:00:05.18
Pass 1	480	00:00:18.29	00:01:06.10
Symbol table sort	0	00:00:03.08	00:00:08.31
Pass 2	160	00:00:03.62	00:00:13.00
Symbol table output	10	00:00:00.11	00:00:00.26
Psect synopsis output	3	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	743	00:00:26.13	00:01:34.21

The working set limit was 1500 pages.
107457 bytes (210 pages) of virtual memory were used to buffer the intermediate code.
There were 110 pages of symbol table space allocated to hold 1944 non-local and 62 local symbols.
761 source lines were read in Pass 1, producing 23 object records in Pass 2.
32 pages of virtual memory were used to define 31 macros.

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

Macro library name	Macros defined
-\$255\$DUA28:[CLIUTL.OBJ]CLIUTL.MLB;1	0
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	10
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	18
TOTALS (all libraries)	28

2096 GETS were required to define 28 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:SHOWFILES/OBJ=OBJ\$:SHOWFILES MSRC\$:SHOWFILES/UPDATE=(ENH\$:SHOWFILES)+EXECMLS\$/LIB+LIB\$:CLIUTL/LIB

0056 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

