

[illegible]


```
1 0001 0 MODULE unlock ( IDENT = 'V04-000'
2 0002 0 ADDRESSING_MODE (EXTERNAL = GENERAL),
3 0003 0 MAIN = unlock) =
4 0004 1 BEGIN
5 0005 1
6 0006 1
7 0007 1 *****
8 0008 1 *
9 0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
10 0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
11 0011 1 * ALL RIGHTS RESERVED.
12 0012 1 *
13 0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
14 0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
15 0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
16 0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
17 0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
18 0018 1 * TRANSFERRED.
19 0019 1 *
20 0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
21 0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
22 0022 1 * CORPORATION.
23 0023 1 *
24 0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
25 0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
26 0026 1 *
27 0027 1 *
28 0028 1 *****
29 0029 1
30 0030 1 ++
31 0031 1 FACILITY: UNLOCK Command
32 0032 1
33 0033 1 ABSTRACT:
34 0034 1
35 0035 1 This utility unlocks files and directories.
36 0036 1
37 0037 1 ENVIRONMENT:
38 0038 1
39 0039 1 VAX/VMS operating system. unprivileged user mode,
40 0040 1
41 0041 1 AUTHOR: Greg Robert, Nov 1979
42 0042 1
43 0043 1
44 0044 1 Modified by:
45 0045 1
46 0046 1 V03-002 AEW0002 Anne E. Warner 28-Feb-1984
47 0047 1 Add support for search lists.
48 0048 1 - remove Related File Name field from RMS definitions.
49 0049 1 - add argument to LIB$FILE_SCAN
50 0050 1
51 0051 1 V03-001 AEW0001 Anne E. Warner 11-Oct-1983
52 0052 1 Intergrate Command Language Interface (CLI) in program.
53 0053 1
54 0054 1 V02-005 MLJ0066 Martin L. Jack, 29-Dec-1981 12:56
55 0055 1 Integrate new LIB$UNLOCK_FILE (formerly LIB$UNLOCK).
56 0056 1
57 0057 1 V204 GRR2004 G. R. Robert 17-Nov-1981
```

UNLOCK
V04-000

J 4
16-Sep-1984 00:32:25 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:10:06 [CLIUTL.SRC]UNLOCK.B32;1

Page 2
(1)

```
.. 58      0058 1  Fixed messages to display full filespecs with passwords
.. 59      0059 1  suppressed on errors and logging.
.. 60      0060 1
.. 61      0061 1  V203  GRR2003      G. R. Robert      16-Nov-1981
.. 62      0062 1  Made all external references addressing mode general.
.. 63      0063 1
.. 64      0064 1  V202  GRR2002      G. R. Robert      14-Sep-1981
.. 65      0065 1  Fixed improper descriptor initialization in confirm_action
.. 66      0066 1  by making them OWN instead of LOCAL.
.. 67      0067 1
.. 68      0068 1  V201  GRR2001      G. R. Robert      12-JUN-1981
.. 69      0069 1  Made UNLOCKDEF.REQ a permanent part of this file
.. 70      0070 1  to clean up the master disk.
.. 71      0071 1
.. 72      0072 1  --
.. 73      0073 1
.. 74      0074 1
.. 75      0075 1  Include files
.. 76      0076 1
.. 77      0077 1
.. 78      0078 1  LIBRARY 'SYSS$LIBRARY:STARLET.L32';      ! VAX/VMS common definitions
.. 79      0079 1
.. 80      0080 1  LIBRARY 'SYSS$LIBRARY:CLIMAC.L32';      ! CLI macros
```



```

82 0081 1 |
83 0082 1 | Common definitions for the UNLOCK command
84 0083 1 |
85 0084 1 |
86 0085 1 |
87 0086 1 |
88 0087 1 |
89 0088 1 |
90 0089 1 | STRUCTURE
91 0090 1 |     BBLOCK [O, P, S, E; N] =
92 0091 1 |         [N]
93 0092 1 |         (BBLOCK+O)<P,S,E>;
94 0093 1 |
95 0094 1 | MACRO
96 0095 1 |
97 0096 1 |     A) Macro to describe a string
98 0097 1 |     B) Macro to generate a quadword string descriptor
99 0098 1 |     C) Macro to generate the address of a string descriptor
100 0099 1 |
101 0100 1 |     PRIMDESC (str) = %CHARCOUNT (str), UPLIT (%ASCII str)%,
102 0101 1 |     INITDESC (str) = BBLOCK [DSC$C_S_BLN] INITIAL (PRIMDESC (str))%,
103 0102 1 |     ADDRDESC (str) = UPLIT (PRIMDESC (str))%,
104 0103 1 |
105 0104 1 |
106 0105 1 |     Macro to signal a condition to the handler
107 0106 1 |
108 0107 1 |
109 M 0108 1 |     write_message(msg) =
110 M 0109 1 |         SIGNAL(msg) | Pass the message code
111 M 0110 1 |         %IF %LENGTH GTR 1 | and if more than 1 arg
112 0111 1 |             %THEN ,%REMAINING %FI) %, | then the rest too
113 0112 1 |
114 0113 1 |
115 0114 1 |     $SHR_MESSAGES - a macro which defines facility-specific message codes
116 0115 1 |     which are based on the system-wide shared message codes.
117 0116 1 |
118 0117 1 |     $SHR_MESSAGES( name, code, (msg,severity), ... )
119 0118 1 |
120 0119 1 |     where:
121 0120 1 |         "name" is the name of the facility (e.g., COPY)
122 0121 1 |         "code" is the corresponding facility code (e.g., 103)
123 0122 1 |         "msg" is the name of the shared message (e.g., BEGIN)
124 0123 1 |         "severity" is the desired message severity (e.g., 1, 0, 2)
125 0124 1 |
126 0125 1 |
127 M 0126 1 |     $SHR_MESSAGES( FACILITY_NAME, FACILITY_CODE ) =
128 M 0127 1 |         [ LITERAL
129 0128 1 |         SHR$MSG_IDS( FACILITY_NAME, FACILITY_CODE, %REMAINING ); %,
130 0129 1 |
131 M 0130 1 |     SHR$MSG_IDS( FACILITY_NAME, FACILITY_CODE ) [ VALUE ] =
132 0131 1 |         SHR$MSG_CALC( FACILITY_NAME, FACILITY_CODE, %REMOVE(VALUE) ) %,
133 0132 1 |
134 M 0133 1 |     SHR$MSG_CALC( FACILITY_NAME, FACILITY_CODE, MSG_ID, SEVERITY ) =
135 M 0134 1 |         %NAME(FACILITY_NAME, '$', MSG_ID) = %NAME('SHR$', MSG_ID) + FACILITY_CODE*65536 +
136 M 0135 1 |         %IF %DECLARED(%NAME('ST$K', SEVERITY))
137 M 0136 1 |             %THEN %NAME('ST$K', SEVERITY)
138 0137 1 |             %ELSE SEVERITY %FI-%;
```

```
: 140      0138 1 !
: 141      0139 1 ! Define message codes.
: 142      0140 1 !
: 143      0141 1 !
: 144      P 0142 1 $shr_messages(msg,146,
: 145      P 0143 1 (notlocked,info), ! File not locked in the first place
: 146      P 0144 1 (unlocked,info), ! File unlocked
: 147      P 0145 1 (searchfail,error), ! Could not find file
: 148      0146 1 );
: 149      0147 1 !
: 150      0148 1 !
: 151      0149 1 ! EQUATED SYMBOLS
: 152      0150 1 !
: 153      0151 1 !
: 154      0152 1 LITERAL
: 155      0153 1 TRUE = 1, ! BOOLEAN TRUE
: 156      0154 1 FALSE = 0, ! BOOLEAN FALSE
: 157      0155 1 OK = 1, ! SUCCESS RETURN CODE
: 158      0156 1 ERROR = 2; ! ERROR RETURN CODE
: 159      0157 1 !
: 160      0158 1 ! VALUES FOR QUALIFIER MASK
: 161      0159 1 !
: 162      0160 1 LITERAL
: 163      0161 1 QUAL_LOG = 0, ! FIRST BIT IS FOR LOG QUALIFIER
: 164      0162 1 QUAL_CONFIRM = 1; ! SECOND BIT IS FOR CONFIRM QUALIFIER
: 165      0163 1 !
: 166      0164 1 ! CLI ROUTINES
: 167      0165 1 !
: 168      0166 1 EXTERNAL ROUTINE
: 169      0167 1 CLIPRESENT,
: 170      0168 1 CLISGET_VALUE;
: 171      0169 1 !
: 172      0170 1 BIND
: 173      0171 1 LOG_DESC = $DESCRIPTOR ('LOG'),
: 174      0172 1 CONFIRM_DESC = $DESCRIPTOR ('CONFIRM');
: 175      0173 1 !
: 176      0174 1 ! CLI RETURN STATUS CODES
: 177      0175 1 !
: 178      0176 1 !
: 179      0177 1 !
: 180      0178 1 !
```


182	0179	1	!		
183	0180	1	!	Table of contents	
184	0181	1	!		
185	0182	1	!		
186	0183	1	!	FORWARD ROUTINE	
187	0184	1	!	unlock,	! Main unlock routine
188	0185	1	!	unlock_action,	! Called for each unlock action
189	0186	1	!	confirm_action,	! Interrogate user
190	0187	1	!	log_results,	! Inform user of results
191	0188	1	!	get_unlockqls,	! Get command qualifiers
192	0189	1	!	handler,	! Condition handler
193	0190	1	!	search_error;	! LIB\$FILE_SCAN error handler
194	0191	1	!		
195	0192	1	!		
196	0193	1	!		
197	0194	1	!	External routines	
198	0195	1	!		
199	0196	1	!		
200	0197	1	!	EXTERNAL ROUTINE	
201	0198	1	!	sys\$fao,	! Expands Formatted ascii messages
202	0199	1	!	lib\$get_command,	! Talk to SYS\$COMMAND
203	0200	1	!	lib\$file_scan,	! Implements wildcarding and stickiness
204	0201	1	!	lib\$unlock_file;	! Unlocks files
205	0202	1	!		

```
: 207      0203 1 |
: 208      0204 1 | Storage definitions
: 209      0205 1 |
: 210      0206 1 |
: 211      0207 1 | OWN
: 212      0208 1 | unlock$flags      : BITVECTOR[32] | General DCL flagword
: 213      0209 1 |      INITIAL(0),  | Initially none present
: 214      0210 1 | qualifier$flags   : BITVECTOR[32] | Qualifier presence bitmap
: 215      0211 1 |      INITIAL(0),  | Initially clear
: 216      0212 1 | worst_error: BBLOCK[4] | Worst error encountered
: 217      0213 1 |      INITIAL(ss$_normal); | Initially normal status
```



```
.. 219      0214 1  !
.. 220      0215 1  ! Define RMS blocks
.. 221      0216 1  !
.. 222      0217 1  !
.. 223      0218 1  !
.. 224      0219 1  ! OWN
.. 225      0220 1  output_nam_result:      ! Resultant output name
.. 226      0221 1  VECTOR [nam$c_maxrss,BYTE],
.. 227      0222 1
.. 228      0223 1  output_nam_expanded:      ! Expanded output name
.. 229      0224 1  VECTOR [nam$c_maxrss,BYTE],
.. 230      0225 1
.. 231      P 0226 1  output_nam: $NAM(      ! File name block
.. 232      P 0227 1  ESA = output_nam_expanded, ! File name before open
.. 233      P 0228 1  ESS = nam$c_maxrss,
.. 234      P 0229 1  RSA = output_nam_result,   ! File name after open
.. 235      0230 1  RSS = nam$c_maxrss),
.. 236      0231 1
.. 237      P 0232 1  output_fab: $FAB(      ! FAB for output
.. 238      0233 1  NAM = output_nam);      ! Address of name block
```

```
: 240 0234 1 ROUTINE unlock =                ! unlock Main routine
: 241 0235 1
: 242 0236 1 ++
: 243 0237 1 Functional description
: 244 0238 1
: 245 0239 1 This is the main control routine for the unlock command.
: 246 0240 1 It is called from the command language interpreter to
: 247 0241 1 unlock files and directories.
: 248 0242 1
: 249 0243 1 Calling sequence
: 250 0244 1
: 251 0245 1 unlock() from the Command Language Interpreter
: 252 0246 1
: 253 0247 1 Input parameters
: 254 0248 1
: 255 0249 1 None
: 256 0250 1
: 257 0251 1 Output parameters
: 258 0252 1
: 259 0253 1 None
: 260 0254 1
: 261 0255 1 Routine value
: 262 0256 1
: 263 0257 1 Worst error encountered during processing or SS$_NORMAL.
: 264 0258 1
: 265 0259 1 -----
: 266 0260 1
: 267 0261 2 BEGIN
: 268 0262 2
: 269 0263 2 LOCAL
: 270 0264 2 file_desc : $BBLOCK[DSC$_S_BLN], ! Descriptor for file
: 271 0265 2 status, ! Status return
: 272 0266 2 scan_context; ! Sticky context arg. for LIB$FILE_SCAN
: 273 0267 2
: 274 0268 2
: 275 0269 2 BUILTIN
: 276 0270 2 FP; ! Define register FP
: 277 0271 2
: 278 0272 2
: 279 0273 2 Initialize descriptor
: 280 0274 2
: 281 0275 2 CH$FILL (0,DSC$_S_BLN, FILE_DESC);
: 282 0276 2 FILE_DESC [DSC$_B_CLASS] = DSC$_K_CLASS_D;
: 283 0277 2
: 284 0278 2
: 285 0279 2
: 286 0280 2 Declare signal handler in order to record the most severe
: 287 0281 2 error message issued, to be returned on exit of image.
: 288 0282 2
: 289 0283 2 .fp = handler; ! Set condition handler
```



```
291 0284 2 |
292 0285 2 |      Parse command qualifiers
293 0286 2 |
294 0287 2 |
295 0288 2 |get_unlockqls();      ! Get command qualifiers
296 0289 2 |
297 0290 2 |
298 0291 2 |      Begin the main loop of the program
299 0292 2 |      Process each file in the input list
300 0293 2 |
301 0294 2 |
302 0295 2 |scan_context = 0;      ! zero sticky context argument
303 0296 2 |
304 0297 2 |WHILE CLISGET_VALUE ($DESCRIPTOR('P1'), FILE_DESC) DO      ! For each output file
305 0298 2 |      BEGIN
306 0299 2 |          output_fab[fab$l_fna] = .file_desc[dsc$a_pointer];
307 0300 2 |          output_fab[fab$b_fns] = .file_desc[dsc$w_length];
308 0301 2 |
309 0302 2 |          ! Call lib$file_scan to handle wildcarding and stickiness
310 0303 2 |          ! lib$file_scan will call unlock_action for each successful
311 0304 2 |          ! file match and search_error for any failures
312 0305 2 |
313 0306 2 |
314 0307 2 |          lib$file_scan (      ! Call file scanner with
315 0308 2 |              output_fab,      ! -fab address
316 0309 2 |              unlock_action,    ! -success routine
317 0310 2 |              search_error,     ! -error routine
318 0311 2 |              scan_context)     ! -sticky context
319 0312 2 |
320 0313 2 |      END;      ! End of WHILE domain
321 0314 2 |
322 0315 2 |RETURN .worst_error;
323 0316 2 |
324 0317 1 |END;
```

```
.TITLE UNLOCK
.IDENT \V04-000\

.PSECT $PLITS,NOWRT,NOEXE,2

47 4F 4C 00000 P.AAB: .ASCII \LOG\
00003 .BLKB 1
00000003 00004 P.AAA: .LONG 3
00000000 00008 .ADDRESS P.AAB
4D 52 49 46 4E 4F 43 0000C P.AAD: .ASCII \CONFIRM\
00013 .BLKB 1
00000007 00014 P.AAC: .LONG 7
00000000 00018 .ADDRESS P.AAD
31 50 0001C P.AAF: .ASCII \P1\
0001E .BLKB 2
00000002 00020 P.AAE: .LONG 2
00000000 00024 .ADDRESS P.AAF

.PSECT $OWNS,NOEXE,2

00000000 00000 UNLOCK$FLAGS:
```

```
00000000 00004 QUALIFIERS$FLAGS: .LONG 0
00000001 00008 WORST_ERROR: .LONG 0
0000000C 0000C OUTPUT_NAM_RESULT: .LONG 1
0010B .BCKB 255
0010C OUTPUT_NAM_EXPANDED: .BLKB 1
0020B .BCKB 255
02 0020C OUTPUT_NAM: .BLKB 1
60 0020D .BYTE 2
FF 0020E .BYTE 96
00 0020F .BYTE -1
00000000 00210 .ADDRESS OUTPUT_NAM_RESULT
00 00214 .BYTE 0
00 00215 .BYTE 0
FF 00216 .BYTE -1
00 00217 .BYTE 0
00000000 00218 .ADDRESS OUTPUT_NAM_EXPANDED
00000000 0021C .LONG 0
0000# 00220 .WORD 0[8]
0000# 00230 .WORD 0[3]
0000# 00236 .WORD 0[3]
00000000 0023C .LONG 0
00000000 00240 .LONG 0
00 00244 .BYTE 0
00 00245 .BYTE 0
00 00246 .BYTE 0
00 00247 .BYTE 0
00 00248 .BYTE 0
00 00249 .BYTE 0
00# 0024A .BYTE 0[2]
00000000 0024C .LONG 0
00000000 00250 .LONG 0
00000000 00254 .LONG 0
00000000 00258 .LONG 0
00000000 0025C .LONG 0
00000000 00260 .LONG 0
00000000# 00264 .LONG 0[2]
03 0026C OUTPUT_FAB: .BYTE 3
50 0026D .BYTE 80
0000 0026E .WORD 0
00000000 00270 .LONG 0
00000000 00274 .LONG 0
00000000 00278 .LONG 0
00000000 0027C .LONG 0
0000 00280 .WORD 0
02 00282 .BYTE 2
00 00283 .BYTE 0
00000000 00284 .LONG 0
00 00288 .BYTE 0
00 00289 .BYTE 0
00 0028A .BYTE 0
```


02	0028B	.BYTE	2
00000000	0028C	.LONG	0
00000000	00290	.LONG	0
00000000	00294	.ADDRESS	OUTPUT_NAM
00000000	00298	.LONG	0
00000000	0029C	.LONG	0
00	002A0	.BYTE	0
00	002A1	.BYTE	0
0000	002A2	.WORD	0
00000000	002A4	.LONG	0
0000	002A8	.WORD	0
00	002AA	.BYTE	0
00	002AB	.BYTE	0
00000000	002AC	.LONG	0
00000000	002B0	.LONG	0
0000	002B4	.WORD	0
00	002B6	.BYTE	0
00	002B7	.BYTE	0
00000000	002B8	.LONG	0

LOG_DESC=	P.AAA
CONFIRM_DESC=	P.AAC
.EXTRN	CLISPRESENT, CLISGET_VALUE
.EXTRN	SYSSFAO, LIB\$GET_COMMAND
.EXTRN	LIB\$FILE_SCAN, LIB\$UNLOCK_FILE
.PSECT	\$CODE\$,NOWRT,2

08	00	5E	003C	00000	UNLOCK:	.WORD	Save R2,R3,R4,R5	0234	
		6E	0C	C2	00002	SUBL2	#12, SP		
			00	2C	00005	MOVCS	#0, (SP), #0, #8, FILE_DESC	0275	
	07	AE	04	AE	0000A				
		6D	02	90	0000C	MOVB	#2, FILE_DESC+3	0276	
	0000V	CF	0000V	CF	9E	00010	HANDLER, (FP)	0283	
			00	FB	00015	CALLS	#0, GET_UNLOCKQLS	0288	
			6E	D4	0001A	CLRL	SCAN_CONTEXT	0295	
			04	AE	9F	0001C	PUSHAB	FILE_DESC	0297
			0000'	CF	9F	0001F	PUSHAB	P.AAE	
	00000000G	00	02	FB	00023	CALLS	#2, CLISGET_VALUE		
		23	50	E9	0002A	BLBC	R0, 2\$		
	0000'	CF	08	AE	D0	0002D	MOVL	FILE_DESC+4, OUTPUT_FAB+44	0299
	0000'	CF	04	AE	90	00033	MOVB	FILE_DESC, OUTPUT_FAB+52	0300
			5E	DD	00039	PUSHL	SP	0307	
	0000V		CF	9F	0003B	PUSHAB	SEARCH_ERROR		
	0000V		CF	9F	0003F	PUSHAB	UNLOCK_ACTION		
	0000'		CF	9F	00043	PUSHAB	OUTPUT_FAB		
	00000000G	00	04	FB	00047	CALLS	#4, LIB\$FILE_SCAN		
			CC	11	0004E	BRB	1\$		
	50		0000'	CF	D0	00050	MOVL	WORST_ERROR, R0	0315
			04	00055	2\$:	RET		0317	

; Routine Size: 86 bytes, Routine Base: \$CODE\$ + 0000

```
326 0318 1 ROUTINE unlock_action (fab) =
327 0319 1
328 0320 1 ----
329 0321 1
330 0322 1 Functional description
331 0323 1
332 0324 1 This routine is called from lib$file_scan whenever
333 0325 1 a successful file match occurs
334 0326 1
335 0327 1 Input parameters
336 0328 1
337 0329 1 fab = Address of block describing the file
338 0330 1 fab$l_nam = pointer to name block
339 0331 1
340 0332 1 Output parameters
341 0333 1
342 0334 1 None
343 0335 1
344 0336 1 ----
345 0337 1
346 0338 2 BEGIN
347 0339 2
348 0340 2 MAP fab: REF BBLOCK; ! Define fab block format
349 0341 2 BIND nam = .fab[fab$l_nam]: BBLOCK; ! Define name block
350 0342 2
351 0343 2 LOCAL desc: VECTOR[2]; ! Temporary string descriptor
352 0344 2 LOCAL status; ! Receives status
353 0345 2
354 0346 2
355 0347 2 If /CONFIRM was set by the user then interrogate him to see if
356 0348 2 this file is to be unlocked
357 0349 2
358 0350 2
359 0351 2 IF (.qualifier$flags[qual_confirm]) ! If confirmation requested
360 0352 2 THEN
361 0353 2 IF NOT (confirm_action ( ! Call confirmation rout. with
362 0354 2 .fab)) ! -address of fab
363 0355 2 THEN return(false); ! If not confirmed then exit
364 0356 2
365 0357 2
366 0358 2 Now load the local descriptor with the file name size and address
367 0359 2 from the nam block
368 0360 2
369 0361 2
370 0362 2 desc[0] = .nam[nam$b_rsl]; ! Resultant file name size
371 0363 2 desc[1] = .nam[nam$l_rsa]; ! Resultant file name address
372 0364 2
373 0365 2 IF .nam [nam$b_rsl] EQL 0 ! If the resultant name is null
374 0366 2 THEN BEGIN ! then use the expanded name:
375 0367 2 desc [0] = .nam[nam$b_esl]; ! expanded file name size
376 0368 2 desc [1] = .nam[nam$l_esa]; ! expanded file name address
377 0369 2 END;
378 0370 2
379 0371 2
380 0372 2
381 0373 2 Call LIB$UNLOCK_FILE to unlock the file
382 0374 2
```



```

383 0375 2
384 0376 IF NOT (status = lib$unlock_file(desc)) ! Call unlock with file name
385 0377 THEN
386 0378 BEGIN
387 0379 write_message(.status); ! Inform the user
388 0380 return(.status); ! Return to caller
389 0381 END;
390 0382
391 0383
392 0384
393 0385
394 0386
395 0387
396 0388
397 0389 IF (.status EQL SSS_WASCLR) ! If file not locked
398 0390 THEN
399 0391 write_message( ! Tell the user
400 0392 msg$_notlocked, ! that the file wasn't locked
401 0393 1, ! (one FAO argument),
402 0394 desc) ! and name the file
403 0395 ELSE ! File was unlocked
404 0396 IF (.qualifier$flags[qual_log]) ! If logging requested
405 0397 THEN log_results (.fab); ! then call the logger
406 0398
407 0399 RETURN(true); ! Both returns above
408 0400 ! are ok!
409 0401
410 0402 1 END;
```

```

001C 00000 UNLOCK_ACTION:
54 00000000G 00 9E 00002 .WORD Save R2,R3,R4
5E 08 C2 00009 MOVAB LIB$SIGNAL, R4
53 04 AC D0 0000C SUBL2 #8, SP
52 28 A3 D0 00010 MOVL FAB, R3
OA 0000' CF 01 E1 00014 MOVL 40(R3), R2
0000V CF 53 DD 0001A BBC #1, QUALIFIERS$FLAGS, 1$
53 01 FB 0001C PUSH R3
6E 03 A2 9A 00024 1$: CALLS #1, CONFIRM_ACTION
04 AE 04 A2 D0 00028 BLBC R0, 6$
03 A2 95 0002D MOVZBL 3(R2), DESC
06E 0B A2 9A 00032 2$: MOVL 4(R2), DESC+4
04 AE 0C A2 D0 00036 TSTB 3(R2)
00000000G 00 5E DD 0003B BNEQ 2$
52 01 FB 0003D MOVL 11(R2), DESC
09 50 D0 00044 MOVL 12(R2), DESC+4
64 01 FB 0004C PUSH SP
50 52 E8 00047 CALLS #1, LIB$UNLOCK_FILE
09 52 DD 0004A MOVL R0, STATUS
64 01 FB 0004C BLBS STATUS, 3$
50 52 D0 0004F PUSH STATUS
04 00052 CALLS #1, LIB$SIGNAL
RET MOVL STATUS, R0
0318
0341
0351
0354
0362
0363
0365
0367
0368
0376
0379
0380
```

UNLOCK
V04-000

1 5
16-Sep-1984 00:32:25
14-Sep-1984 12:10:06

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]UNLOCK.B32;1

Page 14
(9)

01	52	D1	00053	3\$:	CMPL	STATUS, #1	: 0389
	0F	12	00056		BNEQ	4\$	: 0394
	5E	DD	00058		PUSHL	SP	: 0396
	01	DD	0005A		PUSHL	#1	: 0397
64	009212B3	8F	DD	0005C	PUSHL	#9573043	: 0399
	03	FB	00062		CALLS	#3, LIB\$SIGNAL	: 0402
	0C	11	00065		BRB	5\$	: 0389
07	0000'	CF	E9	00067	4\$:	BLBC	QUALIFIERS\$FLAGS, 5\$
	53	DD	0006C		PUSHL	R3	: 0396
0000V	CF	01	FB	0006E	CALLS	#1, LOG_RESULTS	: 0397
	50	01	D0	00073	5\$:	MOVL	#1, R0
		04	00076		RET		: 0399
		50	D4	00077	6\$:	CLRL	R0
		04	00079		RET		: 0402

; Routine Size: 122 bytes, Routine Base: \$CODE\$ + 0056


```

412 0403 1 ROUTINE confirm_action (fab) =
413 0404 1
414 0405 1 ----
415 0406 1
416 0407 1 Functional description
417 0408 1
418 0409 1 This routine is called from the main loop whenever
419 0410 1 confirmation is requested.
420 0411 1
421 0412 1 Input parameters
422 0413 1
423 0414 1 fab = Address of block describing the file
424 0415 1 fab$l_nam = pointer to name block
425 0416 1
426 0417 1 Output parameters
427 0418 1
428 0419 1 TRUE --> Action should be taken
429 0420 1 FALSE --> Action should be cancelled
430 0421 1
431 0422 1 ----
432 0423 1
433 0424 1
434 0425 2 BEGIN
435 0426 2
436 0427 2 MAP fab: REF BBLOCK; : Define fab block format
437 0428 2 BIND nam = .fab[fab$l_nam]: BBLOCK; : Define name block
438 0429 2
439 0430 2 OWN
440 0431 2 file_desc: BBLOCK[dsc$sc_s_bln], : file_descriptor for file name
441 0432 2 fao_desc: BBLOCK[dsc$sc_s_bln], : FAO work area descriptor
442 0433 2 reply_desc: BBLOCK[dsc$sc_s_bln] : Buffer desc. for reply
443 0434 2 INITIAL (
444 0435 2 WORD (0), : -size = 0
445 0436 2 BYTE (dsc$sk_dtype_t), : -argument type = ascii
446 0437 2 BYTE (dsc$sk_class_d), : -class = dynamic
447 0438 2 LONG (0)); : -pointer = 0
448 0439 2 LOCAL
449 0440 2 status: BLOCK[1], : Recieves status
450 0441 2 length: WORD, : Length of resultant message
451 0442 2 char, : Character work area
452 0443 2 fao_buffer: VECTOR[512,BYTE]; : FAO work area
453 0444 2
454 0445 2
455 0446 2
456 0447 2 Initialize descriptors with:
457 0448 2 1) file name --> file_desc
458 0449 2 2) FAO buffer --> fao_desc
459 0450 2
460 0451 2
461 0452 2 file_desc[dsc$w_length] = .nam[nam$b_rsl]; : Resultant file name length
462 0453 2 file_desc[dsc$a_pointer] = .nam[nam$t_rsa]; : Resultant file name address
463 0454 2
464 0455 2 IF .nam[nam$b_rsl] EQL 0 : If no resultant name
465 0456 2 THEN BEGIN : then use expanded name:
466 0457 2 file_desc[dsc$w_length] = .nam[nam$b_esl]; : expanded file name length
467 0458 2 file_desc[dsc$a_pointer] = .nam[nam$t_esa]; : expanded file name address
468 0459 2 END;
```

```
469 0460 2
470 0461 fao_desc[dsc$w_length] = 512;          ! FAO work area size
471 0462 fao_desc[dsc$a_pointer] = fao_buffer; ! FAO work area address
472 0463
473 0464
474 0465
475 0466      Now call SYSSFAO to expand message
476 0467
477 0468
478 0469 IF NOT (status = SYSSFAO (                ! Call system service with
479 0470                                     ! -messages desc. address
480 0471                                     addrdesc('!AS, unlock? (Y or N): '),
481 0472                                     length,          ! -place to put result length
482 0473                                     fao_desc,         ! -descriptor of a work area
483 0474                                     file_desc))      ! -one FAO arguement
484 0475 THEN
485 0476     BEGIN
486 0477     write_message (.status);          ! Tell the user
487 0478     return (.status);                ! Return to the caller
488 0479     END;
489 0480
490 0481
491 0482      Now question user by calling LIB$GET_COMMAND using the result
492 0483      of the FAO call as a prompt string
493 0484
494 0485
495 0486 fao_desc[dsc$w_length] = .length;        ! Move in prompt string size
496 0487
497 0488 IF NOT (status = LIB$GET_COMMAND (        ! Call library routine with
498 0489                                     ! -reply buffer desc. addr.
499 0490                                     reply_desc,      ! -prompt string desc. addr.
500 0491                                     fao_desc))
501 0492 THEN
502 0493     BEGIN
503 0494     IF (.status EQL rms$eof)            ! If this was end of file then
504 0495     THEN status[sts$severity] =        ! make it a severe (i.e. non-
505 0496     sts$k_severe;                      ! returnable) error
506 0497     write_message (.status);          ! Tell the user
507 0498     return (.status);                ! Return to the caller
508 0499     END;
509 0500
510 0501      Now retrieve one charater from the buffer and set the return
511 0502      status depending on its value
512 0503
513 0504
514 0505 IF (.reply_desc[dsc$w_length] EQL 0)    ! Did user hit <CR>?
515 0506 THEN return (false);                  ! Yes, return false
516 0507
517 0508 char = CH$RCHAR (CH$PTR (.reply_desc[dsc$a_pointer]));
518 0509
519 0510 IF (.char EQL 'Y' OR                  ! If user responded 'Y'
520 0511     .char EQL 'y' OR
521 0512     .char EQL 'T' OR
522 0513     .char EQL 't')
523 0514 THEN return (true);                  ! Return 'TAKE ACTION'
524 0515 ELSE return (false);                 ! Return 'CANCEL ACTION'
525 0516 2
```


: 526

0517 1 END;

```
.PSECT SPLITS,NOWRT,NOEXE,2
59 28 20 3F 6B 63 6F 6C 6E 75 20 2C 53 41 21 00028 P.AAH: .ASCII \!AS, unlock? (Y or N): \<0>
      00 20 3A 29 4E 20 72 6F 20 00037
      00000017 00040 P.AAG: .LONG 23
      00000000 00044 .ADDRESS P.AAH
      .PSECT $OWNS$,NOEXE,2
      002BC FILE_DESC:
      .BLKB 8
      002C4 FAO_DESC:
      .BLKB 8
      0000 002CC REPLY_DESC:
      .WORD 0
      0E 002CE .BYTE 14
      02 002CF .BYTE 2
      00000000 002D0 .LONG 0
      .PSECT $CODE$,NOWRT,2
      000C 00000 CONFIRM_ACTION:
      .WORD Save R2,R3
      MOVAB FAO_DESC, R3
      MOVAB -512(SP), SP
      MOVL FAB, R0
      MOVL 40(R0), R0
      MOVZBW 3(R0), FILE_DESC
      MOVL 4(R0), FILE_DESC+4
      TSTB 3(R0)
      BNEQ 1$
      MOVZBW 11(R0), FILE_DESC
      MOVL 12(R0), FILE_DESC+4
      MOVW #512, FAO_DESC
      MOVAB FAO_BUFFER, FAO_DESC+4
      PUSHAB FILE_DESC
      PUSHL R3
      PUSHAB LENGTH
      PUSHAB P.AAG
      CALLS #4, SYSSFAO
      MOVL R0, STATUS
      BLBC STATUS, 2$
      MOVW LENGTH, FAO_DESC
      PUSHL R3
      PUSHAB REPLY_DESC
      CALLS #2, LIB$GET_COMMAND
      MOVL R0, STATUS
      BLBS STATUS, 3$
      CMPL STATUS, #98938
      BNEQ 2$
      INSV #4, #0, #3, STATUS
      0000' CF 9E 00002
      FDFC CE 9E 00007
      04 AC D0 0000C
      28 A0 D0 00010
      F8 A3 03 A0 9B 00014
      FC A3 04 A0 D0 00019
      03 A0 95 0001E
      0A 12 00021
      F8 A3 0B A0 9B 00023
      FC A3 0C A0 D0 00028
      63 0200 8F B0 0002D 1$:
      04 AE 9E 00032
      F8 A3 9F 00037
      53 DD 0003A
      08 AE 9F 0003C
      0000' CF 9F 0003F
      04 FB 00043
      50 D0 0004A
      52 E9 0004D
      6E B0 00050
      53 DD 00053
      08 A3 9F 00055
      00000000G 00 02 FB 00058
      52 50 D0 0005F
      23 52 E8 00062
      63 52 D1 00065
      0001827A 8F 05 12 0006C
      52 03 04 F0 0006E
      0493
      0494
```

UNLOCK
V04-000

M 5
16-Sep-1984 00:32:25
14-Sep-1984 12:10:06

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]UNLOCK.B32;1

Page 18
(10)

00000000G	00	52	DD	00073	2\$:	PUSHL	STATUS	:	0496
	50	01	FB	00075		CALLS	#1, LIB\$SIGNAL	:	
		52	D0	0007C		MOVL	STATUS, R0	:	0497
			04	0007F		RET		:	
		08	A3	B5	3\$:	TSTW	REPLY_DESC	:	0505
			2C	13		BEQL	5\$	:	
	50	0C	B3	9A		MOVZBL	@REPLY_DESC+4, CHAR	:	0508
00000059	8F		50	D1		CMPL	CHAR, #89	:	0510
			1B	13		BEQL	4\$	:	
00000079	8F		50	D1		CMPL	CHAR, #121	:	0511
			12	13		BEQL	4\$	:	
00000054	8F		50	D1		CMPL	CHAR, #84	:	0512
			09	13		BEQL	4\$	:	
00000074	8F		50	D1		CMPL	CHAR, #116	:	0513
			04	12		BNEQ	5\$	:	
	50		01	D0	4\$:	MOVL	#1, R0	:	0515
				04		RET		:	
			50	D4	5\$:	CLRL	R0	:	0517
				04		RET		:	

; Routine Size: 180 bytes, Routine Base: \$CODE\$ + 00D0


```
528 0518 1 ROUTINE log_results (fab) =
529 0519 1
530 0520 1 ----
531 0521 1
532 0522 1 Functional description
533 0523 1
534 0524 1 This routine is called from the main loop whenever
535 0525 1 logging is requested
536 0526 1
537 0527 1 Input parameters
538 0528 1
539 0529 1 fab = Address of block describing the file
540 0530 1 fab$l_nam = pointer to name block
541 0531 1
542 0532 1 Output parameters
543 0533 1
544 0534 1 None
545 0535 1
546 0536 1 ----
547 0537 1
548 0538 2 BEGIN
549 0539 2
550 0540 2 MAP fab: REF BBLOCK; ! Define fab block format
551 0541 2 BIND nam = .fab[fab$l_nam]: BBLOCK; ! Define name block
552 0542 2
553 0543 2 LOCAL desc: VECTOR[2]; ! Temporary string descriptor
554 0544 2
555 0545 2 IF .nam[nam$b_rsl] NEQ 0 ! IF result string nonblank,
556 0546 2 THEN BEGIN ! then display it
557 0547 2 desc[0] = .nam[nam$b_rsl];
558 0548 2 desc[1] = .nam[nam$l_rsa];
559 0549 2 END
560 0550 2 ELSE IF .nam[nam$b_esl] NEQ 0 ! Or if expanded name nonblank
561 0551 2 THEN BEGIN ! then display it
562 0552 2 desc[0] = .nam[nam$b_esl];
563 0553 2 desc[1] = .nam[nam$l_esa];
564 0554 2 END
565 0555 2 ELSE BEGIN
566 0556 2 desc[0] = .fab[fab$b_fns]; ! Otherwise, use original
567 0557 2 desc[1] = .fab[fab$l_fna]; ! name string in FAB
568 0558 2 END;
569 0559 2
570 P 0560 2 write_message(msg$_unlocked, ! Inform the user
571 P 0561 2 ! With one argument
572 0562 2 desc); ! Which is the file name
573 0563 2
574 0564 2 RETURN (true);
575 0565 2
576 0566 1 END;
```

```
0000 00000 LOG_RESULTS:
5E 08 C2 00002 .WORD SUBL2 Save nothing #8, SP
```

```
: 0518
:
```

UNLOCK
V04-000

B 6
16-Sep-1984 00:32:25
14-Sep-1984 12:10:06

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]UNLOCK.B32;1

Page 20
(11)

	51	04	AC	D0	00005	MOVL	FAB, R1	:	0541
	50	28	A1	D0	00009	MOVL	40(R1), R0	:	
		03	A0	95	0000D	TSTB	3(R0)	:	0545
			0B	13	00010	BEQL	1\$	:	
	6E	03	A0	9A	00012	MOVZBL	3(R0), DESC	:	0547
04	AE	04	A0	D0	00016	MOVL	4(R0), DESC+4	:	0548
			19	11	0001B	BRB	3\$	:	0545
		0B	A0	95	0001D	1\$: TSTB	11(R0)	:	0550
			0B	13	00020	BEQL	2\$	:	
	6E	0B	A0	9A	00022	MOVZBL	11(R0), DESC	:	0552
04	AE	0C	A0	D0	00026	MOVL	12(R0), DESC+4	:	0553
			09	11	0002B	BRB	3\$	:	0550
	6E	34	A1	9A	0002D	2\$: MOVZBL	52(R1), DESC	:	0556
04	AE	2C	A1	D0	00031	MOVL	44(R1), DESC+4	:	0557
			5E	DD	00036	3\$: PUSHL	SP	:	0562
			01	DD	00038	PUSHL	#1	:	
		0092129B	8F	DD	0003A	PUSHL	#9573019	:	
00000000G	00		03	FB	00040	CALLS	#3, LIB\$SIGNAL	:	
	50		01	D0	00047	MOVL	#1, R0	:	0564
			04	0004A	RET			:	0566

; Routine Size: 75 bytes, Routine Base: \$CODE\$ + 0184


```
578 0567 1 ROUTINE get_unlockqls =
579 0568 1
580 0569 1 ----
581 0570 1
582 0571 1 Functional description
583 0572 1
584 0573 1 This routine calls CLI to obtain the command line and
585 0574 1 then all command qualifiers.
586 0575 1
587 0576 1 Input parameters
588 0577 1
589 0578 1 None
590 0579 1
591 0580 1 Output parameters
592 0581 1
593 0582 1 qualifier$flags = Bitmap indicating which qualifiers are present
594 0583 1
595 0584 1 ----
596 0585 1
597 0586 2 BEGIN
598 0587 2
599 0588 2
600 0589 2 Initialize mask holding qualifier option bits
601 0590 2
602 0591 2 qualifier$flags = 0;
603 0592 2
604 0593 2
605 0594 2 Look for qualifiers and set bits accordingly
606 0595 2
607 0596 2 IF CLISPPRESENT (log_desc)
608 0597 2 THEN
609 0598 2 qualifier$flags [qual_log] = 1;
610 0599 2
611 0600 2 IF CLISPPRESENT (confirm_desc)
612 0601 2 THEN
613 0602 2 qualifier$flags [qual_confirm] = 1;
614 0603 2
615 0604 2 return true;
616 0605 1 END;
```

000C 00000 GET_UNLOCKQLS:						
53	0000'	CF	9E 00002	.WORD	Save R2,R3	0567
52	00000000G	00	9E 00007	MOVAB	QUALIFIERS\$FLAGS, R3	
		63	D4 0000E	MOVAB	CLISPPRESENT, R2	
	0000'	CF	9F 00010	CLRL	QUALIFIERS\$FLAGS	0591
62		01	FB 00014	PUSHAB	LOG_DESC	0596
03		50	E9 00017	CALLS	#1, CLISPPRESENT	
63		01	88 0001A	BLBC	R0, 1\$	
	0000'	CF	9F 0001D	BISB2	#1, QUALIFIERS\$FLAGS	0598
62		01	FB 00021	PUSHAB	CONFIRM_DESC	0600
03		50	E9 00024	CALLS	#1, CLISPPRESENT	
63		02	88 00027	BLBC	R0, 2\$	
				BISB2	#2, QUALIFIERS\$FLAGS	0602

UNLOCK
V04-000

D 6
16-Sep-1984 00:32:25
14-Sep-1984 12:10:06

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]UNLOCK.B32;1

Page 22
(12)

50

01 D0 0002A 2\$: MOVL #1, R0
04 0002D RET

; 0604
; 0605

; Routine Size: 46 bytes, Routine Base: \$CODE\$ + 01CF


```

: 618 0606 1 ROUTINE handler (signal_args, mechanism_args) =
: 619 0607 1
: 620 0608 1 ---
: 621 0609 1
: 622 0610 1 This condition handler gets control on any signalled
: 623 0611 1 condition in order to save the highest severity error
: 624 0612 1 to be returned by exit from the image.
: 625 0613 1
: 626 0614 1 Inputs:
: 627 0615 1
: 628 0616 1 signal_args = Address of signal argument list
: 629 0617 1 mechanism_args = Address of mechanism argument list
: 630 0618 1
: 631 0619 1 Outputs:
: 632 0620 1
: 633 0621 1 worst_error is updated with highest severity error.
: 634 0622 1
: 635 0623 1 ---
: 636 0624 1
: 637 0625 2 BEGIN
: 638 0626 2
: 639 0627 2 MAP
: 640 0628 2 signal_args: REF BBLOCK, ! Adr of signal arg list
: 641 0629 2 mechanism_args: REF BBLOCK; ! Adr of mech. arg list
: 642 0630 2
: 643 0631 2 LOCAL
: 644 0632 2 code: BBLOCK [LONG]; ! Condition code (longword)
: 645 0633 2
: 646 0634 2 code = .signal_args [chf$l_sig_name]; ! Get condition code
: 647 0635 2
: 648 0636 2 IF .code [sts$v_severity] GTR .worst_error [sts$v_severity]
: 649 0637 2 THEN
: 650 0638 2 worst_error = .code OR sts$m_inhib_msg; ! Set new worst error
: 651 0639 2
: 652 0640 2 ss$_resignal ! Continue signalling
: 653 0641 2
: 654 0642 1 END;
```

				0000	00000	HANDLER: .WORD	Save nothing	: 0606
		50	04	AC	D0	00002	MOVL	: 0634
		50	04	A0	D0	00006	MOVL	
51	0000'	CF		00	EF	0000A	EXTZV	: 0636
51		50		00	ED	00011	CMPZV	
				0A	15	00016	BLEQ	
	0000'	CF		8F	C9	00018	BISL3	: 0638
			50	10000000	8F	3C	MOVZWL	: 0642
			50	0918	8F	3C	MOVZWL	
				04	00027	1\$: RET		

; Routine Size: 40 bytes, Routine Base: \$CODE\$ + 01FD

```

: 656      0643 1 ROUTINE search_error (fab) =
: 657      0644 1
: 658      0645 1 |----
: 659      0646 1
: 660      0647 1 Functional description
: 661      0648 1
: 662      0649 1 This routine is called from RMS whenever an error
: 663      0650 1 occurs during an RMS file function call.
: 664      0651 1
: 665      0652 1 Input parameters
: 666      0653 1
: 667      0654 1 fab = Address of block used in the RMS call.
: 668      0655 1 fab$l_nam = pointer to name block
: 669      0656 1
: 670      0657 1 Output parameters
: 671      0658 1
: 672      0659 1 None
: 673      0660 1 |----
: 674      0661 1
: 675      0662 1
: 676      0663 2 BEGIN
: 677      0664 2
: 678      0665 2 MAP fab: REF BBLOCK; ! Define fab block format
: 679      0666 2 BIND nam = .fab[fab$l_nam]: BBLOCK; ! Define name block
: 680      0667 2
: 681      0668 2 LOCAL desc: VECTOR[2]; ! Temporary string descriptor
: 682      0669 2
: 683      0670 2 IF .nam[nam$b_rsl] NEQ 0 ! If result string nonblank,
: 684      0671 2 THEN BEGIN ! then display it
: 685      0672 2 desc[0] = .nam[nam$b_rsl];
: 686      0673 2 desc[1] = .nam[nam$l_rsa];
: 687      0674 2 END
: 688      0675 2 ELSE IF .nam[nam$b_esl] NEQ 0 ! Or if expanded name nonblank
: 689      0676 2 THEN BEGIN ! then display it
: 690      0677 2 desc[0] = .nam[nam$b_esl];
: 691      0678 2 desc[1] = .nam[nam$l_esa];
: 692      0679 2 END
: 693      0680 2 ELSE BEGIN
: 694      0681 2 desc[0] = .fab[fab$b_fns]; ! Otherwise, use original
: 695      0682 2 desc[1] = .fab[fab$l_fna]; ! name string in FAB
: 696      0683 2 END;
: 697      0684 2
: 698      0685 2 write_message(msg$searchfail,1,DESC, ! Output an error message
: 699      0686 2 .fab[fab$l_sts], ! with fab error code
: 700      0687 2 .fab[fab$l_stv]); ! and secondary code
: 701      0688 2
: 702      0689 2 RETURN (.fab[fab$l_sts]); ! Pass along the error
: 703      0690 2
: 704      0691 1 END;
```

0004 00000 SEARCH_ERROR:

SE

08 C2 00002

.WORD
SUBL2Save R2
#8, SP: 0643
:

UNLOCK
V04-000

G 6
16-Sep-1984 00:32:25
14-Sep-1984 12:10:06

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]UNLOCK.B32;1

Page 25
(14)

	52	04	AC	D0	00005	MOVL	FAB, R2	:	0666
	50	28	A2	D0	00009	MOVL	40(R2), R0	:	
		03	A0	95	0000D	TSTB	3(R0)	:	0670
			0B	13	00010	BEQL	1\$	:	
	6E	03	A0	9A	00012	MOVZBL	3(R0), DESC	:	0672
04	AE	04	A0	D0	00016	MOVL	4(R0), DESC+4	:	0673
			19	11	0001B	BRB	3\$	:	0670
		0B	A0	95	0001D	TSTB	11(R0)	:	0675
			0B	13	00020	BEQL	2\$	:	
	6E	0B	A0	9A	00022	MOVZBL	11(R0), DESC	:	0677
04	AE	0C	A0	D0	00026	MOVL	12(R0), DESC+4	:	0678
			09	11	0002B	BRB	3\$	:	0675
	6E	34	A2	9A	0002D	MOVZBL	52(R2), DESC	:	0681
04	AE	2C	A2	D0	00031	MOVL	44(R2), DESC+4	:	0682
	7E	0B	A2	7D	00036	MOVQ	8(R2), -(SP)	:	0687
		0B	AE	9F	0003A	PUSHAB	DESC	:	
			01	DD	0003D	PUSHL	#1	:	
		0092123A	8F	DD	0003F	PUSHL	#9572922	:	
00000000G	00		05	FB	00045	CALLS	#5, LIB\$SIGNAL	:	
	50	0B	A2	D0	0004C	MOVL	8(R2), R0	:	0689
			04	00050		RET		:	0691

; Routine Size: 81 bytes, Routine Base: \$CODE\$ + 0225

UNLOCK
V04-000

H 6
16-Sep-1984 00:32:25
14-Sep-1984 12:10:06

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]UNLOCK.B32;1

Page 26
(15)

: 706 0692 1 END
: 707 0693 0 ELUDOM

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes
\$SPLITS	72 NOVEC,NOWRT,	RD ,NOEXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
\$OWNS	724 NOVEC, WRT,	RD ,NOEXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
\$CODES	630 NOVEC,NOWRT,	RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
-\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	51	0	581	00:00.9
-\$255\$DUA28:[SYSLIB]CLIMAC.L32;1	14	0	0	9	00:00.1

COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS:UNLOCK/OBJ=OBJ\$:UNLOCK MSRC\$:UNLOCK/UPDATE=(ENH\$:UNLOCK)

: Size: 630 code + 796 data bytes
: Run Time: 00:15.2
: Elapsed Time: 00:51.0
: Lines/CPU Min: 2739
: Lexemes/CPU-Min: 23861
: Memory Used: 100 pages
: Compilation Complete

0060 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

TYPMAIN
LIS

COBPROLOG
REQ

COBACCDAT
LIS

COBACCDWK
LIS

UNLOCK
LIS

UTILSUBS
LIS

INTPAR
SDL

COBDEF
REQ

COBACCDAY
LIS

COBACCUCU
LIS

COBRTL

COBLNK
REQ

COBRTL
MAP