

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57

```

0001 0 MODULE DIRECTORY (
0002 0
0003 0     LANGUAGE (BLISS32),
0004 0     IDENT = 'V04-000',
0005 0     MAIN = DIR$MAIN
0006 0 ) =
0007 1 BEGIN
0008 1
0009 1 .....
0010 1 *
0011 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0012 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0013 1 * ALL RIGHTS RESERVED.
0014 1 *
0015 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0016 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0017 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0018 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0019 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0020 1 * TRANSFERRED.
0021 1 *
0022 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0023 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0024 1 * CORPORATION.
0025 1 *
0026 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0027 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0028 1 *
0029 1 *
0030 1 .....
0031 1
0032 1 **
0033 1
0034 1 FACILITY:    DIRECTORY
0035 1
0036 1 ABSTRACT:
0037 1
0038 1     This module contains the main processing routine for the directory
0039 1     command.  It also contains various error reporting routines.
0040 1
0041 1 ENVIRONMENT:
0042 1
0043 1     VAX/VMS operating system, unprivileged user mode utilities.
0044 1
0045 1 --
0046 1
0047 1 AUTHOR:      L. Mark Pilant      CREATION DATE: 3-Mar-1983
0048 1
0049 1 MODIFIED BY:
0050 1
0051 1     V03-020 LMP0296      L. Mark Pilant,      6-Aug-1984 12:54
0052 1     Note the hack to get /FULL to work with the magtape ACP.
0053 1
0054 1     V03-019 LMP0280      L. Mark Pilant,      19-Jul-1984 12:54
0055 1     Give the correct text on the DIR$_SYNTAX error message.
0056 1
0057 1     V03-018 LMP0276      L. Mark Pilant,      11-Jul-1984 11:51

```

58	0058	1	Some modifications:		
59	0059	1	1) Fix a bug in LMP0263 that caused extra headings to		
60	0060	1	come out.		
61	0061	1	2) Fix the handling of /OUTPUT and /NOOUTPUT.		
62	0062	1			
63	0063	1	V03-017 LMP0263 L. Mark Pilant, 26-Jun-1984 12:58		
64	0064	1	Clear out the version count and saved directory name for		
65	0065	1	each input spec.		
66	0066	1			
67	0067	1	V03-016 JEJ0017 J E Johnson 16-Apr-1984		
68	0068	1	Fix bug caused by V03-014 edit.		
69	0069	1			
70	0070	1	V03-018 BLS0300 Benn Schreiber 11-APR-1984		
71	0071	1	Do not link with SECURESHR to get the format_acl service.		
72	0072	1	Rather, only load it if /acl or /full.		
73	0073	1			
74	0074	1	V03-014 JEJ0017 J E Johnson 27-Mar-1984		
75	0075	1	Clean up the network \$SEARCH XAB fill support to use the		
76	0076	1	NOP flag SRCHXABS.		
77	0077	1			
78	0078	1	V03-013 LMP0211 L. Mark Pilant, 10-Mar-1984 12:44		
79	0079	1	Fix some minor logic problems that occurred when the display		
80	0080	1	logic was changed.		
81	0081	1			
82	0082	1	V03-012 BLS0265 Benn Schreiber 25-Jan-1984		
83	0083	1	Use enhanced lib\$file_scan features for stickyness		
84	0084	1			
85	0085	1	V03-011 LMP0182 L. Mark Pilant, 11-Jan-1984 12:43		
86	0086	1	Note the use of the /SELECT qualifier with an appropriate flag.		
87	0087	1			
88	0088	1	V03-010 LMP0180 L. Mark Pilant, 12-Dec-1983 9:42		
89	0089	1	Correct a bug in the formatting uncovered by the fix in		
90	0090	1	LMP0176.		
91	0091	1			
92	0092	1	V03-009 LMP0176 L. Mark Pilant, 6-Dec-1983 8:54		
93	0093	1	Correct an incorrect piece of logic used to determine the		
94	0094	1	number of columns able to be printed in a display.		
95	0095	1			
96	0096	1	V03-008 LMP0171 L. Mark Pilant, 23-Nov-1983 10:39		
97	0097	1	Correct a bug that caused the size selection item to be		
98	0098	1	dropped on the floor.		
99	0099	1			
100	0100	1	V03-007 LMP0157 L. Mark Pilant, 27-Sep-1983 10:45		
101	0101	1	Add support for a unique message file.		
102	0102	1			
103	0103	1	V03-006 LMP0132 L. Mark Pilant, 3-Aug-1983 10:19		
104	0104	1	Correct the qualifier keyword COLUMN to be COLUMNS to match		
105	0105	1	the documentation.		
106	0106	1			
107	0107	1	V03-005 LMP0119 L. Mark Pilant, 15-Jun-1983 9:29		
108	0108	1	Add support for identifiers.		
109	0109	1			
110	0110	1	V03-004 LMP0108 L. Mark Pilant, 28-Apr-1983 10:49		
111	0111	1	Issue a DIRECTORY message if no files are found, not an RMS		
112	0112	1	message. Also, add support for RMS journaling.		
113	0113	1			
114	0114	1	V03-003 LMP0100 L. Mark Pilant, 14-Apr-1983 11:49		

```
.. 115      0115  1  : Misc fixups.
.. 116      0116  1  :
.. 117      0117  1  :
.. 118      0118  1  : V03-002 LMP0096      L. Mark Pilant,      29-Mar-1983  10:01
.. 119      0119  1  :     Correctly handle locked files.
.. 120      0120  1  :
.. 121      0121  1  : V03-001 LMP0092      L. Mark Pilant,      25-Mar-1983  12:24
.. 122      0122  1  :     Include the FHC XAB when /SIZE is specified. Also fix
.. 123      0123  1  :     the handling of the final error status.
.. 124      0124  1  : ..
.. 125      0125  1  :
.. 126      0126  1  : LIBRARY 'SYSSLIBRARY:LIB';
.. 127      0127  1  : REQUIRE 'SRCS:DIRECTDEF';
```

HACKS WORTH NOTING...

```

: 129 0529 1
: 130 0530 1
: 131 0531 1
: 132 0532 1
: 133 0533 1
: 134 0534 1
: 135 0535 1
: 136 0536 1
: 137 0537 1
: 138 0538 1
: 139 0539 1
: 140 0540 1
: 141 0541 1
: 142 0542 1
: 143 0543 1
: 144 0544 1
: 145 0545 1
: 146 0546 1
: 147 0547 1
: 148 0548 1
: 149 0549 1
: 150 0550 1
: 151 0551 1
: 152 0552 1
: 153 0553 1
: 154 0554 1
: 155 0555 1
: 156 0556 1
: 157 0557 1
: 158 0558 1
: 159 0559 1
: 160 0560 1
: 161 0561 1
: 162 0562 1
: 163 0563 1

```

There are several hacks used by DIRECTORY to improve performance and to compensate for bugs elsewhere in the system.

The first is mechanism that allows the file information requested in the RMS XAB blocks to be filled in while performing a \$SEARCH over the network. If the NAM block attached to the FAB doing the \$SEARCH has the NOP bit NAMSV_SRCHXABS set, then any XABs attached to the FAB will have the requested information filled in if it is available.

The next is used by LIB\$FILE_SCAN to improve performance. Doing a \$SEARCH operation over the network involves a considerable amount of startup overhead (to make the connection). Therefore, LIB\$FILE_SCAN will only do the network \$SEARCH operation if there are wildcard characters present (as determined by the previous \$PARSE). This means that if there are XABs to be filled, and no wildcards are present in the filespec, it is necessary to issue an explicit \$SEARCH (outside of LIB\$FILE_SCAN).

Another hack used here is to not explicitly link with SECURESHR, which contains the format_acl service. Rather, we auto-load it using lib\$find_image_symbol only if /acl or /full is present. This gives a reduction in activation time in the case we don't need to format any acs.

The last hack is to make /FULL work with the magtape ACP. There is a bug in the magtape ACP encountered when doing wildcarding and accessing by file name to the same tape drive. The access by name causes the magtape ACP to loose the wildcard context, resulting in an infinite loop. This is corrected in DIRECTORY by accessing the file by "file-ID" even when /FULL is specified, if the device is a sequential device.

```

: 165 0564 1 FORWARD ROUTINE
: 166 0565 1 DIR$MAIN, ! Main processing routine
: 167 0566 1 DIR$GET_FILE, ! Get a file spec to process
: 168 0567 1 DIR$INPOT_ERROR, ! Signal file scanning error
: 169 0568 1 DIR$FILE_ERROR, ! Signal file error
: 170 0569 1 DIR$OUTPUT; ! General output routine
: 171 0570 1
: 172 0571 1 OWN
: 173 0572 1 FORMAT_ACL_ADDR, ! Address of real SYSS$FORMAT_ACL
: 174 0573 1 OUTPUT_FAB : $FAB_DECL, ! Output file RMS structures
: 175 0574 1
: 176 0575 1 ! OUTPUT_RAB is in DIRECTDEF.REQ because it is referenced by the SIGNAL macro
: 177 0576 1 ! to flush out the RMS buffers when an error occurs.
: 178 0577 1
: 179 0578 1 OUTPUT_NAM : $NAM_DECL,
: 180 0579 1 OUT_EXP_NAM : $BLOCK [NAMSC_MAXRSS],
: 181 0580 1 OUT_RES_NAM : $BLOCK [NAMSC_MAXRSS];
: 182 0581 1
: 183 0582 1 EXTERNAL ROUTINE
: 184 0583 1 CLISGET_VALUE : ADDRESSING_MODE (GENERAL), ! Get a qualifier value
: 185 0584 1 CLISPRESENT : ADDRESSING_MODE (GENERAL), ! See if qualifier present
: 186 0585 1 LIB$FILE_SCAN : ADDRESSING_MODE (GENERAL), ! Search wildcard file spec
: 187 0586 1 LIB$FIND_IMAGE_SYMBOL : ADDRESSING_MODE (GENERAL), ! Image activate
: 188 0587 1
: 189 0588 1 ! Following are the common qualifier scanning routines
: 190 0589 1
: 191 0590 1 LIB$QUAL_FILE_PARSE : ADDRESSING_MODE (GENERAL); ! Set up select

```

```

193 0591 1 ROUTINE DIR$MAIN =
194 0592 1
195 0593 1 !++
196 0594 1
197 0595 1 FUNCTIONAL DESCRIPTION:
198 0596 1
199 0597 1 This routine is the main processing routine for the DIRECTORY command.
200 0598 1 It parses the qualifiers in the command line to determine what
201 0599 1 information is to be displayed for the selected file or files.
202 0600 1
203 0601 1 CALLING SEQUENCE:
204 0602 1
205 0603 1 DIR$MAIN ()
206 0604 1
207 0605 1 INPUT PARAMETERS:
208 0606 1 none
209 0607 1
210 0608 1 IMPLICIT INPUTS:
211 0609 1 none
212 0610 1
213 0611 1 OUTPUT PARAMETERS:
214 0612 1 none
215 0613 1
216 0614 1 IMPLICIT OUTPUTS:
217 0615 1 none
218 0616 1
219 0617 1 ROUTINE VALUE:
220 0618 1 The worst error encountered or $$$_NORMAL.
221 0619 1
222 0620 1 SIDE EFFECTS:
223 0621 1 none
224 0622 1
225 0623 1 !--
226 0624 1
227 0625 2 BEGIN
228 0626 2
229 0627 2 LOCAL
230 0628 2 STATUS, ! Local routine exit status
231 0629 2 CLI_STATUS, ! CLI parse status
232 0630 2 SCAN_CONTEXT, ! filescan context
233 0631 2 INPUT_FAB : $FAB_DECL, ! Input file RMS structures
234 0632 2 INPUT_NAM : $NAM_DECL,
235 0633 2 INP_EXP_NAM : $BLOCK [NAM$C_MAXRSS],
236 0634 2 INP_RES_NAM : $BLOCK [NAM$C_MAXRSS],
237 0635 2 FILE_DESC : $BLOCK [DSC$C_S_BLN], ! File name descr
238 0636 2 VALUE_DESC : $BLOCK [DSC$C_S_BLN], ! Qualifier value
239 0637 2 GETDVI_ARGS : VECTOR [?], ! GETDVI argument list
240 0638 2 INDEV_CLASS : ! Input device class
241 0639 2 INDEV_BUFSIZ, ! Input device buffer size
242 0640 2 XAB_PTR : REF $BLOCK; ! Pointer to current XAB
243 0641 2
244 0642 2 EXTERNAL LITERAL
245 0643 2 CLIS_DEFAULTED, ! Value present by default
246 0644 2 CLIS_NEGATED; ! Qualifier negated
247 0645 2
248 0646 2 EXTERNAL ROUTINE
249 0647 2 DIR$GET_INFO, ! Get information about a file

```



```

250 0648 2 DIR$TOTAL, ! Type out per directory totals
251 0649 2 DIR$GRAND TOTAL, ! Type out grand total info
252 0650 2 LIB$CVT_DTB : ADDRESSING_MODE (GENERAL); ! Convert string to value
253 0651 2 LIB$GET_VM : ADDRESSING_MODE (GENERAL); ! Allocate dynamic memory
254 0652 2
255 0653 2 ! DIRECTORY error messages
256 0654 2
257 0655 2 EXTERNAL LITERAL
258 0656 2 DIR$_NOFILES;
259 0657 2
260 0658 2 ! Initialize all variables
261 0659 2
262 0660 2 SCAN_CONTEXT = 0;
263 0661 2 QUAL_FLAGS = 0;
264 0662 2 WORST_ERROR = $$$_NORMAL;
265 0663 2 CHANNEL = 0;
266 0664 2 CH$FILL (0, NAM$C_DVI, DEVICE_NAME);
267 0665 2 COLUMN COUNT = COLUMN_INDEX = COLUMN_WIDTH = 0;
268 0666 2 VERSION COUNT = VERSION_INDEX = 0;
269 0667 2 PREV DIR LEN = PREV FILE_LEN = 0;
270 0668 2 TOTAL_USED = TOTAL_ALLOC = TOTAL_FILES = 0;
271 0669 2 GRAND_USED = GRAND_ALLOC = GRAND_FILES = GRAND_DIRS = 0;
272 0670 2 COLUMN_WIDTH = 0;
273 0671 2 INDEV_CLASS = INDEV_BUFSIZ = 0;
274 0672 2 FIRST_XAB = XAB_PTR = 0;
275 0673 2 CH$FILL (0, DSC$C_S_BLN, VALUE_DESC);
276 0674 2 VALUE_DESC[DSC$C_CLASS] = DSC$R_CLASS D;
277 0675 2 CH$MOVE (DSC$C_S_BLN, VALUE_DESC, FILE_DESC);
278 0676 2 CH$MOVE (DSC$C_S_BLN, VALUE_DESC, LINE_DESC);
279 0677 2 LINE_DESC[DSC$A_POINTER] = LINE_BUFFER;
280 0678 2
281 0679 2 ! Get the block of memory needed to hold the display information.
282 0680 2
283 0681 2 STATUS = LIB$GET_VM (%REF (DIR_C_LENGTH), DISPLAY_BLOCK);
284 0682 2 IF NOT .STATUS
285 0683 2 THEN
286 0684 2 BEGIN
287 0685 2 SIGNAL (.STATUS);
288 0686 2 RETURN .WORST_ERROR;
289 0687 2 END;
290 0688 2
291 0689 2 ! Initialize all RMS data structures.
292 0690 2
293 P 0691 2 $FAB_INIT (FAB = INPUT_FAB, ! Init input structures
294 P 0692 2 DNA = UPLIT ('*.*;.*'),
295 P 0693 2 DNS = %CHARCOUNT ('*.*;.*'),
296 0694 2 NAM = INPUT_NAM);
297 P 0695 2 $NAM_INIT (NAM = INPUT_NAM,
298 P 0696 2 ESA = INP_EXP_NAM,
299 P 0697 2 ESS = NAM$C_MAXRSS,
300 P 0698 2 RSA = INP_RES_NAM,
301 0699 2 RSS = NAM$C_MAXRSS);
302 0700 2
303 P 0701 2 $FAB_INIT (FAB = OUTPUT_FAB, ! Init output structures
304 P 0702 2 DNA = UPLIT ('DIRECTORY.LIS'),
305 P 0703 2 DNS = %CHARCOUNT ('DIRECTORY.LIS'),
306 P 0704 2 FAC = PUT,

```

```
307 P 0705 FOP = SQO,  
308 P 0706 NAM = OUTPUT_NAM,  
309 0707 RAT = CR);  
310 P 0708 $RAB_INIT (RAB = OUTPUT_RAB,  
311 0709 FAB = OUTPUT_FAB);  
312 P 0710 $NAM_INIT (NAM = OUTPUT_NAM,  
313 P 0711 ESA = OUT_EXP_NAM,  
314 P 0712 ESS = NAM$C_MAXRSS,  
315 P 0713 RSA = OUT_RES_NAM,  
316 0714 RSS = NAM$C_MAXRSS);  
317 0715  
318 0716 ! Parse the various command qualifiers that may have been given on the  
319 0717 ! command line.  
320 0718  
321 0719 ! First check for any of the common qualifiers to determine what XABs  
322 0720 ! may be needed.  
323 0721  
324 0722 IF CL$PRESENT ($DESCRIPTOR ('BEFORE'))  
325 0723 OR CL$PRESENT ($DESCRIPTOR ('SINCE'))  
326 0724 THEN  
327 0725 BEGIN  
328 0726 QUAL_FLAGS[DIR_V_NEED_DAT] = 1; ! DAT XAB required  
329 0727 QUAL_FLAGS[DIR_V_COMM_QUAL] = 1;  
330 0728 END;  
331 0729  
332 0730 IF CL$PRESENT ($DESCRIPTOR ('BY_OWNER'))  
333 0731 THEN  
334 0732 BEGIN  
335 0733 QUAL_FLAGS[DIR_V_NEED_PRO] = 1; ! PRO XAB required  
336 0734 QUAL_FLAGS[DIR_V_COMM_QUAL] = 1;  
337 0735 END;  
338 0736  
339 0737 ! Now check for all the display tailoring qualifiers  
340 0738  
341 0739 QUAL_FLAGS[DIR_V_QUAL_ACL] = CL$PRESENT ($DESCRIPTOR ('ACL'));  
342 0740 QUAL_FLAGS[DIR_V_QUAL_BRIE] = CL$PRESENT ($DESCRIPTOR ('BRIEF'));  
343 0741 IF (CLI_STATUS = QUAL_FLAGS[DIR_V_QUAL_COLU] = CL$PRESENT ($DESCRIPTOR ('COLUMNS')))  
344 0742 THEN  
345 0743 BEGIN  
346 0744 CL$GET_VALUE ($DESCRIPTOR ('COLUMNS'), VALUE_DESC);  
347 0745 STATUS = LIB$CVT_DTB (.VALUE_DESC[DSC$W_LENGTH],  
348 0746 .VALUE_DESC[DSC$A_POINTER],  
349 0747 COLUMN_COUNT);  
350 0748 IF NOT .STATUS OR .COLUMN_COUNT LSS 0  
351 0749 THEN  
352 0750 BEGIN  
353 0751 SIGNAL (DIR$SYNTAX, 1, VALUE_DESC);  
354 0752 RETURN .WORST_ERROR;  
355 0753 END;  
356 0754 IF .COLUMN_COUNT EQL 0 THEN COLUMN_COUNT = 1;  
357 0755 IF .CLI_STATUS EQL CL$DEFAULTED THEN QUAL_FLAGS[DIR_V_COLU_DEF] = 1;  
358 0756 END;  
359 0757 IF (QUAL_FLAGS[DIR_V_QUAL_DATE] = CL$PRESENT ($DESCRIPTOR ('DATE')))  
360 0758 THEN  
361 0759 BEGIN  
362 0760 QUAL_FLAGS[DIR_V_NEED_DAT] = 1; ! DAT XAB required  
363 0761 IF (CL$PRESENT ($DESCRIPTOR ('DATE.ALL'))
```

```

364 0762 3 THEN
365 0763 4 BEGIN
366 0764 4 QUAL_FLAGS[DIR_V_DATE_CRE] = 1;
367 0765 4 QUAL_FLAGS[DIR_V_DATE_EXP] = 1;
368 0766 4 QUAL_FLAGS[DIR_V_DATE_MOD] = 1;
369 0767 4 QUAL_FLAGS[DIR_V_DATE_BAK] = 1;
370 0768 4 COLUMN_WIDTH = .COLUMN_WIDTH + 19 * 4;
371 0769 4 END
372 0770 3 ELSE
373 0771 4 BEGIN
374 0772 4 IF CLISPRESNT ($DESCRIPTOR ('DATE.CREATED'))
375 0773 4 THEN
376 0774 5 BEGIN
377 0775 5 QUAL_FLAGS[DIR_V_DATE_CRE] = 1;
378 0776 5 COLUMN_WIDTH = .COLUMN_WIDTH + 19;
379 0777 4 END;
380 0778 4 IF CLISPRESNT ($DESCRIPTOR ('DATE.EXPIRED'))
381 0779 4 THEN
382 0780 5 BEGIN
383 0781 5 QUAL_FLAGS[DIR_V_DATE_EXP] = 1;
384 0782 5 COLUMN_WIDTH = .COLUMN_WIDTH + 19;
385 0783 4 END;
386 0784 4 IF CLISPRESNT ($DESCRIPTOR ('DATE.MODIFIED'))
387 0785 4 THEN
388 0786 5 BEGIN
389 0787 5 QUAL_FLAGS[DIR_V_DATE_MOD] = 1;
390 0788 5 COLUMN_WIDTH = .COLUMN_WIDTH + 19;
391 0789 4 END;
392 0790 4 IF CLISPRESNT ($DESCRIPTOR ('DATE.BACKUP'))
393 0791 4 THEN
394 0792 5 BEGIN
395 0793 5 QUAL_FLAGS[DIR_V_DATE_BAK] = 1;
396 0794 5 COLUMN_WIDTH = .COLUMN_WIDTH + 19;
397 0795 4 END;
398 0796 3 END;
399 0797 2 END;
400 0798 3 IF (QUAL_FLAGS[DIR_V_QUAL_FID] = CLISPRESNT ($DESCRIPTOR ('FILE_ID')))
401 0799 2 THEN COLUMN_WIDTH = .COLUMN_WIDTH + 21;
402 0800 3 IF (QUAL_FLAGS[DIR_V_QUAL_FOLL] = CLISPRESNT ($DESCRIPTOR ('FULL')))
403 0801 2 THEN
404 0802 3 BEGIN
405 0803 3 QUAL_FLAGS[DIR_V_NEED_FHC] = QUAL_FLAGS[DIR_V_NEED_DAT] = 1;
406 0804 3 QUAL_FLAGS[DIR_V_NEED_PRO] = QUAL_FLAGS[DIR_V_NEED_SUM] = 1;
407 0805 3 QUAL_FLAGS[DIR_V_NEED_JNL] = 1;
408 0806 2 END;
409 0807 2 QUAL_FLAGS[DIR_V_QUAL_GRAN] = CLISPRESNT ($DESCRIPTOR ('GRAND TOTAL'));
410 0808 2 QUAL_FLAGS[DIR_V_QUAL_HEAD] = CLISPRESNT ($DESCRIPTOR ('HEADING'));
411 0809 2
412 0810 2 ! /PRINTER is checked out of sequence because it may affect how /OUTPUT is
413 0811 2 ! handled.
414 0812 2
415 0813 2 IF (QUAL_FLAGS[DIR_V_QUAL_PRIN] = CLISPRESNT ($DESCRIPTOR ('PRINTER')))
416 0814 2 THEN
417 0815 3 BEGIN
418 0816 3 OUTPUT_FAB[FAB$V_SPL] = 1; ! Spool file when closed.
419 0817 3 OUTPUT_FAB[FAB$V_DLT] = 1; ! Delete file after printing
420 0818 2 END;

```

```

421 0819 3 IF (CLI_STATUS = QUAL_FLAGS[DIR_V_QUAL_OUTP] = CLISPRESNT ($DESCRIPTOR ('OUTPUT')))
422 0820 THEN
423 0821 BEGIN
424 0822 CLISGET VALUE ($DESCRIPTOR ('OUTPUT'), FILE_DESC);
425 0823 OUTPUT_FAB[FAB$FNA] = FILE_DESC[DSC$A_POINTER];
426 0824 IF (OUTPUT_FAB[FAB$B_FNS] = FILE_DESC[DSC$W_LENGTH]) EQL 0
427 0825 AND NOT .QUAL_FLAGS[DIR_V_QUAL_PRIN]
428 0826 THEN
429 0827 BEGIN
430 0828 OUTPUT_FAB[FAB$FNA] = UPLIT ('SYSS$OUTPUT:');
431 0829 OUTPUT_FAB[FAB$B_FNS] = %CHARCOUNT ('SYSS$OUTPUT:');
432 0830 END;
433 0831 END
434 0832 ELSE
435 0833 BEGIN
436 0834 IF .CLI_STATUS EQL CLIS_NEGATED
437 0835 THEN
438 0836 BEGIN
439 0837 OUTPUT_FAB[FAB$FNA] = UPLIT ('NL:');
440 0838 OUTPUT_FAB[FAB$B_FNS] = %CHARCOUNT ('NL:');
441 0839 OUTPUT_FAB[FAB$V_SPL] = 0;
442 0840 OUTPUT_FAB[FAB$V_DLT] = 0;
443 0841 END;
444 0842 END;
445 0843 IF (QUAL_FLAGS[DIR_V_QUAL_OWNE] = CLISPRESNT ($DESCRIPTOR ('OWNER')))
446 0844 THEN
447 0845 BEGIN
448 0846 QUAL_FLAGS[DIR_V_NEED_PRO] = 1;
449 0847 QUAL_FLAGS[DIR_V_USE_ID] = CLISPRESNT ($DESCRIPTOR ('OWNER.IDENTIFIER'));
450 0848 END;
451 0849 IF (QUAL_FLAGS[DIR_V_QUAL_PROT] = CLISPRESNT ($DESCRIPTOR ('PROTECTION')))
452 0850 THEN
453 0851 BEGIN
454 0852 QUAL_FLAGS[DIR_V_NEED_PRO] = 1;
455 0853 COLUMN_WIDTH = .COLUMN_WIDTH + 23;
456 0854 END;
457 0855 IF (QUAL_FLAGS[DIR_V_QUAL_SECU] = CLISPRESNT ($DESCRIPTOR ('SECURITY')))
458 0856 THEN
459 0857 BEGIN
460 0858 QUAL_FLAGS[DIR_V_NEED_PRO] = 1;
461 0859 QUAL_FLAGS[DIR_V_QUAL_ACL] = QUAL_FLAGS[DIR_V_QUAL_OWNE] =
462 0860 QUAL_FLAGS[DIR_V_QUAL_PROT] = 1;
463 0861 COLUMN_WIDTH = .COLUMN_WIDTH + 23;
464 0862 END;
465 0863 IF CLISPRESNT ($DESCRIPTOR ('SELECT'))
466 0864 THEN
467 0865 BEGIN
468 0866 MIN_BLOCK = 0;
469 0867 MAX_BLOCK = 1073741823;
470 0868 IF CLISPRESNT ($DESCRIPTOR ('SELECT.SIZE.MINIMUM_SIZE'))
471 0869 THEN
472 0870 BEGIN
473 0871 QUAL_FLAGS[DIR_V_SELE_SIZE] = 1;
474 0872 CLISGET VALUE ($DESCRIPTOR ('SELECT.SIZE.MINIMUM_SIZE'), VALUE_DESC);
475 0873 STATUS = LIB$CVT_DTB (.VALUE_DESC[DSC$W_LENGTH],
476 0874 .VALUE_DESC[DSC$A_POINTER],
477 0875 MIN_BLOCK);

```

```

478 0876 4      IF NOT .STATUS OR .MIN_BLOCK LSS 0
479 0877 4      THEN
480 0878 5          BEGIN
481 0879 5          SIGNAL (DIR$ SYNTAX, 1, VALUE_DESC);
482 0880 5          RETURN .WORST_ERROR;
483 0881 4          END;
484 0882 4          QUAL_FLAGS[DIR_V_NEED_FHC] = 1;
485 0883 3          END;
486 0884 3      IF CL$PRESENT ($DESCRIPTOR ('SELECT.SIZE.MAXIMUM_SIZE'))
487 0885 3      THEN
488 0886 4          BEGIN
489 0887 4          QUAL_FLAGS[DIR_V_SELE_SIZE] = 1;
490 0888 4          CL$GET VALUE ($DESCRIPTOR ('SELECT.SIZE.MAXIMUM_SIZE'), VALUE_DESC);
491 0889 4          STATUS = LIB$CVT_DTB (.VALUE_DESC[DSC$W_LENGTH],
492 0890 4          .VALUE_DESC[DSC$A_POINTER],
493 0891 4          MAX_BLOCK);
494 0892 4          IF NOT .STATUS OR .MAX_BLOCK LSS 0
495 0893 4          THEN
496 0894 5              BEGIN
497 0895 5              SIGNAL (DIR$ SYNTAX, 1, VALUE_DESC);
498 0896 5              RETURN .WORST_ERROR;
499 0897 4              END;
500 0898 4          QUAL_FLAGS[DIR_V_NEED_FHC] = 1;
501 0899 3          END;
502 0900 2      END;
503 0901 2      IF (QUAL_FLAGS[DIR_V_QUAL_SIZE] = CL$PRESENT ($DESCRIPTOR ('SIZE')))
504 0902 2      THEN
505 0903 3          BEGIN
506 0904 3          QUAL_FLAGS[DIR_V_NEED_FHC] = 1;
507 0905 3          IF CL$PRESENT ($DESCRIPTOR ('SIZE.ALL'))
508 0906 3          THEN QUAL_FLAGS[DIR_V_SIZE_ALL] = 1;
509 0907 3          IF CL$PRESENT ($DESCRIPTOR ('SIZE.ALLOCATION'))
510 0908 3          THEN QUAL_FLAGS[DIR_V_SIZE_ALLO] = 1;
511 0909 3          IF CL$PRESENT ($DESCRIPTOR ('SIZE.USED'))
512 0910 3          THEN QUAL_FLAGS[DIR_V_SIZE_USED] = 1;
513 0911 2          END;
514 0912 2      QUAL_FLAGS[DIR_V_QUAL_TOTL] = CL$PRESENT ($DESCRIPTOR ('TOTAL'));
515 0913 2      QUAL_FLAGS[DIR_V_QUAL_TRAI] = CL$PRESENT ($DESCRIPTOR ('TRAILING'));
516 0914 3      IF (QUAL_FLAGS[DIR_V_QUAL_VERS] = CL$PRESENT ($DESCRIPTOR ('VERSIONS')))
517 0915 3      THEN
518 0916 3          BEGIN
519 0917 3          CL$GET VALUE ($DESCRIPTOR ('VERSIONS'), VALUE_DESC);
520 0918 3          STATUS = LIB$CVT_DTB (.VALUE_DESC[DSC$W_LENGTH],
521 0919 3          .VALUE_DESC[DSC$A_POINTER],
522 0920 3          VERSION_COUNT);
523 0921 3          IF NOT .STATUS OR .VERSION_COUNT LEQ 0
524 0922 3          THEN
525 0923 4              BEGIN
526 0924 4              SIGNAL (DIR$ SYNTAX, 1, VALUE_DESC);
527 0925 4              RETURN .WORST_ERROR;
528 0926 3              END;
529 0927 2          END;
530 0928 2      IF (QUAL_FLAGS[DIR_V_QUAL_WIDT] = CL$PRESENT ($DESCRIPTOR ('WIDTH')))
531 0929 2      THEN
532 0930 3          BEGIN
533 0931 3          CL$GET VALUE ($DESCRIPTOR ('WIDTH.DISPLAY'), VALUE_DESC);
534 0932 3          STATUS = LIB$CVT_DTB (.VALUE_DESC[DSC$W_LENGTH],

```

```

535 0933 .VALUE_DESC[DSC$A_POINTER],
536 0934 DISPLAY WIDTH);
537 0935 IF NOT .STATUS OR .DISPLAY_WIDTH LSS 0 !*****
538 0936 THEN
539 0937 BEGIN
540 0938 SIGNAL (DIR$ SYNTAX, 1, VALUE_DESC);
541 0939 RETURN .WORST_ERROR;
542 0940 END;
543 0941 CLISGET VALUE ($DESCRIPTOR ('WIDTH.FILENAME'), VALUE_DESC);
544 0942 STATUS = LIB$CVT_DTB (.VALUE_DESC[DSC$W_LENGTH],
545 0943 .VALUE_DESC[DSC$A_POINTER],
546 0944 FILENAME WIDTH);
547 0945 IF NOT .STATUS OR .FILENAME_WIDTH LSS 0 !*****
548 0946 THEN
549 0947 BEGIN
550 0948 SIGNAL (DIR$ SYNTAX, 1, VALUE_DESC);
551 0949 RETURN .WORST_ERROR;
552 0950 END;
553 0951 IF .FILENAME_WIDTH EQL 0 THEN FILENAME_WIDTH = 19; !*****
554 0952 CLISGET VALUE ($DESCRIPTOR ('WIDTH.OWNER'), VALUE_DESC);
555 0953 STATUS = LIB$CVT_DTB (.VALUE_DESC[DSC$W_LENGTH],
556 0954 .VALUE_DESC[DSC$A_POINTER],
557 0955 OWNER WIDTH);
558 0956 IF NOT .STATUS OR .OWNER_WIDTH LSS 0 !*****
559 0957 THEN
560 0958 BEGIN
561 0959 SIGNAL (DIR$ SYNTAX, 1, VALUE_DESC);
562 0960 RETURN .WORST_ERROR;
563 0961 END;
564 0962 IF .OWNER_WIDTH EQL 0 THEN OWNER_WIDTH = 20; !*****
565 0963 CLISGET VALUE ($DESCRIPTOR ('WIDTH.SIZE'), VALUE_DESC);
566 0964 STATUS = LIB$CVT_DTB (.VALUE_DESC[DSC$W_LENGTH],
567 0965 .VALUE_DESC[DSC$A_POINTER],
568 0966 SIZE WIDTH);
569 0967 IF NOT .STATUS OR .SIZE_WIDTH LSS 0 !*****
570 0968 THEN
571 0969 BEGIN
572 0970 SIGNAL (DIR$ SYNTAX, 1, VALUE_DESC);
573 0971 RETURN .WORST_ERROR;
574 0972 END;
575 0973 IF .SIZE_WIDTH EQL 0 THEN SIZE_WIDTH = 6; !*****
576 0974 END;
577 0975
578 0976 ! Open the specified output file/device.
579 0977
580 0978 STATUS = $CREATE (FAB = OUTPUT_FAB);
581 0979 IF NOT .STATUS
582 0980 THEN
583 0981 BEGIN
584 0982 DIR$FILE_ERROR (DIR$_OPENOUT, OUTPUT_FAB);
585 0983 RETURN .WORST_ERROR;
586 0984 END;
587 0985 STATUS = $CONNECT (RAB = OUTPUT_RAB);
588 0986 IF NOT .STATUS
589 0987 THEN
590 0988 BEGIN
591 0989 DIR$FILE_ERROR (DIR$_OPENOUT, OUTPUT_FAB);

```

```

: 592 0990 3 RETURN .WORST_ERROR;
: 593 0991 END;
: 594 0992
: 595 0993 ! Determine the width of the output device.
: 596 0994
: 597 0995 IF (.OUTPUT_FAB[FAB$DEV]) < $BITPOSITION (DEV$V_TRM), 1>
: 598 0996 THEN
: 599 0997 BEGIN
600 0998 CH$FILL (0, 7*4, GETDVI_ARGS);
601 0999 GETDVI_ARGS[0] = DVI$ DEVCLASS^16 OR 4;
602 1000 GETDVI_ARGS[1] = INDEV_CLASS;
603 1001 GETDVI_ARGS[3] = DVI$ DEVBUFSIZ^16 OR 4;
604 1002 GETDVI_ARGS[4] = INDEV_BUFSIZ;
605 1003
606 P 1004 STATUS = $GETDVI (DEVNAM = $DESCRIPTOR ('SYS$OUTPUT'),
607 1005 ITMLST = GETDVI_ARGS);
608 1006
609 1007 IF NOT .STATUS
610 1008 THEN
611 1009 BEGIN
612 1010 SIGNAL (.STATUS);
613 1011 RETURN .WORST_ERROR;
614 1012 END;
615 1013 IF .DISPLAY_WIDTH EQL 0
616 1014 THEN
617 1015 BEGIN
618 1016 IF .INDEV_CLASS NEQ DC$ TERM THEN INDEV_BUFSIZ = 132;
619 1017 DISPLAY_WIDTH = .INDEV_BUFSIZ;
620 1018 END;
621 1019
622 1020 ! If the number of columns is defaulted and an information qualifier is
623 1021 ! specified, set the column count to 1.
624 1022
625 1023 IF (.QUAL_FLAGS[DIR_V_QUAL_DATE] OR .QUAL_FLAGS[DIR_V_QUAL_OWNE]
626 1024 OR .QUAL_FLAGS[DIR_V_QUAL_PROT] OR .QUAL_FLAGS[DIR_V_QUAL_SIZE]
627 1025 OR .QUAL_FLAGS[DIR_V_QUAL_FID] OR NOT .QUAL_FLAGS[DIR_V_QUAL_HEAD])
628 1026 AND .QUAL_FLAGS[DIR_V_CO$U_DEF]
629 1027 THEN COLUMN_COUNT = 1;
630 1028
631 1029 ! Check to see if XABs are needed to gather information.
632 1030
633 1031 IF .QUAL_FLAGS[DIR_V_NEED_FHC]
634 1032 THEN
635 1033 BEGIN
636 1034 IF .FIRST_XAB EQL 0
637 1035 THEN FIRST_XAB = XAB_PTR = INFO_XABFHC
638 1036 ELSE (XAB_PTR[XAB$RXT] = INFO_XABFHC; XAB_PTR = INFO_XABFHC);
639 1037 END;
640 1038 IF .QUAL_FLAGS[DIR_V_NEED_DAT]
641 1039 THEN
642 1040 BEGIN
643 1041 IF .FIRST_XAB EQL 0
644 1042 THEN FIRST_XAB = XAB_PTR = INFO_XABDAT
645 1043 ELSE (XAB_PTR[XAB$RXT] = INFO_XABDAT; XAB_PTR = INFO_XABDAT);
646 1044 END;
647 1045 IF .QUAL_FLAGS[DIR_V_NEED_PRO]
648 1046 THEN

```

```

649 1047 BEGIN
650 1048 IF .FIRST_XAB EQL 0
651 1049 THEN FIRST_XAB = XAB_PTR = INFO_XABPRO
652 1050 ELSE (XAB_PTR[XAB$R_NXT] = INFO_XABPRO; XAB_PTR = INFO_XABPRO);
653 1051 END;
654 1052 IF .QUAL_FLAGS[DIR_V_NEED_SUM]
655 1053 THEN
656 1054 BEGIN
657 1055 IF .FIRST_XAB EQL 0
658 1056 THEN FIRST_XAB = XAB_PTR = INFO_XABSUM
659 1057 ELSE (XAB_PTR[XAB$R_NXT] = INFO_XABSUM; XAB_PTR = INFO_XABSUM);
660 1058 END;
661 1059 IF .QUAL_FLAGS[DIR_V_NEED_JNL]
662 1060 THEN
663 1061 BEGIN
664 1062 IF .FIRST_XAB EQL 0
665 1063 THEN FIRST_XAB = XAB_PTR = INFO_XABJNL
666 1064 ELSE (XAB_PTR[XAB$R_NXT] = INFO_XABJNL; XAB_PTR = INFO_XABJNL);
667 1065 INFO_XABJNL[XAB$A_AIA] = DISPLAY_BLOCK[DIR_T_AI_NAME];
668 1066 INFO_XABJNL[XAB$B_AIS] = XAB$C_MAXJNLNAM;
669 1067 INFO_XABJNL[XAB$B_BIA] = DISPLAY_BLOCK[DIR_T_BI_NAME];
670 1068 INFO_XABJNL[XAB$B_BIS] = XAB$C_MAXJNLNAM;
671 1069 INFO_XABJNL[XAB$A_ATA] = DISPLAY_BLOCK[DIR_T_AT_NAME];
672 1070 INFO_XABJNL[XAB$B_ATS] = XAB$C_MAXJNLNAM;
673 1071 END;
674 1072
675 1073 ! At this point all of the qualifiers have been parsed. Now determine the
676 1074 ! column width and the maximum number of columns that can be printed given
677 1075 ! specified (or default) display width. This value is minimized with the
678 1076 ! value given on the /COLUMN qualifier.
679 1077
680 1078 COLUMN_WIDTH = .COLUMN_WIDTH + .FILENAME_WIDTH + 1;
681 1079 IF .QUAL_FLAGS[DIR_V_QUAL_OWNE] THEN COLUMN_WIDTH = .COLUMN_WIDTH + .OWNER_WIDTH + 2;
682 1080 IF .QUAL_FLAGS[DIR_V_QUAL_SIZE]
683 1081 THEN
684 1082 BEGIN
685 1083 IF .QUAL_FLAGS[DIR_V_SIZE_ALL]
686 1084 THEN COLUMN_WIDTH = .COLUMN_WIDTH + .SIZE_WIDTH * 2 + 2
687 1085 ELSE COLUMN_WIDTH = .COLUMN_WIDTH + .SIZE_WIDTH + 2;
688 1086 END;
689 1087 IF (.QUAL_FLAGS[DIR_V_DATE_CRE] OR .QUAL_FLAGS[DIR_V_DATE_MOD]
690 1088 OR .QUAL_FLAGS[DIR_V_DATE_EXP] OR .QUAL_FLAGS[DIR_V_DATE_BAK]
691 1089 OR .QUAL_FLAGS[DIR_V_QUAL_OWNE] OR .QUAL_FLAGS[DIR_V_QUAL_PROT]
692 1090 OR .QUAL_FLAGS[DIR_V_QUAL_SIZE] OR .QUAL_FLAGS[DIR_V_QUAL_FID])
693 1091 THEN
694 1092 BEGIN
695 1093 COLUMN_WIDTH = .COLUMN_WIDTH + 4;
696 1094 COLUMN_COUNT = MINU (.COLUMN_COUNT, (.DISPLAY_WIDTH + 4) / .COLUMN_WIDTH);
697 1095 END
698 1096 ELSE COLUMN_COUNT = MINU (.COLUMN_COUNT, .DISPLAY_WIDTH / .COLUMN_WIDTH);
699 1097 IF .COLUMN_COUNT LEQ 0 OR .QUAL_FLAGS[DIR_V_QUAL_ACL] THEN COLUMN_COUNT = 1;
700 1098
701 1099 ! LIB$QUAL_FILE_PARSE is going to parse the common qualifiers. It sets up
702 1100 ! a data base which describes the results for LIB$QUAL_FILE_MATCH to use.
703 1101
704 1102 STATUS = LIB$QUAL_FILE_PARSE (%REF (LIB$M_COF_BACKUP OR
705 1103 LIB$M_COF_BEFORE OR

```



```

706 1104 2 LIBSM_CQF_CREATED OR
707 1105 2 LIBSM_CQF_EXCLUDE OR
708 1106 2 LIBSM_CQF_EXPIRED OR
709 1107 2 LIBSM_CQF_MODIFIED OR
710 1108 2 LIBSM_CQF_SINCE OR
711 1109 2 LIBSM_CQF_BYOWNER
712 1110 2 ), (CMR_QUAL_CTX);
713 1111 2 IF NOT .STATUS
714 1112 2 THEN
715 1113 2 BEGIN
716 1114 2 SIGNAL (.STATUS);
717 1115 2 RETURN .WORST_ERROR;
718 1116 2 END;
719 1117 2
720 1118 2 (LISGET VALUE ($DESCRIPTOR ('INPUT'), FILE_DESC);
721 1119 2 INPUT_FAB[FAB$L_FNA] = .FILE_DESC[DSC$A_POINTER];
722 1120 2 INPUT_FAB[FAB$B_FNS] = .FILE_DESC[DSC$W_LENGTH];
723 1121 2
724 1122 2
725 1123 2 ! If /FULL or /ACL, then image activate SECURESHR, which contains
726 1124 2 ! the routine SYSSFORMAT_ACL.
727 1125 2
728 1126 2 IF .QUAL_FLAGS[DIR_V_QUAL_FULL]
729 1127 2 OR .QUAL_FLAGS[DIR_V_QUAL_ACL]
730 1128 2 THEN BEGIN
731 1129 2 STATUS = LIB$FIND_IMAGE_SYMBOL($DESCRIPTOR('SECURESHR'),
732 1130 2 $DESCRIPTOR('SYSSFORMAT_ACL'), FORMAT_ACL_ADDR);
733 1131 2
734 1132 2 IF NOT .STATUS
735 1133 2 THEN BEGIN
736 1134 2 SIGNAL (.STATUS);
737 1135 2 RETURN .WORST_ERROR;
738 1136 2 END;
739 1137 2
740 1138 2 ! Process each file specification specified in the command line.
741 1139 2
742 1140 2 DO
743 1141 2 BEGIN
744 1142 2
745 1143 2 ! The following is a KLUDGE to get the XAB information across the network.
746 1144 2 ! If the NOP field of the NAM block has the SRCHXABS flag set, then any
747 1145 2 ! XABs (supported by the DAP protocol) connected to the FAB are filled in.
748 1146 2
749 1147 2 IF .QUAL_FLAGS[DIR_V_NEED_FHC] OR .QUAL_FLAGS[DIR_V_NEED_DAT]
750 1148 2 OR .QUAL_FLAGS[DIR_V_NEED_PRO] OR .QUAL_FLAGS[DIR_V_NEED_SUM]
751 1149 2 OR .QUAL_FLAGS[DIR_V_NEED_JNL]
752 1150 2 THEN
753 1151 2 BEGIN
754 1152 2 INPUT_NAM[NAM$V_SRCHXABS] = 1;
755 1153 2 INPUT_FAB[FAB$L_XAB] = .FIRST_XAB;
756 1154 2 END;
757 1155 2
758 1156 2 LIB$FILE_SCAN (INPUT_FAB,
759 1157 2 DIR$GET_INFO, ! File found action routine
760 1158 2 DIR$INPUT_ERROR, ! Input error action routine
761 1159 2 SCAN_CONTEXT); ! Context for stickyness
762 1160 2 END

```

```

: 763      1161 2 UNTIL NOT DIR$GET_FILE(INPUT_FAB);
: 764      1162
: 765      1163 2 IF .LINE_DESC[DSC$W_LENGTH] GTR 0 THEN DIR$OUTPUT (0, LINE_DESC);
: 766      1164 2 IF .TOTAL_FILES NEQ 0 THEN DIR$TOTAL ();
: 767      1165 2 IF .GRAND_DIRS GTR 1
: 768      1166 2 OR .QUAL_FLAGS[DIR_V_QUAL_GRAN]
: 769      1167 2 THEN DIR$GRAND_TOTAL ();                ! Display grand totals
: 770      1168
: 771      1169 2 ! If no files have been selected, and no other errors have occurred, return
: 772      1170 2 ! a status of RMSS_FNF instead of success.
: 773      1171
: 774      1172 2 IF .WORST_ERROR AND NOT .QUAL_FLAGS[DIR_V_FILE_FOUND]
: 775      1173 2 THEN
: 776      1174 2 BEGIN
: 777      1175 2 SIGNAL (DIR$NOFILES);
: 778      1176 2 WORST_ERROR = (RMSS_FNF AND NOT ST$M_SEVERITY) OR ST$K_WARNING
: 779      1177 2 OR ST$M_INHIB_MSG;
: 780      1178 2 END;
: 781      1179
: 782      1180 2 STATUS = $CLOSE (FAB = OUTPUT_FAB);
: 783      1181 2 IF NOT .STATUS THEN DIR$FILE_ERROR (DIR$_CLOSEOUT, OUTPUT_FAB);
: 784      1182
: 785      1183 2 RETURN .WORST_ERROR;
: 786      1184
: 787      1185 2 END;                ! End of routine DIR_MAIN

```

```

.TITLE DIRECTORY
.IDENT \V04-000\
.PSECT DIR$COMMON,NOEXE, OVR,0

```

```

00000 QUAL_FLAGS:
      .BLKB 8
00008 COLUMN_COUNT:
      .BLKB 4
0000C COLUMN_INDEX:
      .BLKB 4
00010 COLUMN_WIDTH:
      .BLKB 4
00014 WORST_ERROR:
      .BLKB 4
00018 CMN_QUAL_CTX:
      .BLKB 4
0001C DISPLAY_BLOCK:
      .BLKB 4
00020 CHANNEL: .BLKB 4
00024 DEVICE_NAME:
      .BLKB 16
00034 LINE_DESC:
      .BLKB 8
0003C LINE_BUFFER:
      .BLKB 1024
0043C TOTAL_USED:
      .BLKB 4
00440 TOTAL_ALLOC:
      .BLKB 4

```

	00444	TOTAL_FILES:		
		.BLKB	4	
	00448	GRAND_USED:		
		.BLKB	4	
	0044C	GRAND_ALLOC:		
		.BLKB	4	
	00450	GRAND_FILES:		
		.BLKB	4	
	00454	GRAND_DIRS:		
		.BLKB	4	
	00458	PREV_DIR:		
		.BLKB	255	
	00557		1	
	00558	PREV_DIR_LEN:		
		.BLKB	4	
	0055C	PREV_FILE:		
		.BLKB	255	
	0065B		1	
	0065C	PREV_FILE_LEN:		
		.BLKB	4	
	00660	VERSION_COUNT:		
		.BLKB	4	
	00664	VERSION_INDEX:		
		.BLKB	4	
	00668	FIRST_XAB:		
		.BLKB	4	
22	0066C	INFO_XABJNL:		
		.BYTE	34	
3C	0066D		60	
		.BYTE		
0000	0066E	.WORD	0	
00000000	00670	.LONG	0	
0000	00674	.WORD	0	
0000	00676	.WORD	0	
00	00678	.BYTE	0	
00	00679	.BYTE	0	
0000	0067A	.WORD	0	
00000000	0067C	.LONG	0	
00	00680	.BYTE	0	
00	00681	.BYTE	0	
0000	00682	.WORD	0	
00000000	00684	.LONG	0	
00	00688	.BYTE	0	
00	00689	.BYTE	0	
0000	0068A	.WORD	0	
00000000	0068C	.LONG	0	
	00690	.BLKB	24	
16	006A8	INFO_XABSUM:		
		.BYTE	22	
0C	006A9	.BYTE	12	
0000	006AA	.WORD	0	
00000000	006AC	.LONG	0	
00	006B0	.BYTE	0	
00	006B1	.BYTE	0	
0000	006B2	.WORD	0	
13	006B4	INFO_XABPRO:		
		.BYTE	19	
58	006B5	.BYTE	86	

.....
.....
.....

0000	006B6	.WORD	0
00000000	006B8	.LONG	0
FFFF	006BC	.WORD	-1
00	006BE	.BYTE	0
00	006BF	.BYTE	0
0000 0000	006C0	.WORD	0, 0
00	006C4	.BYTE	0
00	006C5	.BYTE	0
0000	006C6	.WORD	0
00000000	006C8	.LONG	0
00000000	006CC	.LONG	0
0000	006D0	.WORD	0
0000	006D2	.WORD	0
00000000	006D4	.LONG	0
00000000	006D8	.LONG	0
	006DC	.BLKB	48
12	0070C	INFO_XABDAT:	
		.BYTE	18
2C	0070D	.BYTE	44
0000	0070E	.WORD	0
00000000	00710	.LONG	0
0000	00714	.WORD	0
0000	00716	.WORD	0
00000000#	00718	.LONG	0[2]
00000000#	00720	.LONG	0[2]
00000000	00728	.LONG	0
00000000	0072C	.LONG	0
00000000#	00730	.LONG	0[2]
1D	00738	INFO_XABFHC:	
		.BYTE	29
2C	00739	.BYTE	44
0000	0073A	.WORD	0
00000000	0073C	.LONG	0
00000000#	00740	.LONG	0[9]
02	00764	INFO_NAM:	
		.BYTE	2
60	00765	.BYTE	96
00	00766	.BYTE	0
00	00767	.BYTE	0
00000000	00768	.LONG	0
00	0076C	.BYTE	0
00	0076D	.BYTE	0
00	0076E	.BYTE	0
00	0076F	.BYTE	0
00000000	00770	.LONG	0
00000000	00774	.LONG	0
0000#	00778	.WORD	0[8]
0000#	00788	.WORD	C[3]
0000#	0078E	.WORD	0[3]
00000000	00794	.LONG	0
00000000	00798	.LONG	0
00	0079C	.BYTE	0
00	0079D	.BYTE	0
00	0079E	.BYTE	0
00	0079F	.BYTE	0
00	007A0	.BYTE	0
00	007A1	.BYTE	0

.....

```
00# 007A2 .BYTE 0[2]
00000000 007A4 .LONG 0
00000000 007A8 .LONG 0
00000000 007AC .LONG 0
00000000 007B0 .LONG 0
00000000 007B4 .LONG 0
00000000 007B8 .LONG 0
00000000# 007BC .LONG 0[2]
03 007C4 INFO_FAB:
      .BYTE 3
      .BYTE 80
      .WORD 0
01000000 007C8 .LONG 16777216
00000000 007CC .LONG 0
00000000 007D0 .LONG 0
00000000 007D4 .LONG 0
0000 007D8 .WORD 0
02 007DA .BYTE 2
43 007DB .BYTE 67
00000000 007DC .LONG 0
00 007E0 .BYTE 0
00 007E1 .BYTE 0
00 007E2 .BYTE 0
02 007E3 .BYTE 2
00000000 007E4 .LONG 0
00000000 007E8 .LONG 0
00000000 007EC .ADDRESS INFO_NAM
00000000 007F0 .LONG 0
00000000 007F4 .LONG 0
00 007F8 .BYTE 0
00 007F9 .BYTE 0
0000 007FA .WORD 0
00000000 007FC .LONG 0
0000 00800 .WORD 0
00 00802 .BYTE 0
00 00803 .BYTE 0
00000000 00804 .LONG 0
00000000 00808 .LONG 0
0000 0080C .WORD 0
00 0080E .BYTE 0
00 0080F .BYTE 0
00000000 00810 .LONG 0
00814 DISPLAY_WIDTH:
      .BLKB 4
00818 FILENAME_WIDTH:
      .BLKB 4
0081C OWNER_WIDTH:
      .BLKB 4
00820 SIZE_WIDTH:
      .BLKB 4
00824 MIN_BLOCK:
      .BLKB 4
00828 MAX_BLOCK:
      .BLKB 4
0082C ACL_LENGTH:
      .BLKB 4
00830 OUTPUT_RAB:
```

.....

```

                                .BLKB 68
                                .PSECT $SPLITS,NOWRT,NOEXE,2
00 00 53 49 4C 2E 59 00 00 00 2A 3B 2A 2E 2A 00000 P.AAA: .ASCII \*.:*:\<0><0><0>
                    52 4F 54 43 45 52 49 44 00008 P.AAB: .ASCII \DIRECTORY.LIS\<0><0><0>
                                00017
                                00018 P.AAD: .ASCII \BEFORE\
                                0001E
                                00020 P.AAC: .BLKB 2
                                00024 P.AAC: .LONG 6
                                00028 P.AAF: .ADDRESS P.AAD
                    45 43 4E 49 53 0002D P.AAF: .ASCII \SINCE\
                                00030 P.AAE: .BLKB 3
                                00034 P.AAE: .LONG 5
                                00038 P.AAH: .ADDRESS P.AAF
                    52 45 4E 57 4F 5F 59 42 00040 P.AAH: .ASCII \BY_OWNER\
                                00044 P.AAG: .LONG 8
                                00048 P.AAJ: .ADDRESS P.AAH
                    4C 43 41 0004B P.AAJ: .ASCII \ACL\
                                0004C P.AAI: .BLKB 1
                                00050 P.AAI: .LONG 3
                                00054 P.AAL: .ADDRESS P.AAJ
                    46 45 49 52 42 00059 P.AAL: .ASCII \BRIEF\
                                00060 P.AAK: .BLKB 3
                                00064 P.AAN: .LONG 5
                                00068 P.AAN: .ADDRESS P.AAL
                    53 4E 4D 55 4C 4F 43 00074 P.AAN: .ASCII \COLUMNS\
                                00078 P.AAM: .BLKB 1
                                00080 P.AAM: .LONG 7
                                00084 P.AAP: .ADDRESS P.AAN
                    53 4E 4D 55 4C 4F 43 00088 P.AAP: .ASCII \COLUMNS\
                                00090 P.AAO: .BLKB 1
                                00098 P.AAO: .LONG 7
                                00100 P.AAP: .ADDRESS P.AAP
                                00104 P.AAR: .ASCII \DATE\
                                00108 P.AAQ: .LONG 4
                                00112 P.AAR: .ADDRESS P.AAR
                    4C 4C 41 2E 45 54 41 44 00120 P.AAT: .ASCII \DATE.ALL\
                                00124 P.AAS: .LONG 8
                                00128 P.AAT: .ADDRESS P.AAT
                    44 45 54 41 45 52 43 2E 45 54 41 44 00140 P.AAV: .ASCII \DATE.CREATED\
                                00144 P.AAU: .LONG 12
                                00148 P.AAV: .ADDRESS P.AAV
                    44 45 52 49 50 58 45 2E 45 54 41 44 00160 P.AAX: .ASCII \DATE.EXPIRED\
                                00164 P.AAW: .LONG 12
                                00168 P.AAX: .ADDRESS P.AAX
                    44 45 49 46 49 44 4F 4D 2E 45 54 41 44 00180 P.AAZ: .ASCII \DATE.MODIFIED\
                                00184 P.AAY: .BLKB 3
                                00188 P.AAY: .LONG 13
                                00192 P.AAZ: .ADDRESS P.AAZ
                    50 55 4B 43 41 42 2E 45 54 41 44 00200 P.ABB: .ASCII \DATE.BACKUP\
                                00204 P.ABB: .BLKB 1
                                00208 P.ABA: .LONG 11
                                00212 P.ABB: .ADDRESS P.ABB
                                00216 P.ABD: .ASCII \FILE_ID\
                    44 49 5F 45 4C 49 46 00220 P.ABD: .BLKB 1
                                00224 P.ABC: .LONG 7
                                00000006
                                00000000
                                00000005
                                00000000
                                00000003
                                00000000
                                00000005
                                00000000
                                00000007
                                00000000
                                00000007
                                00000000
                                00000004
                                00000000
                                00000008
                                00000000
                                0000000C
                                00000000
                                0000000D
                                00000000
                                0000000B
                                00000000
                                00000007

```

.....

00

58	41	4D	2E	45	5A	49	53	2E	54	43	45	4C	45	53	00230	P.ACJ:	.ASCII	\SELECT.SIZE.MAXIMUM_SIZE\				
						45	5A	49	53	5F	4D	55	4D	49	0023F							
															00000018	00248	P.ACI:	.LONG 24				
															00000000	0024C		.ADDRESS P.ACJ				
											45	5A	49	53	00250	P.ACL:	.ASCII	\SIZE\				
															00000004	00254	P.ACK:	.LONG 4				
															00000000	00258		.ADDRESS P.ACL				
							4C	4C	41	2E	45	5A	49	53	0025C	P.ACN:	.ASCII	\SIZE.ALL\				
															00000008	00264	P.ACM:	.LONG 8				
															00000000	00268		.ADDRESS P.ACN				
4E	4F	49	54	41	43	4F	4C	4C	41	2E	45	5A	49	53	0026C	P.ACP:	.ASCII	\SIZE.ALLOCATION\				
															00270	P.ACO:	.BLKB	1				
															0000000F	0027C		.LONG 15				
															00000000	00280		.ADDRESS P.ACP				
						44	45	53	55	2E	45	5A	49	53	00284	P.ACR:	.ASCII	\SIZE.USED\				
															0028D		.BLKB	3				
															00000009	00290	P.ACQ:	.LONG 9				
															00000000	00294		.ADDRESS P.ACR				
										4C	41	54	4F	54	00298	P.ACT:	.ASCII	\TOTAL\				
															0029D		.BLKB	3				
															00000005	002A0	P.ACS:	.LONG 5				
															00000000	002A4		.ADDRESS P.ACT				
						47	4E	49	4C	49	41	52	54		002A8	P.ACV:	.ASCII	\TRAILING\				
															00000008	002B0	P.ACU:	.LONG 8				
															00000000	002B4		.ADDRESS P.ACV				
						53	4E	4F	49	53	52	45	56		002B8	P.ACX:	.ASCII	\VERSIONS\				
															00000008	002C0	P.ACW:	.LONG 8				
															00000000	002C4		.ADDRESS P.ACX				
						53	4E	4F	49	53	52	45	56		002C8	P.ACZ:	.ASCII	\VERSIONS\				
															00000008	002D0	P.ACY:	.LONG 8				
															00000000	002D4		.ADDRESS P.ACZ				
										48	54	44	49	57	002D8	P.ADB:	.ASCII	\WIDTH\				
															002DD		.BLKB	3				
															00000005	002E0	P.ADA:	.LONG 5				
															00000000	002E4		.ADDRESS P.ADB				
						59	41	4C	50	53	49	44	2E	48	54	44	49	57	002E8	P.ADD:	.ASCII	\WIDTH.DISPLAY\
																002F5		.BLKB	3			
															0000000D	002F8	P.ADC:	.LONG 13				
															00000000	002FC		.ADDRESS P.ADD				
45	4D	41	4E	45	4C	49	46	2E	48	54	44	49	57		00300	P.ADF:	.ASCII	\WIDTH.FILENAME\				
															0030E		.BLKB	2				
															0000000E	00310	P.ADE:	.LONG 14				
															00000000	00314		.ADDRESS P.ADF				
						52	45	4E	57	4F	2E	48	54	44	49	57	00318	P.ADH:	.ASCII	\WIDTH.OWNER\		
															00323		.BLKB	1				
															0000000B	00324	P.ADG:	.LONG 11				
															00000000	00328		.ADDRESS P.ADH				
						45	5A	49	53	2E	48	54	44	49	57	0032C	P.ADJ:	.ASCII	\WIDTH.SIZE\			
															00336		.BLKB	2				
															0000000A	00338	P.ADI:	.LONG 10				
															00000000	0033C		.ADDRESS P.ADJ				
						54	55	50	54	55	4F	24	53	59	53	00340	P.ADL:	.ASCII	\SYS\$OUTPUT\			
															0034A		.BLKB	2				
															0000000A	0034C	P.ADK:	.LONG 10				
															00000000	00350		.ADDRESS P.ADL				
										54	55	50	4E	49	00354	P.ADN:	.ASCII	\INPUT\				
															00359		.BLKB	3				

				0660	C8	7C	00038	CLRQ	VERSION COUNT	0666	
				065C	C8	D4	0003C	CLRL	PREV_FILE_LEN	0667	
				0558	C8	D4	00040	CLRL	PREV_DIR_LEN		
				0440	C8	7C	00044	CLRQ	TOTAL_ALLOC	0668	
				043C	C8	D4	00048	CLRL	TOTAL_USED		
				0450	C8	7C	0004C	CLRQ	GRAND_FILES	0669	
				0448	C8	7C	00050	CLRQ	GRAND_USED		
				10	AB	D4	00054	CLRL	COLUMN WIDTH	0670	
				04	AE	7C	00057	CLRQ	INDEV CLASS	0671	
					56	D4	0005A	CLRL	XAB_PTR	0672	
08		00	6E	0668	C8	D4	0005C	CLRL	FIRST_XAB		
					00	2C	00060	MOVCS	#0, (SP), #0, #8, VALUE_DESC	0673	
					2C	AE	00065				
					02	90	00067	MOVB	#2, VALUE_DESC+3	0674	
	34	AE	2F		08	28	0006B	MOVCS	#8, VALUE_DESC, FILE_DESC	0675	
	34	AB	2C		08	28	00071	MOVCS	#8, VALUE_DESC, LINE_DESC	0676	
			38		AB	9E	00077	MOVAB	LINE_BUFFER, LINE_DESC+4	0677	
					3C	AB	9F	PUSHAB	DISPLAY_BLOCK	0681	
			J4		1C	AB	9F	0007C			
					01CB	8F	3C	0007F	MOVZWL	#459, 4(SP)	
					04	AE	9F	00085	PUSHAB	4(SP)	
			00000000G		02	FB	00088	CALLS	#2, LIB\$GET_VM		
					57	D0	0008F	MOVL	R0, STATUS		
					3D	57	EB	00092	BLBS	STATUS, 4\$	
				0830	C8	9F	00095	1\$: PUSHAB	OUTPUT_RAB	0682	
			00000000G		01	FB	00099	CALLS	#1, SYS\$FLUSH	0685	
				0830	C8	9F	000A0	PUSHAB	OUTPUT_RAB		
			00000000G		01	FB	000A4	CALLS	#1, SYS\$WAIT		
					57	DD	000AB	PUSHL	STATUS		
			00000000G		01	FB	000AD	CALLS	#1, LIB\$SIGNAL		
					07	57	93	000B4	BITB	STATUS, #7	
					16	13	000B7	BEQL	3\$		
50		57			00	EF	000B9	EXTZV	#0, #3, STATUS, R0		
50	14	AB			00	ED	000BE	CMPZV	#0, #3, WORST_ERROR, R0		
					09	18	000C4	BGEQ	3\$		
	14	AB		57	10000000	8F	C9	000C6	2\$: BISL3	#268435456, STATUS, WORST_ERROR	
						087F	31	000CF	3\$: BRW	86\$	
0050	8F		00		6E	00	2C	000D2	4\$: MOVCS	#0, (SP), #0, #80, \$RMS_PTR	
					BC	AD	000D9				
					5003	8F	80	000DB	MOVW	#20483, \$RMS_PTR	
						02	90	000E1	MOVB	#2, \$RMS_PTR+22	
						02	90	000E5	MOVB	#2, \$RMS_PTR+31	
					FF50	CD	9E	000E9	MOVAB	INPUT_NAME, \$RMS_PTR+40	
						6A	9E	000EF	MOVAB	P.AAA, \$RMS_PTR+48	
0060	8F		00		6E	05	90	000F3	MOVB	#5, \$RMS_PTR+53	
						00	2C	000F7	MOVCS	#0, (SP), #0, #96, \$RMS_PTR	
					FF50	CD	000FE				
					6002	8F	80	00101	MOVW	#24578, \$RMS_PTR	
						01	8E	00108	MNEGB	#1, \$RMS_PTR+2	
						3C	AE	9E	0010D	MOVAB	INP_RES_NAME, \$RMS_PTR+4
						01	8E	00113	MNEGB	#1, \$RMS_PTR+10	
0050	8F		00		013C	CE	9E	00118	MOVAB	INP_EXP_NAME, \$RMS_PTR+12	
						00	2C	0011F	MOVCS	#0, (SP), #0, #80, \$RMS_PTR	
						6B		00126			
					5003	8F	80	00127	MOVW	#20483, \$RMS_PTR	
						40	8F	9A	0012C	MOVZBL	#64, \$RMS_PTR+4
						01	90	00131	MOVB	#1, \$RMS_PTR+22	
					0202	8F	80	00135	MOVW	#514, \$RMS_PTR+30	

			28	AB	50	AB	9E	0013B	MOVAB	OUTPUT_NAM, \$RMS_PTR+40		
			30	AB	08	AA	9F	00140	MOVAB	P.AAB, \$RMS_PTR+48		
0044	8F	00	35	AB		0D	90	00145	MOVAB	#13, \$RMS_PTR+53		
				6E		00	2C	00149	MOVCS	#0, (SP), #0, #68, \$RMS_PTR	0709	
			0830	C8	0830	C8		00150				
			4401	C8	4401	8F	B0	00153	MOVW	#17409, \$RMS_PTR		
0060	8F	00	086C	C8		6B	9E	0015A	MOVAB	OUTPUT_FAB, \$RMS_PTR+60		
				6E		00	2C	0015F	MOVCS	#0, (SP), #0, #98, \$RMS_PTR	0714	
					50	AB		00166				
			50	AB	6002	8F	B0	00168	MOVW	#24578, \$RMS_PTR		
			52	AB		01	8E	0016E	MNEGB	#1, \$RMS_PTR+2		
			54	AB	01B0	CB	9E	00172	MOVAB	OUT_RES_NAM, \$RMS_PTR+4		
			5A	AB		01	8E	00178	MNEGB	#1, \$RMS_PTR+10		
			5C	AB	00B0	CB	9E	0017C	MOVAB	OUT_EXP_NAM, \$RMS_PTR+12		
					20	AA	9F	00182	PUSHAB	P.AAC	0722	
				69		01	FB	00185	CALLS	#1, CLISPRESENT		
				09		50	EB	00188	BLBS	R0, 5\$		
					30	AA	9F	0018B	PUSHAB	P.AAE	0723	
				69		01	FB	0018E	CALLS	#1, CLISPRESENT		
			03	06		50	E9	00191	BLBC	R0, 6\$		
				03	0240	8F	AB	00194	BISW2	#576, QUAL_FLAGS+3	0726	
					40	AA	9F	0019A	PUSHAB	P.AAG	0730	
				69		01	FB	0019D	CALLS	#1, CLISPRESENT		
				06		50	E9	001A0	BLBC	R0, 7\$		
			03	03	0440	8F	AB	001A3	BISW2	#1088, QUAL_FLAGS+3	0733	
					4C	AA	9F	001A9	PUSHAB	P.AAI	0739	
				69		01	FB	001AC	CALLS	#1, CLISPRESENT		
68	01			00		50	F0	001AF	INSV	R0, #0, #1, QUAL_FLAGS		
					5C	AA	9F	001B4	PUSHAB	P.AAK	0740	
				69		01	FB	001B7	CALLS	#1, CLISPRESENT		
68	01			01		50	F0	001BA	INSV	R0, #1, #1, QUAL_FLAGS		
					6C	AA	9F	001BF	PUSHAB	P.AAM	0741	
				69		01	FB	001C2	CALLS	#1, CLISPRESENT		
68	01			02		50	F0	001C5	INSV	R0, #2, #1, QUAL_FLAGS		
				52		50	D0	001CA	MOVL	R0, CLI_STATUS		
				40		50	E9	001CD	BLBC	R0, 11\$		
					2C	AE	9F	001D0	PUSHAB	VALUE_DESC	0744	
					7C	AA	9F	001D3	PUSHAB	P.AAO		
			00C0000G	00		02	FB	001D6	CALLS	#2, CLISGET_VALUE		
					08	AB	9F	001DD	PUSHAB	COLUMN_COUNT	0745	
					34	AE	DD	001E0	PUSHL	VALUE_DESC+4	0746	
				7E	34	AE	3C	001E3	MOVZWL	VALUE_DESC, -(SP)	0745	
			0000000G	00		03	FB	001E7	CALLS	#3, LIB\$CVT_DTB		
				57		50	D0	001EE	MOVL	R0, STATUS		
				03		57	E8	001F1	BLBS	STATUS, 9\$	0748	
					0393	31	001F4	3\$:	BRW	40\$		
					08	AB	D5	001F7	9\$:	TSTL	COLUMN_COUNT	
						F8	19	001FA	BLSS	8\$		
						04	12	001FC	BNEQ	10\$	0754	
			08	AB		01	D0	001FE	MOVL	#1, COLUMN_COUNT		
			0000000G	8F		52	D1	00202	10\$:	CPL	CLI_STATUS, #CLIS_DEFAULTED	0755
						05	12	00209	BNEQ	11\$		
			03	AB	80	8F	88	0020B	BISB2	#128, QUAL_FLAGS+3		
					0088	CA	9F	00210	11\$:	PUSHAB	P.AAQ	0757
				69		01	FB	00214	CALLS	#1, CLISPRESENT		
68	01			03		50	F0	00217	INSV	R0, #3, #1, QUAL_FLAGS		
				62		50	E9	0021C	BLBC	R0, 16\$		

	04	A8		02	88	0021F	BISB2	#2, QUAL_FLAGS+4	0760
			0098	CA	9F	00223	PUSHAB	P.AAS	0761
		69		01	FB	00227	CALLS	#1, CLISPRESNT	
		0E		50	E9	0022A	BLBC	RO, 12\$	
		68	F0	8F	88	0022D	BISB2	#240, QUAL_FLAGS	0767
	10	A8	0000004C	8F	CO	00231	ADDL2	#76, COLUMN_WIDTH	0768
				46	11	00239	BRB	16\$	0761
			00AC	CA	9F	0023B	12\$: PUSHAB	P.AAU	0772
		69		01	FB	0023F	CALLS	#1, CLISPRESNT	
		07		50	E9	00242	BLBC	RO, 13\$	
		68		10	88	00245	BISB2	#16, QUAL_FLAGS	0775
	10	A8		13	CO	00248	ADDL2	#19, COLUMN_WIDTH	0776
			00C0	CA	9F	0024C	13\$: PUSHAB	P.AAW	0778
		69		01	FB	00250	CALLS	#1, CLISPRESNT	
		07		50	E9	00253	BLBC	RO, 14\$	
		68		20	88	00256	BISB2	#32, QUAL_FLAGS	0781
	10	A8		13	CO	00259	ADDL2	#19, COLUMN_WIDTH	0782
			00D8	CA	9F	0025D	14\$: PUSHAB	P.AAY	0784
		69		01	FB	00261	CALLS	#1, CLISPRESNT	
		08		50	E9	00264	BLBC	RO, 15\$	
		68	40	8F	88	00267	BISB2	#64, QUAL_FLAGS	0787
	10	A8		13	CO	0026B	ADDL2	#19, COLUMN_WIDTH	0788
			00EC	CA	9F	0026F	15\$: PUSHAB	P.ABA	0790
		69		01	FB	00273	CALLS	#1, CLISPRESNT	
		08		50	E9	00276	BLBC	RO, 16\$	
		68	80	8F	88	00279	BISB2	#128, QUAL_FLAGS	0793
	10	A8		13	CO	0027D	ADDL2	#19, COLUMN_WIDTH	0794
			00FC	CA	9F	00281	16\$: PUSHAB	P.ABC	0798
		69		01	FB	00285	CALLS	#1, CLISPRESNT	
01	A8		01	50	F0	00288	INSV	RO, #0, #1, QUAL_FLAGS+1	
		04		50	E9	0028E	BLBC	RO, 17\$	
	10	A8		15	CO	00291	ADDL2	#21, COLUMN_WIDTH	0799
			0108	CA	9F	00295	17\$: PUSHAB	P.ABE	0800
		69		01	FB	00299	CALLS	#1, CLISPRESNT	
01	A8		01	50	F0	0029C	INSV	RO, #1, #1, QUAL_FLAGS+1	
		04		50	E9	002A2	BLBC	RO, 18\$	
	04	A8		1F	88	002A5	BISB2	#31, QUAL_FLAGS+4	0805
			J11C	CA	9F	002A9	18\$: PUSHAB	P.ABG	0807
		69		01	FB	002AD	CALLS	#1, CLISPRESNT	
01	A8		01	50	F0	002B0	INSV	RO, #2, #1, QUAL_FLAGS+1	
		02		CA	9F	002B6	PUSHAB	P.ABI	0808
		69		01	FB	002BA	CALLS	#1, CLISPRESNT	
01	A8		01	50	F0	002BD	INSV	RO, #3, #1, QUAL_FLAGS+1	
		03		CA	9F	002C3	PUSHAB	P.ABK	0813
		69		01	FB	002C7	CALLS	#1, CLISPRESNT	
01	A8		01	50	F0	002CA	INSV	RO, #6, #1, QUAL_FLAGS+1	
		05		50	E9	002D0	BLBC	RO, 19\$	
	05	A8		8F	88	002D3	BISB2	#160, OUTPUT_FAB+5	0817
			A0	CA	9F	002D8	19\$: PUSHAB	P.ABM	0819
		69		01	FB	002DC	CALLS	#1, CLISPRESNT	
01	A8		01	50	F0	002DF	INSV	RO, #4, #1, QUAL_FLAGS+1	
		04		50	D0	002E5	MOVL	RO, CLI_STATUS	
		52		50	E9	002E8	BLBC	RO, 20\$	
		30		AE	9F	002EB	PUSHAB	FILE_DESC	0822
			34	CA	9F	002EE	PUSHAB	P.ABO	
			015C	02	FB	002F2	CALLS	#2, CLISGET_VALUE	
	00000000G	00		AE	D0	002F9	MOVL	FILE_DESC+4, OUTPUT_FAB+44	0823
	2C	A8		38					

		50	34	AE 3C 002FE	MOVZWL	FILE_DESC, R0	0824	
	34	AB		50 90 00302	MOVB	R0, OUTPUT_FAB+52		
				50 D5 00306	TSTL	R0		
				29 12 00308	BNEQ	21\$		
24	01	AB		06 E0 0030A	BBS	#6, QUAL_FLAGS+1, 21\$	0825	
	2C	AB	0164	CA 9E 0030F	MOVAB	P.ABQ, OUTPUT_FAB+44	0828	
	34	AB		0B 90 00315	MOVB	#11, OUTPUT_FAB+52	0829	
				18 11 00319	BRB	21\$	0819	
	00000000G	8F		52 D1 0031B	20\$:	CMP	CLI_STATUS, #CLIS_NEGATED	0834
				0F 12 00322	BNEQ	21\$		
	2C	AB	0170	CA 9E 00324	MOVAB	P.ABR, OUTPUT_FAB+44	0837	
	34	AB		03 90 0032A	MOVB	#3, OUTPUT_FAB+52	0838	
	05	AB	A0	8F 8A 0032E	BICB2	#160, OUTPUT_FAB+5	0840	
			017C	CA 9F 00333	21\$:	PUSHAB	P.ABS	0843
		69		01 FB 00337	CALLS	#1, CLISPRESENT		
01	AB	01		50 F0 0033A	INSV	R0, #5, #1, QUAL_FLAGS+1		
		11		50 E9 00340	BLBC	R0, 22\$		
	04	AB		04 88 00343	BISB2	#4, QUAL_FLAGS+4	0846	
			0194	CA 9F 00347	PUSHAB	P.ABU	0847	
		69		01 FB 0034B	CALLS	#1, CLISPRESENT		
04	AB	01		50 F0 0034E	INSV	R0, #6, #1, QUAL_FLAGS+4		
		06	01A8	CA 9F 00354	22\$:	PUSHAB	P.ABW	0849
		69		01 FB 00358	CALLS	#1, CLISPRESENT		
01	AB	01		50 F0 0035B	INSV	R0, #7, #1, QUAL_FLAGS+1		
		08		50 E9 00361	BLBC	R0, 23\$		
	04	AB		04 88 00364	BISB2	#4, QUAL_FLAGS+4	0852	
	10	AB		17 C0 00368	ADDL2	#23, COLUMN_WIDTH	0853	
			01B8	CA 9F 0036C	23\$:	PUSHAB	P.ABY	0855
		69		01 FB 00370	CALLS	#1, CLISPRESENT		
02	AB	01		50 F0 00373	INSV	R0, #0, #1, QUAL_FLAGS+2		
		0F		50 E9 00379	BLBC	R0, 24\$		
	01	AB	040000A0	8F C8 0037C	BISL2	#67109024, QUAL_FLAGS+1	0860	
		68		01 88 00384	BISB2	#1, QUAL_FLAGS	0859	
	10	AB		17 C0 00387	ADDL2	#23, COLUMN_WIDTH	0861	
			01C8	CA 9F 0038B	24\$:	PUSHAB	P.ACA	0863
		69		01 FB 0038F	CALLS	#1, CLISPRESENT		
		52		50 E9 00392	BLBC	R0, 26\$		
	0828	CB	0824	C8 D4 00395	CLRL	MIN_BLOCK	0866	
			3FFFFFFF	8F D0 00399	MOVL	#1073741823, MAX_BLOCK	0867	
			01E8	CA 9F 003A2	PUSHAB	P.ACC	0868	
		69		01 FB 003A6	CALLS	#1, CLISPRESENT		
		34		50 E9 003A9	BLBC	R0, 25\$		
	02	AB		04 88 003AC	BISB2	#4, QUAL_FLAGS+2	0871	
			2C	AE 9F 003B0	PUSHAB	VALUE_DESC	0872	
			0208	CA 9F 003B3	PUSHAB	P.ACE		
	00000000G	00		02 FB 003B7	CALLS	#2, CLISGET_VALUE		
			0824	C8 9F 003BE	PUSHAB	MIN_BLOCK	0873	
			34	AE DD 003C2	PUSHL	VALUE_DESC+4	0874	
			34	AE 3C 003C5	MOVZWL	VALUE_DESC, -(SP)	0873	
	00000000G	00		03 FB 003C9	CALLS	#3, LIB\$CVT_DTB		
		57		50 D0 003D0	MOVL	R0, STATUS		
		3B		57 E9 003D3	BLBC	STATUS, 27\$	0876	
			0824	C8 D5 003D6	TSTL	MIN_BLOCK		
				3F 19 003DA	BLSS	30\$		
	04	AB		01 88 003DC	BISB2	#1, QUAL_FLAGS+4	0882	
			0228	CA 9F 003E0	25\$:	PUSHAB	P.ACG	0884
		69		01 FB 003E4	CALLS	#1, CLISPRESENT		

			37	50	E9	003E7	26\$:	BLBC	R0, 31\$		
	02		A8	04	88	003EA		BISB2	#4, QUAL_FLAGS+2		0887
				2C	AE	9F	003EE	PUSHAB	VALUE_DESC		0888
				0248	CA	9F	003F1	PUSHAB	P.ACI		
		00000000G	00	02	FB	003F5		CALLS	#2, CLISGET_VALUE		
				0828	C8	9F	003FC	PUSHAB	MAX_BLOCK		0889
				34	AE	DD	00400	PUSHL	VALUE_DESC+4		0890
			7E	34	AE	3C	00403	MOVZWL	VALUE_DESC, -(SP)		0889
		00000000G	00	03	FB	00407		CALLS	#3, LIB\$CVT_DTB		
			57	50	DO	0040E		MOVL	R0, STATUS		
			03	57	E8	00411	27\$:	BLBS	STATUS, 29\$		0892
				0173	31	00414	28\$:	BRW	40\$		
				0828	C8	D5	00417	29\$:	TSTL	MAX_BLOCK	
					F7	19	0041B	30\$:	BLSS	28\$	
			04	01	88	0041D		BISB2	#1, QUAL_FLAGS+4		0898
				0254	CA	9F	00421	31\$:	PUSHAB	P.ACK	0901
			69	01	FB	00425		CALLS	#1, CLISPRESENT		
02	A8		03	50	F0	00428		INSV	R0, #3, #1, QUAL_FLAGS+2		
			2F	50	E9	0042E		BLBC	R0, 34\$		
			04	01	88	00431		BISB2	#1, QUAL_FLAGS+4		0904
				0264	CA	9F	00435	PUSHAB	P.ACM		0905
			69	01	FB	00439		CALLS	#1, CLISPRESENT		
			04	50	E9	0043C		BLBC	R0, 32\$		
			02	10	88	0043F		BISB2	#16, QUAL_FLAGS+2		0906
				027C	CA	9F	00443	32\$:	PUSHAB	P.ACO	0907
			69	01	FB	00447		CALLS	#1, CLISPRESENT		
			04	50	E9	0044A		BLBC	R0, 33\$		
			02	20	88	0044D		BISB2	#32, QUAL_FLAGS+2		0908
				0290	CA	9F	00451	33\$:	PUSHAB	P.ACO	0909
			69	01	FB	00455		CALLS	#1, CLISPRESENT		
			05	50	E9	00458		BLBC	R0, 34\$		
			02	40	8F	88	0045B		BISB2	#64, QUAL_FLAGS+2	0910
				02A0	CA	9F	00460	34\$:	PUSHAB	P.ACS	0912
			69	01	FB	00464		CALLS	#1, CLISPRESENT		
02	A8		07	50	F0	00467		INSV	R0, #7, #1, QUAL_FLAGS+2		
				02B0	CA	9F	0046D	PUSHAB	P.ACU		0913
			69	01	FB	00471		CALLS	#1, CLISPRESENT		
03	A8		00	50	F0	00474		INSV	R0, #0, #1, QUAL_FLAGS+3		
				02C0	CA	9F	0047A	PUSHAB	P.ACW		0914
			69	01	FB	0047E		CALLS	#1, CLISPRESENT		
03	A8		01	50	F0	00481		INSV	R0, #1, #1, QUAL_FLAGS+3		
			2F	50	E9	00487		BLBC	R0, 36\$		
				2C	AE	9F	0048A	PUSHAB	VALUE_DESC		0917
				02D0	CA	9F	0048D	PUSHAB	P.ACY		
		00000000G	00	02	FB	00491		CALLS	#2, CLISGET_VALUE		
				0660	C8	9F	00498	PUSHAB	VERSION_COUNT		0918
				34	AE	DD	0049C	PUSHL	VALUE_DESC+4		0919
			7E	34	AE	3C	0049F	MOVZWL	VALUE_DESC, -(SP)		0918
		00000000G	00	03	FB	004A3		CALLS	#3, LIB\$CVT_DTB		
			57	50	DO	004AA		MOVL	R0, STATUS		
			06	57	E9	004AD		BLBC	STATUS, 35\$		0921
				0660	C8	D5	004B0	TSTL	VERSION_COUNT		
					03	14	004B4	BGTR	36\$		
				00D1	31	004B6	35\$:	BRW	40\$		
				02E0	CA	9F	004B9	36\$:	PUSHAB	P.ADA	0928
			69	01	FB	004BD		CALLS	#1, CLISPRESENT		
03	A8		02	50	F0	004C0		INSV	R0, #2, #1, QUAL_FLAGS+3		

	03		50	E8	004C6	BLBS	R0, 37\$		
			0100	31	004C9	BRW	43\$		
		2C	AE	9F	004CC	37\$: PUSHAB	VALUE_DESC		0931
00000000G	00	02F8	CA	9F	004CF	PUSHAB	P.ADC		
			02	FB	004D3	CALLS	#2, CLISGET_VALUE		
		0814	C8	9F	004DA	PUSHAB	DISPLAY_WIDTH		0932
			34	AE	DD	004DE	PUSHL	VALUE_DESC+4	0933
00000000G	7E		34	AE	3C	004E1	MOVZWL	VALUE_DESC, -(SP)	0932
	00		03	FB	004E5	CALLS	#3, LIB\$CVT_DTB		
	57		50	DD	004EC	MOVL	R0, STATUS		
	C4		57	E9	004EF	BLBC	STATUS, 35\$		0935
		0814	C8	D5	004F2	TSTL	DISPLAY_WIDTH		
			BE	19	004F6	BLSS	35\$		
		2C	AE	9F	004F8	PUSHAB	VALUE_DESC		0941
00000000G	00	0310	CA	9F	004F8	PUSHAB	P.ADE		
			02	FB	004FF	CALLS	#2, CLISGET_VALUE		
		0818	C8	9F	00506	PUSHAB	FILENAME_WIDTH		0942
			34	AE	DD	0050A	PUSHL	VALUE_DESC+4	0943
00000000G	7E		34	AE	3C	0050D	MOVZWL	VALUE_DESC, -(SP)	0942
	00		03	FB	00511	CALLS	#3, LIB\$CVT_DTB		
	57		50	DD	00518	MOVL	R0, STATUS		
	6C		57	E9	0051B	BLBC	STATUS, 40\$		0945
		0818	C8	D5	0051E	TSTL	FILENAME_WIDTH		
			66	19	00522	BLSS	40\$		
			05	12	00524	BNEQ	38\$		0951
0818	C8		13	DD	00526	MOVL	#19, FILENAME_WIDTH		
		2C	AE	9F	0052B	38\$: PUSHAB	VALUE_DESC		0952
00000000G	00	0324	CA	9F	0052E	PUSHAB	P.ADG		
			02	FB	00532	CALLS	#2, CLISGET_VALUE		
		081C	C8	9F	00539	PUSHAB	OWNER_WIDTH		0955
			34	AE	DD	0053D	PUSHL	VALUE_DESC+4	0954
00000000G	7E		34	AE	3C	00540	MOVZWL	VALUE_DESC, -(SP)	0953
	00		03	FB	00544	CALLS	#3, LIB\$CVT_DTB		
	57		50	DD	0054B	MOVL	R0, STATUS		
	39		57	E9	0054E	BLBC	STATUS, 40\$		0956
		081C	C8	D5	00551	TSTL	OWNER_WIDTH		
			33	19	00555	BLSS	40\$		
			05	12	00557	BNEQ	39\$		0962
081C	C8		14	DD	00559	MOVL	#20, OWNER_WIDTH		
		2C	AE	9F	0055F	39\$: PUSHAB	VALUE_DESC		0963
00000000G	00	0338	CA	9F	00561	PUSHAB	P.ADI		
			02	FB	00565	CALLS	#2, CLISGET_VALUE		
		0820	C8	9F	0056C	PUSHAB	SIZE_WIDTH		0964
			34	AE	DD	00570	PUSHL	VALUE_DESC+4	0965
00000000G	7E		34	AE	3C	00573	MOVZWL	VALUE_DESC, -(SP)	0964
	00		03	FB	00577	CALLS	#3, LIB\$CVT_DTB		
	57		50	DD	0057E	MOVL	R0, STATUS		
	06		57	E9	00581	BLBC	STATUS, 40\$		0967
		0820	C8	D5	00584	TSTL	SIZE_WIDTH		
			3B	18	00588	BGEQ	42\$		
00000000G	00	0830	C8	9F	0058A	40\$: PUSHAB	OUTPUT_RAB		0970
			01	FB	0058E	CALLS	#1, SYSSFLUSH		
00000000G	00	0830	C8	9F	00595	PUSHAB	OUTPUT_RAB		
			01	FB	00599	CALLS	#1, SYSSWAIT		
		2C	AE	9F	005A0	PUSHAB	VALUE_DESC		
			01	DD	005A3	PUSHL	#1		
		007910FC	8F	DD	005A5	PUSHL	#7934204		

04	14	AB	00000000G	00	03	FB	005AB	CALLS	#3, LIBSSIGNAL	
				03	00	ED	005B2	CMPZV	#0, #3, WORST_ERROR, #4	
					08	18	005B8	BGEQ	41\$	
			14	AB	107910FL	8F	005BA	MOVL	#276369660, WORST_ERROR	
						038C	31 005C2	41\$:	BRW	86\$
							05 12 005C5	42\$:	BNEQ	43\$
			0820	C8		06	005C7	43\$:	MOVL	#6, SIZE_WIDTH
						5B	005CC		PUSHL	R11
			00000000G	00	01	FB	005CE		CALLS	#1, SYSSCREATE
				57	50	DO	005D5		MOVL	R0, STATUS
				11	57	E9	005D8		BLBC	STATUS, 44\$
					0830	C8	005DB		PUSHAB	OUTPUT_RAB
			00000000G	00	01	FB	005DF		CALLS	#1, SYSSCONNECT
				57	50	DO	005E6		MOVL	R0, STATUS
				0B	57	E8	005E9		BLBS	STATUS, 45\$
						5B	005EC	44\$:	PUSHL	R11
					007910A4	8F	005EE		PUSHL	#7934116
						0355	31 005F4		BRW	85\$
						02	E1 005F7	45\$:	BBC	#2, OUTPUT_FAB+64, 46\$
1C		3E		40	AB	00	2C 005FC		MOVCS	#0, (SP), #0, #28, GETDVI_ARGS
		00					AE 00601			
				10	AE	00040004	8F DO 00603		MOVL	#262148, GETDVI_ARGS
				14	AE	00040004	AE 9E 0060B		MOVAB	INDEV_CLASS, GETDVI_ARGS+4
				1C	AE	00080004	8F DO 00610		MOVL	#524292, GETDVI_ARGS+12
				20	AE	00080004	AE 9E 00618		MOVAB	INDEV_BUFSIZ, GETDVI_ARGS+16
							7E 7C 0061D		CLRQ	-(SP)
							7E 7C 0061F		CLRQ	-(SP)
				20	AE	9F 00621			PUSHAB	GETDVI_ARGS
					034C	CA	9F 00624		PUSHAB	P.ADK
						7E	7C 00628		CLRQ	-(SP)
			00000000G	00	08	FB	0062A		CALLS	#8, SYSSGETDVI
				57	50	DO	00631		MOVL	R0, STATUS
				03	57	E8	00634		BLBS	STATUS, 46\$
					01CA	31	00637		BRW	71\$
					0814	C8	D5 0063A	46\$:	TSTL	DISPLAY_WIDTH
						15	12 0063E		BNEQ	48\$
			00000042	8F	04	AE	D1 00640		CMP	INDEV_CLASS, #66
						05	13 00648		BEQ	47\$
				08	AE	84 8F 9A 0064A		MOVZBL	#132, INDEV_BUFSIZ	
				0814	C8	08	AE DO 0064F	47\$:	MOVL	INDEV_BUFSIZ, DISPLAY_WIDTH
							03 E0 00655	48\$:	BBS	#3, QUAL_FLAGS, 49\$
				18	01	AB	05 E0 00659		BBS	#5, QUAL_FLAGS+1, 49\$
				13			01 AB 95 0065E		TSTB	QUAL_FLAGS+1
							0E 19 00661		BLSS	49\$
				09	02	AB	03 E0 00663		BBS	#3, QUAL_FLAGS+2, 49\$
						05	01 AB EB 00668		BLBS	QUAL_FLAGS+1, 49\$
				09	01	AB	03 E0 0066C		BBS	#3, QUAL_FLAGS+1, 50\$
							03 AB 95 00671	49\$:	TSTB	QUAL_FLAGS+3
							04 18 00674		BGEQ	50\$
				08	AB	01	DO 00676		MOVL	#1, COLUMN_COUNT
					1D	04	AB E9 0067A	50\$:	BLBC	QUAL_FLAGS+4, 52\$
					0668	C8	D5 0067E		TSTL	FIRST_XAB
							0C 12 00682		BNEQ	51\$
							56 C8 9E 00684		MOVAB	INFO_XABFHC, XAB_PTR
			0668	C8	0738	56	DO 00689		MOVL	XAB_PTR, FIRST_XAB
							0B 11 0068E		BRB	52\$
				04	A6	0738	C8 9E 00690	51\$:	MOVAB	INFO_XABFHC, 4(XAB_PTR)

1D	04	56 A8	0738	C8	9E	00696		MOVAB	INFO_XABFHC, XAB_PTR		
				01	E1	00698	52\$:	BBC	#1, QUAL_FLAGS+4, 54\$		1038
			0668	C8	D5	006A0		TSTL	FIRST_XAB		1041
				0C	12	006A4		BNEQ	53\$		
		56	070C	C8	9E	006A6		MOVAB	INFO_XABDAT, XAB_PTR		1042
	0668	C8		56	D0	006AB		MOVL	XAB_PTR, FIRST_XAB		
				0B	11	006B0		BRB	54\$		
		04	070C	C8	9E	006B2	53\$:	MOVAB	INFO_XABDAT, 4(XAB_PTR)		1043
		56	070C	C8	9E	006B8		MOVAB	INFO_XABDAT, XAB_PTR		
1D	04	A8		02	E1	006BD	54\$:	BBC	#2, QUAL_FLAGS+4, 56\$		1045
			0668	C8	D5	006C2		TSTL	FIRST_XAB		1048
				0C	12	006C6		BNEQ	55\$		
		56	06B4	C8	9E	006C8		MOVAB	INFO_XABPRO, XAB_PTR		1049
	0668	C8		56	D0	006CD		MOVL	XAB_PTR, FIRST_XAB		
				0B	11	006D2		BRB	56\$		
		04	06B4	C8	9E	006D4	55\$:	MOVAB	INFO_XABPRO, 4(XAB_PTR)		1050
		56	06B4	C8	9E	006DA		MOVAB	INFO_XABPRO, XAB_PTR		
1D	04	A8		03	E1	006DF	56\$:	BBC	#3, QUAL_FLAGS+4, 58\$		1052
			0668	C8	D5	006E4		TSTL	FIRST_XAB		1055
				0C	12	006E8		BNEQ	57\$		
		56	06A8	C8	9E	006EA		MOVAB	INFO_XABSUM, XAB_PTR		1056
	0668	C8		56	D0	006EF		MOVL	XAB_PTR, FIRST_XAB		
				0B	11	006F4		BRB	58\$		
		04	06A8	C8	9E	006F6	57\$:	MOVAB	INFO_XABSUM, 4(XAB_PTR)		1057
		56	06A8	C8	9E	006FC		MOVAB	INFO_XABSUM, XAB_PTR		
45	04	A8		04	E1	00701	58\$:	BBC	#4, QUAL_FLAGS+4, 61\$		1059
			0668	C8	D5	00706		TSTL	FIRST_XAB		1062
				0C	12	0070A		BNEQ	59\$		
		56	066C	C8	9E	0070C		MOVAB	INFO_XABJNL, XAB_PTR		1063
	0668	C8		56	D0	00711		MOVL	XAB_PTR, FIRST_XAB		
				0B	11	00716		BRB	60\$		
		04	066C	C8	9E	00718	59\$:	MOVAB	INFO_XABJNL, 4(XAB_PTR)		1064
		56	066C	C8	9E	0071E		MOVAB	INFO_XABJNL, XAB_PTR		
		50	1C	A8	D0	00723	60\$:	MOVL	DISPLAY_BLOCK, R0		1065
	0684	C8	0199	C0	9E	00727		MOVAB	409(R0), INFO_XABJNL+24		
	0680	C8		10	90	0072E		MOVAB	#16, INFO_XABJNL+20		1066
	067C	C8	01AA	C0	9E	00733		MOVAB	426(R0), INFO_XABJNL+16		1067
	0678	C8		10	90	0073A		MOVAB	#16, INFO_XABJNL+12		1068
	068C	C8	018B	C0	9E	0073F		MOVAB	443(R0), INFO_XABJNL+32		1069
	0688	C8		10	90	00746		MOVAB	#16, INFO_XABJNL+28		1070
50	10	A8	0818	C8	C1	0074B	61\$:	ADDL3	FILENAME_WIDTH, COLUMN_WIDTH, R0		1078
	10	A8	01	A0	9E	00752		MOVAB	1(R0), COLUMN_WIDTH		
0C	01	A8		05	E1	00757		BBC	#5, QUAL_FLAGS+1, 62\$		1079
50	10	A8	081C	C8	C1	0075C		ADDL3	OWNER_WIDTH, COLUMN_WIDTH, R0		
	10	A8	02	A0	9E	00763		MOVAB	2(R0), COLUMN_WIDTH		
22	02	A8		03	E1	00768	62\$:	BBC	#3, QUAL_FLAGS+2, 64\$		1080
11	02	A8		04	E1	0076D		BBC	#4, QUAL_FLAGS+2, 63\$		1083
		50	0820	C8	D0	00772		MOVL	SIZE_WIDTH, R0		1084
		10	10	B840	3E	00777		MOVAB	@COLUMN_WIDTH[R0], COLUMN_WIDTH		
		10		02	C0	0077D		ADDL2	#2, COLUMN_WIDTH		
				0C	11	00781		BRB	64\$		
50	10	A8	0820	C8	C1	00783	63\$:	ADDL3	SIZE_WIDTH, COLUMN_WIDTH, R0		1085
	10	A8	02	A0	9E	0078A		MOVAB	2(R0), COLUMN_WIDTH		
1F		68		04	E0	0078F	64\$:	BBS	#4, QUAL_FLAGS, 65\$		1087
1B		68		06	E0	00793		BBS	#6, QUAL_FLAGS, 65\$		
17		68		05	E0	00797		BBS	#5, QUAL_FLAGS, 65\$		1088
				68	95	0079B		TSTB	QUAL_FLAGS		

OE	01	AB	01	13	19	0079D	BLSS	65\$		
				05	E0	0079F	BBS	#5, QUAL_FLAGS+1, 65\$		1089
				A8	95	007A4	TSTB	QUAL_FLAGS+1		
04	02	AB	01	09	19	007A7	BLSS	65\$		
		19		03	E0	007A9	BBS	#3, QUAL_FLAGS+2 65\$		1090
	10	AB		A8	E9	007AE	BLBC	QUAL_FLAGS+1, 66\$		
51	0814	C8		04	C0	007B2	ADDL2	#4, COLUMN_WIDTH		1093
		51		04	C1	007B6	ADDL3	#4, DISPLAY_WIDTH, R1		1094
		50		A8	C6	007BC	DIVL2	COLUMN_WIDTH, R1		
		51		A8	D0	007C0	MOVL	COLUMN_COUNT, R0		
				50	D1	007C4	CMPL	R0, R1		
				12	1A	007C7	BGTRU	67\$		
				13	11	007C9	BRB	68\$		
51	0814	C8	10	A8	C7	007CB	DIVL3	COLUMN_WIDTH, DISPLAY_WIDTH, R1		1096
		50	08	A8	D0	007D2	MOVL	COLUMN_COUNT, R0		
		51		50	D1	007D6	CMPL	R0, R1		
		50		03	1B	007D9	BLEQU	68\$		
	08	AB		51	D0	007DB	MOVL	R1, R0		
		04		50	D0	007DE	MOVL	R0, COLUMN_COUNT		
		08		03	15	007E2	BLEQ	69\$		1097
		08		68	E9	007E4	BLBC	QUAL_FLAGS, 70\$		
		04		01	D0	007E7	MOVL	#1, COLUMN_COUNT		
		AE	18	A8	9F	007EB	PUSHAB	CMN QUAL_CTX		1102
			01FE	8F	3C	007EE	MOVZWL	#510, 4(SP)		1108
			04	AE	9F	007F4	PUSHAB	4(SP)		1102
00000000G	00			02	FB	007F7	CALLS	#2, LIB\$QUAL_FILE_PARSE		
				50	D0	007FE	MOVL	R0, STATUS		
				57	E8	00801	BLBS	STATUS, 74\$		1111
				37	C8	9F	00804	PUSHAB	OUTPUT_RAB	1114
00000000G	00		0830	01	FB	00808	CALLS	#1, SYS\$FLUSH		
				0830	C8	9F	0080F	PUSHAB	OUTPUT_RAB	
00000000G	00			01	FB	00813	CALLS	#1, SYS\$WAIT		
				57	DD	0081A	PUSHL	STATUS		
00000000G	00			01	FB	0081C	CALLS	#1, LIB\$SIGNAL		
				57	93	00823	BITB	STATUS, #7		
				03	12	00826	BNEQ	73\$		
				0126	31	00828	BRW	86\$		
50		57		00	E0	0082B	EXTZV	#0, #3, STATUS, R0		
50	14	AB		00	ED	00830	CMPZV	#0, #3, WORST_ERROR, R0		
				F0	1B	00836	BGEQ	72\$		
				F88B	31	00838	BRW	2\$		
				34	AE	9F	0083B	PUSHAB	FILE_DESC	1118
				035C	CA	9F	0083E	PUSHAB	P.ADM	
00000000G	00			02	FB	00842	CALLS	#2, CLISGET_VALUE		
	DC	AD		38	AE	D0	00849	MOVL	FILE_DESC+4, INPUT_FAB+44	1119
	E4	AC		34	AE	90	0084E	MOVW	FILE_DESC, INPUT_FAB+52	1120
03	01	AB		01	E0	00853	BBS	#1, QUAL_FLAGS+1, 75\$		1126
		1B		68	E9	00858	BLBC	QUAL_FLAGS, 76\$		1127
				FC	AB	9F	0085B	PUSHAB	FORMAT_ACL_ADDR	1129
				0388	CA	9F	0085E	PUSHAB	P.ADQ	1130
				0370	CA	9F	00862	PUSHAB	P.ADO	1129
00000000G	00			03	FB	00866	CALLS	#3, LIB\$FIND_IMAGE_SYMBOL		
				50	D0	0086D	MOVL	R0, STATUS		
				57	E8	00870	BLBS	STATUS, 75\$		1131
				F81F	31	00873	BRW	1\$		
				04	AB	E8	00876	BLBS	QUAL_FLAGS+4, 77\$	1147
OF	04	AB		01	E0	0087A	BBS	#1, QUAL_FLAGS+4, 77\$		

0A	04	A8	02	E0	0087F	BBS	#2, QUAL_FLAGS+4, 77\$: 1148		
05	04	A8	03	E0	00884	BBS	#3, QUAL_FLAGS+4, 77\$: 1149		
0C	04	A8	04	E1	00889	BBC	#4, QUAL_FLAGS+4, 78\$: 1152		
	FF58	CD	8F	88	0088E	77\$:	BISB2	#64, INPUT_NAM+8	: 1153	
	D4	AD	0668	C8	D0	00894	MOVL	FIRST_XAB, INPUT_FAB+36	: 1156	
			0C	AE	9F	0089A	78\$:	PUSHAB	SCAN_CONTEXT	: 1161
			0000V	CF	9F	0089D	PUSHAB	DIR\$INPUT_ERROR	: 1163	
			0000G	CF	9F	008A1	PUSHAB	DIR\$GET_INFO	: 1164	
			B0	AD	9F	008A5	PUSHAB	INPUT_FAB	: 1165	
00000000G	00		B0	04	FB	008AB	CALLS	#4, LIB\$FILE_SCAN	: 1166	
				AD	9F	008AF	PUSHAB	INPUT_FAB	: 1167	
0000V	CF			01	FB	008B2	CALLS	#1, DIR\$GET_FILE	: 1172	
	BC			50	E8	008B7	BLBS	RO, 76\$: 1175	
			34	A8	B5	008BA	TSTW	LINE_DESC	: 1177	
			34	0A	13	008BD	BEQL	79\$: 1180	
				A8	9F	008BF	PUSHAB	LINE_DESC	: 1181	
				7E	D4	008C2	CLRL	-(SP)	: 1183	
0000V	CF		0444	02	FB	008C4	79\$:	CALLS	#2, DIR\$OUTPUT	: 1185
				C8	D5	008C9	TSTL	TOTAL_FILES	: 1186	
				05	13	008CD	BEQL	80\$: 1187	
0000G	CF		0454	00	FB	008CF	80\$:	CALLS	#0, DIR\$TOTAL	: 1188
	01			C8	D1	008D4	81\$:	CMP	GRAND_DIRS, #1	: 1189
				05	14	008D9	BGTR	81\$: 1190	
05	01	A8		02	E1	008DB	BBC	#2, QUAL_FLAGS+1, 82\$: 1191	
	0000G	CF		00	FB	008E0	81\$:	CALLS	#0, DIR\$GRAND_TOTAL	: 1192
		4C	14	A8	E9	008E5	82\$:	BLBC	WORST_ERROR, 84\$: 1193
47	04	A8		05	E0	008E9	BBS	#5, QUAL_FLAGS+4, 84\$: 1194	
			0830	C8	9F	008EE	PUSHAB	OUTPUT_RAB	: 1195	
00000000G	00			01	FB	008F2	CALLS	#1, SYS\$FLUSH	: 1196	
			0830	C8	9F	008F9	PUSHAB	OUTPUT_RAB	: 1197	
00000000G	00			01	FB	008FD	CALLS	#1, SYS\$WAIT	: 1198	
			00000000G	8F	DD	00904	PUSHL	#DIRS_NOFILES	: 1199	
00000000G	00		00000000*	01	FB	0090A	CALLS	#1, LIB\$SIGNAL	: 1200	
				8F	D5	00911	TSTL	#<DIRS_NOFILES&7>	: 1201	
				14	13	00917	BEQL	83\$: 1202	
00000000*	8F	14	A8	00	ED	00919	CMPZV	#0, #3, WORST_ERROR, #<DIRS_NOFILES&7>	: 1203	
				08	18	00923	BGEQ	83\$: 1204	
	14	A8	00000000*	8F	D0	00925	MOVL	#<DIRS_NOFILES!268435456>, WORST_ERROR	: 1205	
	14	A8	10018290	8F	D0	0092D	83\$:	MOVL	#268534416, WORST_ERROR	: 1206
				5B	DD	00935	84\$:	PUSHL	R11	: 1207
00000000G	00			01	FB	00937	CALLS	#1, SYS\$CLOSE	: 1208	
	57			50	D0	0093E	MOVL	RO, STATUS	: 1209	
	0D			57	E8	00941	BLBS	STATUS, 86\$: 1210	
			0079105A	5B	DD	00944	PUSHL	R11	: 1211	
				8F	DD	00946	PUSHL	#7934042	: 1212	
0000V	CF			02	FB	0094C	85\$:	CALLS	#2, DIR\$FILE_ERROR	: 1213
	50		14	A8	D0	00951	86\$:	MOVL	WORST_ERROR, RO	: 1214
				04	00955	RET			: 1215	

: Routine Size: 2390 bytes, Routine Base: \$CODE\$ + 0000

```

789 1186 1 ROUTINE DIR$GET_FILE (FILE_FAB) =
790 1187 1
791 1188 1 |++
792 1189 1
793 1190 1 | FUNCTIONAL DESCRIPTION:
794 1191 1
795 1192 1 |     This routine gets the next file specification in the command line.
796 1193 1 |     If there are no more files, the routine returns zero.  Otherwise,
797 1194 1 |     the file specification is placed in the specified FAB for later
798 1195 1 |     parsing and searching.
799 1196 1
800 1197 1 | CALLING SEQUENCE:
801 1198 1 |     DIR$GET_FILE (ARG1)
802 1199 1
803 1200 1 | INPUT PARAMETERS:
804 1201 1 |     ARG1. address of the FAB into which the file spec is placed
805 1202 1
806 1203 1 | IMPLICIT INPUTS:
807 1204 1 |     none
808 1205 1
809 1206 1 | OUTPUT PARAMETERS:
810 1207 1 |     none
811 1208 1
812 1209 1 | IMPLICIT OUTPUTS:
813 1210 1 |     none
814 1211 1
815 1212 1 | ROUTINE VALUE:
816 1213 1 |     1 if a file specification was found
817 1214 1 |     0 otherwise
818 1215 1
819 1216 1 | SIDE EFFECTS:
820 1217 1 |     The retrieved file specification is placed into the specified
821 1218 1 |     FAB for later parsing.
822 1219 1
823 1220 1 | --
824 1221 1
825 1222 2 BEGIN
826 1223 2
827 1224 2 MAP
828 1225 2     FILE_FAB      : REF $BBLOCK;           ! FAB address
829 1226 2
830 1227 2 LOCAL
831 1228 2     FILE_DESC     : $BBLOCK [DSC$C_S_BLN],   ! File name descr
832 1229 2     SCAN_FLAGS    : $BBLOCK [4];           ! $FILESCAN flags
833 1230 2
834 1231 2 ! Initialise needed variables.
835 1232 2
836 1233 2 CH$FILL (0, DSC$C_S_BLN, FILE_DESC);
837 1234 2 FILE_DESC[DSC$B_CLASS] = DSC$R_CLASS_D;
838 1235 2
839 1236 2 ! If there are no more file specifications, return with zero.
840 1237 2
841 1238 2 IF NOT CL$GET_VALUE ($DESCRIPTOR ('INPUT'), FILE_DESC) THEN RETURN 0;
842 1239 2
843 1240 2 ! Otherwise, fill in the appropriate fields in the FAB.
844 1241 2
845 1242 2 FILE_FAB[FAB$FNA] = .FILE_DESC[DSC$A_POINTER];

```

```

: 846 1243 2 FILE_FAB[FAB$B_FNS] = .FILE_DESC[DSC$W_LENGTH];
: 847 1244 2
: 848 1245 2 ! Determine whether or not the new spec is to get a new heading.
: 849 1246 2
: 850 1247 2 SCAN_FLAGS = 0;
: 851 1248 2 $FILESKAN (SRC$TR = FILE_DESC, FLDFLAGS = SCAN_FLAGS);
: 852 1249 2 IF .SCAN_FLAGS[FSCN$V_NODE] OR .SCAN_FLAGS[FSCN$V_DEVICE]
: 853 1250 2 OR .SCAN_FLAGS[FSCN$V_ROOT] OR .SCAN_FLAGS[FSCN$V_DIRECTORY]
: 854 1251 2 THEN
: 855 1252 2 BEGIN
: 856 1253 2     VERSION_INDEX = 0;
: 857 1254 2     PREV_DIR_LEN = PREV_FILE_LEN = 0;
: 858 1255 2 END;
: 859 1256 2
: 860 1257 2 RETURN 1;
: 861 1258 2
: 862 1259 1 END;

```

! End of routine DIR\$GET_FILE

```

.PSECT $PLITS,NOWRT,NOEXE,2
54 55 50 4E 49 00390 P.ADT: .ASCII \INPUT\
00000005 00395 .BLKB 3
00000000' 00398 P.ADS: .LONG 5
00000000' 0039C .ADDRESS P.ADT
.EXTRN SYSS$FILESKAN
.PSECT $CODE$,NOWRT,2
007C 00000 DIR$GET
.E:
08 00 56 00000000' EF 9E 00002 .WORD Save R2,R3,R4,R5,R6 : 1186
5E 0C C2 00009 MOVAB VERSION_INDEX, R6
6E 00 2C 0000C SUBL2 #12, SP
07 AE 04 AE 00011 MOVCS #0, (SP), #0, #8, FILE_DESC : 1233
04 AE 02 90 00013 MOVB #2, FILE_DESC+3 : 1234
04 AE 9F 00017 PUSHAB FILE_DESC : 1238
0000' CF 9F 0001A PUSHAB P.ADS
00000000G 00 02 FB 0001E CALLS #2, CLISGET_VALUE
3A 50 E9 00025 BLBC R0, 3$
50 04 AC D0 00028 MOVL FILE_FAB, R0 : 1242
2C A0 08 AE D0 0002C MOVL FILE_DESC+4, 44(R0)
34 A0 04 AE 90 00031 MOVAB FILE_DESC, 52(R0) : 1243
6E D4 00036 CLRL SCAN_FLAGS : 1247
5E DD 00038 PUSHL SP : 1248
7E D4 0003A CLRL -(SP)
00000000G 00 0C AE 9F 0003C PUSHAB FILE_DESC
03 FB 0003F CALLS #3, SYSS$FILESKAN
0C 6E E8 00046 BLBS SCAN_FLAGS, 1$ : 1249
08 6E 01 E0 00049 BBS #1, SCAN_FLAGS, 1$
04 6E 02 E0 0004D BBS #2, SCAN_FLAGS, 1$ : 1250
09 6E 03 E1 00051 BBC #3, SCAN_FLAGS, 2$
66 D4 00055 1$: CLRL VERSION_INDEX : 1253
F8 A6 D4 00057 CLRL PREV_FILE_LEN : 1254
FEF4 C6 D4 0005A CLRL PREV_DIR_LEN

```

DIRECTORY
V04-000

B 1
15-Sep-1984 23:38:58
14-Sep-1984 12:19:31

VAX-11 Bliss-32 V4.0-742
[DIR.SRC]DIRECTORY.B32;1

Page 36
(5)

DIS
V04

50

01	D0	0005E	2\$:	MOVL	#1, R0
	04	00061		RET	
50	D4	00062	3\$:	CLRL	R0
	04	00064		RET	

: 1257
:
: 1259
:

; Routine Size: 101 bytes, Routine Base: \$CODE\$ + 0956

.....

```

: 864 1260 1 GLOBAL ROUTINE DIR$INPUT_ERROR (FILE_FAB) =
: 865 1261 1
: 866 1262 1 :++
: 867 1263 1
: 868 1264 1 FUNCTIONAL DESCRIPTION:
: 869 1265 1
: 870 1266 1 This routine is used to signal errors received on the input file.
: 871 1267 1
: 872 1268 1 CALLING SEQUENCE:
: 873 1269 1 DIR$INPUT_ERROR (ARG1)
: 874 1270 1
: 875 1271 1 INPUT PARAMETERS:
: 876 1272 1 ARG1: address of th. FAB
: 877 1273 1
: 878 1274 1 IMPLICIT INPUTS:
: 879 1275 1 none
: 880 1276 1
: 881 1277 1 OUTPUT PARAMETERS:
: 882 1278 1 none
: 883 1279 1
: 884 1280 1 IMPLICIT OUTPUTS:
: 885 1281 1 none
: 886 1282 1
: 887 1283 1 ROUTINE VALUE:
: 888 1284 1 1
: 889 1285 1
: 890 1286 1 SIDE EFFECTS:
: 891 1287 1 The error is signaled by placing the appropriate message into
: 892 1288 1 the output file.
: 893 1289 1
: 894 1290 1 --
: 895 1291 1
: 896 1292 2 BEGIN
: 897 1293 2
: 898 1294 2 MAP
: 899 1295 2 FILE_FAB : REF $BLOCK; ! FAB address
: 900 1296 2
: 901 1297 2 IF .FILE_FAB[FAB$L STS] NEQ RMSS_FNF
: 902 1298 2 THEN DIR$FILE_ERROR (DIR$_OPENIN, .FILE_FAB);
: 903 1299 2
: 904 1300 2 RETURN 1;
: 905 1301 2
: 906 1302 1 END; ! End of routine DIR$INPUT_ERROR

```

```

00018292 50 04 AC D0 00002 .ENTRY DIR$INPUT_ERROR, Save nothing : 1260
00018292 8F 08 A0 D1 00006 MOVL FILE_FAB, R0 : 1297
0000V CF 0079109A 0D 13 0000E CMPL 8(R0), #98962
01 D0 0001D 1$ 00 DD 00010 BEQL 1$ : 1298
04 00020 02 FB 00018 PUSHL R0
01 D0 0001D 01 DD 00012 PUSHL #7934106
04 00020 01 D0 0001D CALLS #2, DIR$FILE_ERROR : 1300
04 00020 01 D0 0001D MOVL #1, R0 : 1302
04 00020 01 D0 00020 RET

```

DIIRECTORY
V04-000

D 1
15-Sep-1984 23:38:58
14-Sep-1984 12:19:31

VAX-11 Bliss-32 V4.0-742
[DIR.SRC]DIIRECTORY.B32;1

Page 38
(6)

DIS
V04

; Routine Size: 33 bytes, Routine Base: \$CODES + 09BB

.....


```

908 1303 1 GLOBAL ROUTINE DIR$FILE_ERROR (ERROR_CODE, FILE_FAB) =
909 1304 1
910 1305 1 ++
911 1306 1
912 1307 1 FUNCTIONAL DESCRIPTION:
913 1308 1
914 1309 1     This routine is used to signal errors received on files.
915 1310 1
916 1311 1 CALLING SEQUENCE:
917 1312 1     DIR$FILE_ERROR (ARG1, ARG2)
918 1313 1
919 1314 1 INPUT PARAMETERS:
920 1315 1     ARG1: error code
921 1316 1     ARG2: address of the FAB
922 1317 1
923 1318 1 IMPLICIT INPUTS:
924 1319 1     none
925 1320 1
926 1321 1 OUTPUT PARAMETERS:
927 1322 1     none
928 1323 1
929 1324 1 IMPLICIT OUTPUTS:
930 1325 1     none
931 1326 1
932 1327 1 ROUTINE VALUE:
933 1328 1     1
934 1329 1
935 1330 1 SIDE EFFECTS:
936 1331 1     none
937 1332 1
938 1333 1 --
939 1334 1
940 1335 2 BEGIN
941 1336 2
942 1337 2 MAP
943 1338 2     FILE_FAB           : REF $BLOCK;           ! FAB address
944 1339 2
945 1340 2 BIND
946 1341 2     FILE_NAME         = .FILE_FAB[FAB$S_NAME] : $BLOCK;           ! NAME block address
947 1342 2
948 1343 2 LOCAL
949 1344 2     FILE_NAME         : $BLOCK [DSC$S_BLN];       ! Local file name descr
950 1345 2
951 1346 2 CH$FILL (0, DSC$S_BLN, FILE_NAME);
952 1347 2 IF .FILE_NAME[NAM$B_RSL] NEQ 0
953 1348 2 THEN
954 1349 2     BEGIN
955 1350 2         FILE_NAME[DSC$W_LENGTH] = .FILE_NAME[NAM$B_RSL];
956 1351 2         FILE_NAME[DSC$A_POINTER] = .FILE_NAME[NAM$C_RSA];
957 1352 2     END
958 1353 2 ELSE IF .FILE_NAME[NAM$B_ESL] NEQ 0
959 1354 2 THEN
960 1355 2     BEGIN
961 1356 2         FILE_NAME[DSC$W_LENGTH] = .FILE_NAME[NAM$B_ESL];
962 1357 2         FILE_NAME[DSC$A_POINTER] = .FILE_NAME[NAM$C_ESA];
963 1358 2     END
964 1359 2 ELSE

```

```

: 965      1360      3      BEGIN
: 966      1361      3      FILE_NAME[DSC$W_LENGTH] = .FILE_FAB[FAB$B_FNS];
: 967      1362      3      FILE_NAME[DSC$A_POINTER] = .FILE_FAB[FAB$C_FNA];
: 968      1363      3      END;
: 969      1364      3
: 970      P 1365      3      SIGNAL (.ERROR_CODE, 1, FILE_NAME, .FILE_FAB[FAB$L_STS],
: 971      1366      3      .FILE_FAB[FAB$L_STV]);
: 972      1367      3
: 973      1368      3      IF .WORST_ERROR EQL (.ERROR_CODE OR STS$M_INHIB_MSG)
: 974      1369      3      THEN WORST_ERROR = .FILE_FAB[FAB$L_STS] OR STS$M_INHIB_MSG;
: 975      1370      3
: 976      1371      3      RETURN 1;
: 977      1372      3
: 978      1373      1      END;

```

! End of routine DIR\$FILE_ERROR

				01FC 00000	.ENTRY DIR\$FILE_ERROR, Save R2,R3,R4,R5,R6,R7,R8	1303
		58 00000000'	EF 9E 00002		MOVAB WORST_ERROR, R8	
		5E	08 C2 00009		SUBL2 #8, SP	
		57 08	AC D0 0000C		MOVL FILE_FAB, R7	1341
		56 28	A7 D0 00010		MOVL 40(R7), R6	
08	00	6E	00 2C 00014		MOVCS #0, (SP), #0, #8, FILE_NAME	1346
			6E 00019			
			03 A6 95 0001A		TSTB 3(R6)	1347
			08 13 0001D		BEQL 1\$	
		6E 03	A6 9B 0001F		MOVZBW 3(R6), FILE_NAME	1350
	04	AE 04	A6 D0 00023		MOVL 4(R6), FILE_NAME+4	1351
			19 11 00028		BRB 3\$	1347
			08 A6 95 0002A	1\$:	TSTB 11(R6)	1353
			08 13 0002D		BEQL 2\$	
		6E 08	A6 9B 0002F		MOVZBW 11(R6), FILE_NAME	1356
	04	AE 0C	A6 D0 00033		MOVL 12(R6), FILE_NAME+4	1357
			09 11 00038		BRB 3\$	1353
		6E 34	A7 9B 0003A	2\$:	MOVZBW 52(R7), FILE_NAME	1361
	04	AE 2C	A7 D0 0003E	3\$:	MOVL 44(R7), FILE_NAME+4	1362
			081C C8 9F 00043		PUSHAB OUTPUT_RAB	1366
		00000000G 00	01 FB 00047		CALLS #1, SYSSFLUSH	
			081C C8 9F 0004E		PUSHAB OUTPUT_RAB	
		00000000G 00	01 FB 00052		CALLS #1, SYSSWAIT	
		7E	08 A7 7D 00059		MOVQ 8(R7), -(SP)	
			08 AE 9F 0005D		PUSHAB FILE_NAME	
			01 DD 0C060		PUSHL #1	
		52 04	AC D0 00062		MOVL ERROR_CODE, R2	
			52 DD 00066		PUSHL R2	
		00000000G 00	05 FB 00068		CALLS #5, LIB\$SIGNAL	
		07	52 93 0006F		BITB R2, #7	
			14 13 00072		BEQL 4\$	
50	52	03	00 EF 00074		EXTZV #0, #3, R2, R0	
50	68	03	00 ED 00079		CMPZV #0, #3, WORST_ERROR, R0	
			08 18 0007E		BGEQ 4\$	
		68 52 10000000	8F C9 00080		BISL3 #268435456, R2, WORST_ERROR	
52	01	1C	01 F0 00088	4\$:	INSV #1, #28, #1, R2	1368
		52	68 D1 0008D		CMLP WORST_ERROR, R2	
			09 12 00090		BNEQ 5\$	

DIRECTORY
V04-000

G 1
15-Sep-1984 23:38:58
14-Sep-1984 12:19:31

VAX-11 Bliss-32 V4.0-742
[DIR.SRC]DIRECTORY.B32;1

Page 41
(7)

D1
V0

68	08	A7 10000000	8F C9 00092	BISL3	#268435456, 8(R7), WORST_ERROR	: 1369
		50	01 D0 0009B 58:	MOVL	#1, R0	: 1371
			04 C009E	RET		: 1373

; Routine Size: 159 bytes, Routine Base: \$CODE\$ + 09DC

.....

```

980 1374 1 GLOBAL ROUTINE DIR$OUTPUT (MESSAGE_CODE, CONTROL_STRING, ARGS) =
981 1375 1
982 1376 1 :++
983 1377 1
984 1378 1 FUNCTIONAL DESCRIPTION:
985 1379 1
986 1380 1 This routine accepts, as input, an $FAO control string and any
987 1381 1 arguments to be formatted by the control string. The formatted
988 1382 1 line is then written to the desired output file.
989 1383 1
990 1384 1 CALLING SEQUENCE:
991 1385 1 DIR$OUTPUT (ARG1, ARG2, ..., ARGn)
992 1386 1
993 1387 1 INPUT PARAMETERS:
994 1388 1 ARG1: message code for the text to display
995 1389 1 ARG2: address of the $FAO control string
996 1390 1 ARG3 - ARGn: arguments to be formatted
997 1391 1
998 1392 1 IMPLICIT INPUTS:
999 1393 1 none
1000 1394 1
1001 1395 1 OUTPUT PARAMETERS:
1002 1396 1 none
1003 1397 1
1004 1398 1 IMPLICIT OUTPUTS:
1005 1399 1 none
1006 1400 1
1007 1401 1 ROUTINE VALUE:
1008 1402 1 1
1009 1403 1
1010 1404 1 SIDE EFFECTS:
1011 1405 1 none
1012 1406 1
1013 1407 1 --
1014 1408 1
1015 1409 2 BEGIN
1016 1410 2
1017 1411 2 MAP
1018 1412 2 CONTROL_STRING : REF $BLOCK; ! Address of the control string
1019 1413 2
1020 1414 2 LOCAL
1021 1415 2 FAO_CTL_STRING : REF $BLOCK, ! Addr of $FAO control string
1022 1416 2 MESSAGE_DESC : $BLOCK [DSC$S_BLN], ! Message text descr
1023 1417 2 MESSAGE_TEXT : VECTOR [256, BYTE], ! Message text
1024 1418 2 STATUS; ! Local routine exit status
1025 1419 2
1026 1420 2 ! If there is a message code present, get the message text via a $GETMSG.
1027 1421 2 ! Otherwise, use the descriptor supplied.
1028 1422 2
1029 1423 2 IF .MESSAGE_CODE NEQ 0
1030 1424 2 THEN
1031 1425 2 BEGIN
1032 1426 2 CH$FILL (0, DSC$S_BLN, MESSAGE_DESC);
1033 1427 2 MESSAGE_DESC[DSC$W_LENGTH] = 256;
1034 1428 2 MESSAGE_DESC[DSC$A_POINTER] = MESSAGE_TEXT;
1035 1429 2 $GETMSG (MSGID = .MESSAGE_CODE,
P 1430 2 MSGLEN = MESSAGE_DESC[DSC$W_LENGTH],

```

```

1037 P 1431          BUFADR = MESSAGE_DESC,
1038   1432          FLAGS = 1);
1039   1433          FAO_CTL_STRING = MESSAGE_DESC;
1040   1434          END
1041   1435 ELSE FAO_CTL_STRING = .CONTROL_STRING;
1042   1436
1043   1437 ! Format the line.
1044   1438
1045   1439 IF .FAO_CTL_STRING NEQA LINE_DESC
1046   1440 THEN
1047   1441 BEGIN
1048   1442   CH$FILL (0, DSC$C_S_BLN, LINE_DESC);
1049   1443   LINE_DESC[DSC$W_LENGTH] = 1024;
1050   1444   LINE_DESC[DSC$A_POINTER] = LINE_BUFFER;
1051   1445
1052   1446   $FAOL (CTRSTR = .FAO_CTL_STRING,
1053   1447           OUTLEN = LINE_DESC,
1054   1448           OUTBUF = LINE_DESC,
1055   1449           PRMLST = ARGST);
1056   1450
1057   1451   OUTPUT_RAB[RAB$L_RBF] = .LINE_DESC[DSC$A_POINTER];
1058   1452   OUTPUT_RAB[RAB$W_RSZ] = .LINE_DESC[DSC$W_LENGTH];
1059   1453   END
1060   1454 ELSE
1061   1455 BEGIN
1062   1456   OUTPUT_RAB[RAB$L_RBF] = .FAO_CTL_STRING[DSC$A_POINTER];
1063   1457   OUTPUT_RAB[RAB$W_RSZ] = .FAO_CTL_STRING[DSC$W_LENGTH];
1064   1458   END;
1065   1459
1066   1460 STATUS = $RMS_PUT (RAB = OUTPUT_RAB);
1067   1461 IF NOT .STATUS THEN DIR$FILE_ERROR (DIR$WRITEERR, OUTPUT_RAB);
1068   1462
1069   1463 LINE_DESC[DSC$W_LENGTH] = 0;
1070   1464
1071   1465 RETURN 1;
1072   1466
1073   1467 1 END;

```

! End of routine DIR\$OUTPUT

						.EXTRN	SYSSGETMSG, SYSSFAOL			
						.EXTRN	SYSSPUT			
						.ENTRY	DIR\$OUTPUT, Save R2,R3,R4,R5,R6,R7	: 1374		
		57	00000000'	EF	9E	00002	MOVAB	LINE_DESC, R7		
		5E	FEF8	CE	9E	00009	MOVAB	-264(SP), SP		
			04	AC	D5	0000E	TSTL	MESSAGE_CODE	: 1423	
				2A	13	00011	BEQL	1\$		
08	00	6E		00	2C	00013	MOVCS	#0, (SP), #0, #8, MESSAGE_DESC	: 1426	
			F8	AD		00018				
		F8	AD	0100	8F	80	0001A	MOVW	#256, MESSAGE_DESC	: 1427
		FC	AD		6E	9E	00020	MOVAB	MESSAGE_TEXT, MESSAGE_DESC+4	: 1428
			7E		01	7D	00024	MOVQ	#1, -(SP)	: 1432
				F8	AD	9F	00027	PUSHAB	MESSAGE_DESC	
				F8	AD	9F	0002A	PUSHAB	MESSAGE_DESC	
				04	AC	DD	0002D	PUSHL	MESSAGE_CODE	
		00000000G	00	05	FB	00030	CALLS	#5, SYSSGETMSG		

08	00	56	F8	AD 9E 00037	MOVAB	MESSAGE_DESC, FAO_CTL_STRING	:	1433
				04 11 00038	BRB	2\$:	1423
		56	08	AC D0 0003D 1\$:	MOVL	CONTROL_STRING, FAO_CTL_STRING	:	1435
		50		67 9E 00041 2\$:	MOVAB	LINE_DESC, R0	:	1439
		50		56 D1 00044	CPL	FAO_CTL_STRING, R0	:	
				2D 13 00047	BEQL	3\$:	
		6E		00 2C 00049	MOVCS	#0, (SP), #0, #8, LINE_DESC	:	1442
				67 0004E			:	
		67	0400	8F B0 0004F	MOVW	#1024, LINE_DESC	:	1443
	04	A7	08	A7 9E 00054	MOVAB	LINE_BUFFER, LINE_DESC+4	:	1444
			0C	AC 9F 00059	PUSHAB	ARGS	:	1449
				57 DD 0005C	PUSHL	R7	:	
			00C0	8F BB 0005E	PUSHR	#*M<R6,R7>	:	
	00000000G	00		04 FB 00062	CALLS	#4, SYS\$FAOL	:	
	0824	C7	04	A7 D0 00069	MOVL	LINE_DESC+4, OUTPUT_RAB+40	:	1451
	081E	C7		67 B0 0006F	MOVW	LINE_DESC, OUTPUT_RAB+34	:	1452
				0B 11 00074	BRB	4\$:	1439
	0824	C7	04	A6 D0 00076 3\$:	MOVL	4(FAO_CTL_STRING), OUTPUT_RAB+40	:	1456
	081E	C7		66 B0 0007C	MOVW	(FAO_CTL_STRING), OUTPUT_RAB+34	:	1457
			07FC	C7 9F 00081 4\$:	PUSHAB	OUTPUT_RAB	:	1460
	00000000G	00		01 FB 00085	CALLS	#1, SYS\$PUT	:	
		0F		50 E8 0008C	BLBS	STATUS, 5\$:	1461
			07FC	C7 9F 0008F	PUSHAB	OUTPUT_RAB	:	
			007910D4	8F DD 00093	PUSHL	#7934184	:	
	FEC3	CF		02 FB 00099	CALLS	#2, DIR\$FILE_ERROR	:	
				67 B4 0009E 5\$:	CLRW	LINE_DESC	:	1463
		50		01 D0 000A0	MOVL	#1, R0	:	1465
				04 000A3	RET		:	1467

; Routine Size: 164 bytes, Routine Base: \$CODE\$ + 0A7B

```

: 1075      1468 1 GLOBAL ROUTINE SYSS$FORMAT_ACL =
: 1076      1469 1 ++
: 1077      1470 1
: 1078      1471 1 FUNCTIONAL DESCRIPTION:
: 1079      1472 1
: 1080      1473 1     This is a dummy routine to satisfy the global reference of
: 1081      1474 1     the $FORMAT_ACL macro. It simply calls the real service,
: 1082      1475 1     which has been dynamically loaded.
: 1083      1476 1
: 1084      1477 1 CALLING SEQUENCE:
: 1085      1478 1     via $FORMAT_ACL macro
: 1086      1479 1
: 1087      1480 1 INPUT PARAMETERS:
: 1088      1481 1
: 1089      1482 1 IMPLICIT INPUTS:
: 1090      1483 1     FORMAT_ACL_ADDR contains the loaded address of SYSS$FORMAT_ACL
: 1091      1484 1
: 1092      1485 1 OUTPUT PARAMETERS:
: 1093      1486 1     none
: 1094      1487 1
: 1095      1488 1 IMPLICIT OUTPUTS:
: 1096      1489 1     none
: 1097      1490 1
: 1098      1491 1 ROUTINE VALUE:
: 1099      1492 1     status returned from sys$format_acl service
: 1100      1493 1
: 1101      1494 1 SIDE EFFECTS:
: 1102      1495 1     none
: 1103      1496 1
: 1104      1497 1 --
: 1105      1498 2 BEGIN
: 1106      1499 2 BUILTIN
: 1107      1500 2     CALLG,AP;
: 1108      1501 2
: 1109      1502 2 LOCAL
: 1110      1503 2     STATUS;
: 1111      1504 2
: 1112      1505 2 RETURN CALLG(.AP, @FORMAT_ACL_ADDR)
: 1113      1506 1 END;

```

```

          0000' DF          0000 0000          .ENTRY SYSS$FORMAT_ACL, Save nothing          : 1468
          6C FA 00002          CALLG (AP), @FORMAT_ACL_ADDR          : 1505
          04 00007          RET          : 1506

```

; Routine Size: 8 bytes, Routine Base: \$CODE\$ + 0B1F

```

: 1114      1507 1
: 1115      1508 1 END
: 1116      1509 0 ELUDOM

```

PSECT SUMMARY

Name	Bytes	Attributes
DIR\$COMMON	2164	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, OVR, NOPIC, ALIGN(0)
\$OWNS	691	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$PLITS	928	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODES	2855	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

Library Statistics

File	----- Symbols -----		Pages Mapped	Processing Time
	Total	Loaded Percent		
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	190 1	1000	00:01.9

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:DIRECTORY/OBJ=OBJ\$:DIRECTORY MSRCS\$:DIRECTORY/UPDATE=(ENHS:DIRECTORY)

: Size: 2855 code + 3783 data bytes
 : Run Time: 00:59.6
 : Elapsed Time: 02:56.1
 : Lines/CPU Min: 1518
 : Lexemes/CPU-Min: 28952
 : Memory Used: 746 pages
 : Compilation Complete

The image displays a grid of 100 terminal window screenshots, arranged in 10 rows and 10 columns. Each window shows a different system utility or command-line output. The windows are densely packed and contain various text-based data, including lists, tables, and command prompts. Some windows are larger than others, indicating they are active or selected. The overall appearance is that of a multi-processor system terminal session.

Key window titles and content visible include:

- DIRMSG LIS**: A window showing a list of messages or logs.
- MAIN LIS**: A window showing a list of mainframe or system data.
- DIR**: A window showing a directory listing.
- DIRECTORY MAP**: A window showing a graphical or structured map of a directory.
- DIRECTDEF REQ**: A window showing a request for directory definitions.
- OUTPUT LIS**: A window showing a list of output files or logs.
- DISPLYDEF SCL**: A window showing a display definition for a screen layout.
- DIRECTMSG LIS**: A window showing a list of directory messages.

DISMOUNT MAP

DISKQIOTA LIS

DISKQ

DISKQIOTA MAP

DISPLAY LIS

The image contains a dense grid of text, likely a technical manual or code listing. The text is arranged in a regular grid pattern across the page. Several specific terms are highlighted in larger, bolded text, including 'DISMOUNT MAP', 'DISKQIOTA LIS', 'DISKQ', 'DISKQIOTA MAP', and 'DISPLAY LIS'. The overall appearance is that of a high-resolution scan of a printed document, possibly a microfiche or a high-density data card.