

[illegible]

Sy

LI

LI
LILI
LILI
LILI
LILI
LI

LI

LI

LI

LI
LILI
LILI
LI

11

LI
LI

LI

LI

21

100

LI
LI

11

100

10

111

LI

LI

LI
LI

11

10

1000

10

10

1

1

10

```

LL          IIIIII          SSSSSSSS
LL          IIIIII          SSSSSSSS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SSSSSS
LL          II             SSSSSS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LLLLLLLLLLLL IIIIII          SSSSSSSS
LLLLLLLLLLLL IIIIII          SSSSSSSS

```

```
1 0001 0 MODULE LIB$TIMER (
2 0002 0 IDENT = '1-011'
3 0003 0 ) =
4 0004 1 BEGIN
5 0005 1
6 0006 1 *****
7 0007 1 *
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 1 * ALL RIGHTS RESERVED.
11 0011 1 *
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0017 1 * TRANSFERRED.
18 0018 1 *
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0021 1 * CORPORATION.
22 0022 1 *
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0025 1 *
26 0026 1 *
27 0027 1 *****
28 0028 1
29 0029 1
30 0030 1 ++
31 0031 1 FACILITY: Run Time Library - User Callable
32 0032 1
33 0033 1 ABSTRACT:
34 0034 1
35 0035 1 This module implements the RTL Timing Facility, replacing
36 0036 1 subroutines TIMRB and TIMRE. It consists of four routines:
37 0037 1 LIB$INIT_TIMER, LIB$SHOW_TIMER, LIB$STAT_TIMER and LIB$FREE_TIMER.
38 0038 1
39 0039 1 ENVIRONMENT: User mode, can be made AST reentrant.
40 0040 1
41 0041 1 AUTHOR: Steven B. Lionel, CREATION DATE: 01-Dec-1978
42 0042 1
43 0043 1 MODIFIED BY:
44 0044 1
45 0045 1 SBL 01-Dec-1978 : VERSION 01
46 0046 1 1-001 - Original. SBL 01-Dec-1978
47 0047 1 1-002 - Fix error in calculating CPU HOURS. SBL 02-Jan-1979
48 0048 1 1-003 - Correct some typos in comments and rearrange the module
49 0049 1 header to conform to the other RTL modules. JBS 30-JUL-1979
50 0050 1 1-004 - Fix some more comments. SBL 6-August-1979
51 0051 1 1-005 - FAO_STRING_0 was missing a descriptor! SBL 29-Oct-1979
52 0052 1 1-006 - Changed the JPIP_PARAMS to be read only by putting them in
53 0053 1 the LIB$CODE psect. Also added EDIT field and updated
54 0054 1 the copyright date. LB 27-Aug-1981
55 0055 1 1-007 - Make text for SHOW_TIMER of all values shorter, so that it won't
56 0056 1 tend to overflow an 80-column screen. Remove the access test
57 0057 1 on the handle block so that users who set watchpoints won't get
```

```
.. 58      0058 1 | unnecessarily upset. SBL 2-Nov-1981
.. 59      0059 1 | 1-008 - Use LIB$GET_EF to get an event flag number for the call to
.. 60      0060 1 | $GETJPI. SBL 30-Nov-1981
.. 61      0061 1 | 1-009 - Use $GETJPIW to ensure synchronous operation. DG 26-Oct-1983
.. 62      0062 1 | 1-010 - Conditionalize setting of LSTORAGE[0] to eliminate non-zero
.. 63      0063 1 | read-only data in a copy-on-ref page, a performance improvement.
.. 64      0064 1 | MDL 6-Jul-1984
.. 65      0065 1 | 1-011 - Fix 1-010. Change INITIAL attribute in declaration of STORAGE
.. 66      0066 1 | to contain only binary zeroes, move INITIALIZED flag from LOCAL
.. 67      0067 1 | to OWN storage, move test of flag to correct branch of IF statement.
.. 68      0068 1 | SSA 8-Aug-1984
.. 69      0069 1 | --
.. 70      0070 1 |
.. 71      0071 1 | !<BLF/PAGE>
```

```
73 0072 1 |
74 0073 1 | SWITCHES:
75 0074 1 |
76 0075 1 |
77 0076 1 | SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
78 0077 1 |
79 0078 1 |
80 0079 1 | LINKAGES:
81 0080 1 |
82 0081 1 |     NONE
83 0082 1 |
84 0083 1 | TABLE OF CONTENTS:
85 0084 1 |
86 0085 1 |
87 0086 1 | FORWARD ROUTINE
88 0087 1 |     LIB$INIT_TIMER,           ! Initialize timing statistics
89 0088 1 |     LIB$SHOW_TIMER,          ! Output statistics
90 0089 1 |     LIB$STAT_TIMER,          ! Return statistic values
91 0090 1 |     LIB$FREE_TIMER,          ! Deallocate statistics storage block
92 0091 1 |     LIB$GETJPI;              ! Local routine to acquire statistics from system
93 0092 1 |
94 0093 1 |
95 0094 1 | INCLUDE FILES:
96 0095 1 |
97 0096 1 |
98 0097 1 | REQUIRE 'RTLIN:RTLPSECT';    ! DECLARE_PSECTS macro
99 0192 1 |
100 0193 1 | LIBRARY 'RTLSTARLE';
101 0194 1 |
102 0195 1 |
103 0196 1 | EQUATED SYMBOLS:
104 0197 1 |
105 0198 1 |
106 0199 1 | LITERAL
107 0200 1 |     N_OF_VALUES = 5,         ! Number of values desired
108 0201 1 |     BLOCKSIZE = (N_OF_VALUES + 2)*%UPVAL, ! Length of storage block in bytes
109 0202 1 |     LISTSIZE = N_OF_VALUES*3 - 2; ! Length of JPI argument list in longwords
110 0203 1 |
111 0204 1 | DECLARE_PSECTS (LIB);        ! Define psepts
112 0205 1 |
113 0206 1 |
114 0207 1 | OWN STORAGE:
115 0208 1 |
116 0209 1 |
117 0210 1 | OWN
118 0211 1 |     STORAGE : VECTOR [N_OF_VALUES + 2]
119 0212 1 |         INITIAL (BYTE (LONG(0), REP N_OF_VALUES OF LONG(0), LONG (0))),
120 0213 1 |         INITIALIZED : INITIAL(0); ! indicates whether STORAGE initialized
121 0214 1 |
122 0215 1 | !+
123 0216 1 | JPIPARAMS used to reside in the above OWN storage section, but in order
124 0217 1 | to make it read-only, it has been added into the _LIB$CODE psect, which
125 0218 1 | is already set up as read-only.
126 0219 1 | !-
127 0220 1 |
128 0221 1 | OWN
129 0222 1 |     JPIPARAMS : VECTOR [LISTSIZE] PSECT (_LIB$CODE) INITIAL (WORD (%UPVAL), ! Buffer length
```

```
130 0223 1 WORD (JPI$_PAGEFLTS),      | JPI code
131 0224 1 LONG (0),                  | Buffer address
132 0225 1 LONG (0),                  | Unused here
133 0226 1 WORD (%UPVAL),              | Repeat for other
134 0227 1 WORD (JPI$_CPUTIM),          | codes
135 0228 1 LONG (0), LONG (0), WORD (%UPVAL), WORD (JPI$_BUFIO), LONG (0), LONG (0), WORD (%UPVAL), WORD (
136 0229 1 JPI$_DIRIO), LONG (0), LONG (0), LONG (0)); ! Terminate list
137 0230 1
138 0231 1 |
139 0232 1 | EXTERNAL REFERENCES:
140 0233 1 |
141 0234 1 |
142 0235 1 EXTERNAL ROUTINE
143 0236 1 SYSS$AO,                    | Formats ASCII output
144 0237 1 LIB$PUT_OUTPUT,              | Sends string to SYSS$OUTPUT
145 0238 1 LIB$GET_EF,                  | Get event flag number
146 0239 1 LIB$FREE_EF,                 | Free event flag number
147 0240 1 LIB$FREE_VM,                 | Frees virtual memory
148 0241 1 LIB$GET_VM;                  | Gets virtual memory
149 0242 1
150 0243 1 EXTERNAL LITERAL
151 0244 1 LIB$_INVARG;                 | Invalid argument error condition
152 0245 1
```

```

154 0246 1 GLOBAL ROUTINE LIB$INIT_TIMER (
155 0247 1 HANDLE
156 0248 1 ) =
157 0249 1
158 0250 1 ++
159 0251 1 FUNCTIONAL DESCRIPTION:
160 0252 1
161 0253 1 This routine gets from the operating system the current
162 0254 1 values of specified times and counts, and stores them
163 0255 1 for future use by other RTL timer routines. Depending
164 0256 1 on the optional argument, "handle", the values are stored
165 0257 1 in one of three places. See FORMAL PARAMETERS for more
166 0258 1 details.
167 0259 1
168 0260 1 CALLING SEQUENCE:
169 0261 1
170 0262 1 status.wlc.v = LIB$INIT_TIMER ([handle.ml])
171 0263 1
172 0264 1 FORMAL PARAMETERS:
173 0265 1
174 0266 1 handle - Optional. Determines where the values
175 0267 1 of times and counts will be stored.
176 0268 1
177 0269 1 If missing, they will be stored in OWN
178 0270 1 storage. This call is not
179 0271 1 reentrant.
180 0272 1
181 0273 1 If zero, a block of dynamic heap storage
182 0274 1 will be allocated by a call to LIB$GET_VM,
183 0275 1 the values placed in that block and the
184 0276 1 address of the block returned in "handle".
185 0277 1 This method is AST reentrant.
186 0278 1
187 0279 1 If non zero, it is considered to be the
188 0280 1 address of a storage block previously
189 0281 1 allocated by a call to LIB$INIT_TIMER. If
190 0282 1 so, the block is reused, and fresh times
191 0283 1 and counts are stored in it.
192 0284 1
193 0285 1 IMPLICIT INPUTS:
194 0286 1
195 0287 1 If "handle" is non-zero, the block of storage it refers to is
196 0288 1 assumed to have been initialized by a previous call to
197 0289 1 LIB$INIT_TIMER.
198 0290 1
199 0291 1 IMPLICIT OUTPUTS:
200 0292 1
201 0293 1 Upon exit, the block of storage referred to by "handle" will
202 0294 1 contain the times and counts returned by a call to SYS$GETJPI.
203 0295 1
204 0296 1 ROUTINE VALUE:
205 0297 1 COMPLETION CODES:
206 0298 1
207 0299 1 SS$ NORMAL - Successful completion
208 0300 1 LIB$_INVARG - Invalid argument to routine. "handle"
209 0301 1 is non-zero and the block it refers to
210 0302 1 was not initialized on a previous call

```

```
211 0303 1 |
212 0304 1 | LIB$INSVIRMEM - to LIB$INIT_TIMER.
213 0305 1 | - "handle" is present and non-zero but there
214 0306 1 | is insufficient virtual memory available
215 0307 1 | LIB$ADBLOSIZ - This indicates a coding error in the RTL.
216 0308 1 | \This error should NEVER happen!\
217 0309 1 |
218 0310 1 | SIDE EFFECTS:
219 0311 1 |
220 0312 1 | If "handle" is present, and zero, a block of dynamic memory
221 0313 1 | is allocated.
222 0314 1 |
223 0315 1 | --
224 0316 1 |
225 0317 2 | BEGIN
226 0318 2 |
227 0319 2 | LOCAL
228 0320 2 | LSTORAGE : REF VECTOR [N_OF_VALUES + 2], ! Points to storage block
229 0321 2 | STATUS; ! Return status from functions
230 0322 2 |
231 0323 2 | BUILTIN
232 0324 2 | NULLPARAMETER;
233 0325 2 |
234 0326 2 | +
235 0327 2 | If "handle" was not specified, then use OWN storage.
236 0328 2 | Else either allocate some or reuse a block, depending on the
237 0329 2 | value of "handle"
238 0330 2 | -
239 0331 2 |
240 0332 3 | IF (NULLPARAMETER (1))
241 0333 2 | THEN
242 0334 3 | BEGIN
243 0335 3 | LSTORAGE = STORAGE; ! Use OWN storage
244 0336 4 | IF ( NOT .INITIALIZED)
245 0337 3 | THEN
246 0338 4 | BEGIN
247 0339 4 | LSTORAGE [0] = %ASCII'TIMR';
248 0340 4 | INITIALIZED = 1
249 0341 3 | END;
250 0342 3 | END
251 0343 2 | ELSE
252 0344 3 | BEGIN
253 0345 3 |
254 0346 4 | IF (..HANDLE EQL 0)
255 0347 3 | THEN
256 0348 3 | ! Allocate dynamic storage
257 0349 4 | BEGIN
258 0350 4 | STATUS = LIB$GET_VM (%REF (BLOCKSIZE), .HANDLE);
259 0351 4 |
260 0352 4 | IF ( NOT .STATUS) THEN RETURN (.STATUS);
261 0353 4 |
262 0354 4 | LSTORAGE = ..HANDLE;
263 0355 4 | +
264 0356 4 | Set first longword of storage block to 'TIMR'. This is a quick
265 0357 4 | consistency check to protect against being passed a block not
266 0358 4 | previously initialized here.
267 0359 4 | -
```

```

: 268      0360      4      LSTORAGE [0] = %ASCII'TIMR';
: 269      0361      END;
: 270      0362
: 271      0363      LSTORAGE = ..HANDLE;      ! Reuse previous block
: 272      0364
: 273      0365      IF (.LSTORAGE [0] NEQU %ASCII'TIMR') THEN RETURN (LIB$_INVARG); ! Consistency check
: 274      0366
: 275      0367      END;
: 276      0368
: 277      0369      STATUS = LIB$$GETJPI (LSTORAGE [1]);      ! Get times/counts and store in storage block 1-5
: 278      0370      RETURN (.STATUS);
: 279      0371      1      END;      ! End of routine LIB$INIT_TIMER
```

.TITLE LIB\$TIMER

.IDENT \1-011\

.PSECT _LIB\$DATA,NOEXE, PIC,2

```
00000000 00000 STORAGE: .LONG 0
00000000# 00004 .LONG 0[5]
00000000 00018 .LONG 0
00000000 0001C INITIALIZED:
. LONG 0
```

.PSECT _LIB\$CODE,NOWRT, SHR, PIC,2

```
0004 00000 JPIPARAMS:
. WORD 4
040A 00002 .WORD 1034
00000000 00004 .LONG 0
00000000 00008 .LONG 0
0004 0000C .WORD 4
0407 0000E .WORD 1031
00000000 00010 .LONG 0
00000000 00014 .LONG 0
0004 00018 .WORD 4
040C 0001A .WORD 1036
00000000 0001C .LONG 0
00000000 00020 .LONG 0
0004 00024 .WORD 4
040B 00026 .WORD 1035
00000000 00028 .LONG 0
00000000 0002C .LONG 0
00000000 00030 .LONG 0
```

```
.EXTRN SYSS$FAO, LIB$PUT OUTPUT
.EXTRN LIB$GET_EF, LIB$FREE_EF
.EXTRN LIB$FREE_VM, LIB$GET_VM
.EXTRN LIB$_INVARG
```

```
54 00000000' 001C 00000
5E EF 9E 00002
04 C2 00009
6C 95 0000C
05 13 0000E
04 AC D5 00010
```

```
.ENTRY LIB$INIT_TIMER, Save R2,R3,R4
MOVAB INITIALIZED, R4
SUBL2 #4, SP
TSTB (AP)
BEQL 1$
TSTL 4(AP)
```

: 0246

: 0332

			13	12	00013		BNEQ	2\$		
	52	E4	A4	9E	00015	1\$:	MOVAB	STORAGE, LSTORAGE		0335
	45		64	E8	00019		BLBS	INITIALIZED, 4\$		0336
	62	524D4954	8F	D0	0001C		MOVL	#1380796756, (LSTORAGE)		0339
	64		01	D0	00023		MOVL	#1, INITIALIZED		0340
			39	11	00026		BRB	4\$		0332
	53	04	AC	D0	00028	2\$:	MOVL	HANDLE, R3		0346
			63	D5	0002C		TSTL	(R3)		
			1D	12	0002E		BNEQ	3\$		
			53	DD	00030		PUSHL	R3		0350
04	AE		1C	D0	00032		MOVL	#28, 4(SP)		
		04	AE	9F	00036		PUSHAB	4(SP)		
00000000G	00		02	FB	00039		CALLS	#2, LIB\$GET_VM		
	26		50	E9	00040		BLBC	STATUS, 5\$		0352
	52		63	D0	00043		MOVL	(R3), LSTORAGE		0354
	62	524D4954	8F	D0	00046		MOVL	#1380796756, (LSTORAGE)		0360
	52		63	D0	0004D	3\$:	MOVL	(R3), LSTORAGE		0363
524D4954	8F		62	D1	00050		CMPL	(LSTORAGE), #1380796756		0365
			08	13	00057		BEQL	4\$		
	50	00000000G	8F	D0	00059		MOVL	#LIB\$_INVARG, R0		
				04	00060		RET			
			A2	9F	00061	4\$:	PUSHAB	4(LSTORAGE)		0369
0000V	CF		01	FB	00064		CALLS	#1, LIB\$\$GETJPI		
			04	00069	5\$:		RET			0371

; Routine Size: 106 bytes, Routine Base: _LIB\$CODE + 0034

; 280 0372 1

```
282 0373 1 GLOBAL ROUTINE LIB$SHOW_TIMER (HANDLE,
283 0374 1     CODE,
284 0375 1     ACTION,
285 0376 1     USERARG
286 0377 1 ) =
287 0378 1
288 0379 1
289 0380 1 ++
290 0381 1 FUNCTIONAL DESCRIPTION:
291 0382 1
292 0383 1 LIB$SHOW_TIMER is used to obtain the accumulated times/counts
293 0384 1 since the last time LIB$INIT_TIMER was called. In the default
294 0385 1 mode, with neither CODE nor ACTION specified in the call, the
295 0386 1 routine will output to SYS$OUTPUT a line giving the following
296 0387 1 five items of information:
297 0388 1     ELAPSED = hhhh:mm:ss.cc - Elapsed real time
298 0389 1     CPU = hhhh:mm:ss.cc - Elapsed CPU time
299 0390 1     BUFIO = nnnn - Count of buffered I/O operations
300 0391 1     DIRIO = nnnn - Count of direct I/O operations
301 0392 1     PAGEFLT = nnnn - Count of pagefaults
302 0393 1
303 0394 1 Optionally, only one of the five statistics can be output to
304 0395 1 SYS$OUTPUT and/or the line of information can be passed to a
305 0396 1 user-specified "action routine", for processing different from
306 0397 1 the default.
307 0398 1
308 0399 1
309 0400 1 CALLING SEQUENCE:
310 0401 1
311 0402 1     status.wlc.v = LIB$SHOW_TIMER ( [[[handle.rl.r], code.rl.r],
312 0403 1                                     action.flc.rp], userarg.rl.v)
313 0404 1
314 0405 1
315 0406 1 FORMAL PARAMETERS:
316 0407 1
317 0408 1     handle - (Optional) If specified, must be the same value as
318 0409 1 returned by a previous call to LIB$INIT_TIMER.
319 0410 1 If omitted, the routine's OWN storage will be used.
320 0411 1 If handle is omitted, and LIB$INIT_TIMER has not previously
321 0412 1 been called, Elapsed time will show the actual time-of-day
322 0413 1 and the remaining values will be those accumulated since process login.
323 0414 1
324 0415 1     code - (Optional) Specifies the particular statistic desired.
325 0416 1 If given, it must be one of the following values:
326 0417 1     LIB$K_TMRCLP = 1 = Elapsed time
327 0418 1     LIB$K_TMRCPU = 2 = CPU time
328 0419 1     LIB$K_TMRBUF = 3 = Buffered I/O
329 0420 1     LIB$K_TMRDIR = 4 = Direct I/O
330 0421 1     LIB$K_TMRFLT = 5 = Page faults
331 0422 1 If omitted or zero, all five statistics will be returned
332 0423 1 on one line.
333 0424 1
334 0425 1     action
335 0426 1     userarg - (Optional) These two parameters allow the user to direct
336 0427 1 the output of LIB$SHOW_TIMER somewhere other than SYS$OUTPUT.
337 0428 1 If "action" is given, it is the address of a function
338 0429 1 procedure to call. The arguments to this function are
```

```

339      0430 1 | described below. The function should return either
340      0431 1 | a success or failure condition value, which will be
341      0432 1 | returned as the value of LIB$SHOW_TIMER.
342      0433 1 |
343      0434 1 | CALLING SEQUENCE OF ACTION ROUTINE:
344      0435 1 |
345      0436 1 |     status.wlc.v = (action) (out_string.rt.ds [, userarg.rl.v])
346      0437 1 |
347      0438 1 |     out_string      - A descriptor of a fixed length string
348      0439 1 |                      containing the statistics desired. It
349      0440 1 |                      is formatted exactly as it would be if
350      0441 1 |                      output to SYS$OUTPUT. The leading character
351      0442 1 |                      is blank.
352      0443 1 |
353      0444 1 |     userarg         - If 'userarg' is passed to LIB$SHOW_TIMER,
354      0445 1 |                      it is passed directly on to the action
355      0446 1 |                      routine. Note that this is passed BY VALUE
356      0447 1 |                      both to LIB$SHOW_TIMER and to the action routine.
357      0448 1 |
358      0449 1 |
359      0450 1 | IMPLICIT INPUTS:
360      0451 1 |
361      0452 1 |     It is assumed that LIB$INIT_TIMER has previously been called, and
362      0453 1 |     that the results of that call are stored in either OWN storage or
363      0454 1 |     a block pointed to by 'handle'.
364      0455 1 |
365      0456 1 | IMPLICIT OUTPUTS:
366      0457 1 |
367      0458 1 |     NONE
368      0459 1 |
369      0460 1 | ROUTINE VALUE:
370      0461 1 | COMPLETION CODES:
371      0462 1 |
372      0463 1 |     SS$NORMAL      - Successful completion.
373      0464 1 |     LIB$_INVARG     - Invalid arguments. This can be caused by an
374      0465 1 |                      invalid value for 'code' or 'handle'.
375      0466 1 |     Other codes as may be returned by LIB$PUT_OUTPUT or the user's
376      0467 1 |     action routine.
377      0468 1 |
378      0469 1 | SIDE EFFECTS:
379      0470 1 |
380      0471 1 |     NONE
381      0472 1 |
382      0473 1 | --
383      0474 1 |
384      0475 2 | BEGIN
385      0476 2 |
386      0477 2 | BUILTIN
387      0478 2 |     ACTUALCOUNT,
388      0479 2 |     NULLPARAMETER,
389      0480 2 |     SUBM,
390      0481 2 |     ASHQ,
391      0482 2 |     EDIV;
392      0483 2 |
393      0484 2 | LOCAL
394      0485 2 |     STATUS,
395      0486 2 |     TEMPCODE,

```

```

! Returned condition values
! Local value of CODE

```

```
396 0487 2 LSTORAGE, | Will contain address of storage block
397 0488 2 TSTORAGE : VECTOR [N OF VALUES + 1], | Gets current times/counts
398 0489 2 FAO_DESC : BLOCK [8, BYTE], | Descriptor for FAO string
399 0490 2 OUT_DESC : BLOCK [8, BYTE], | Output string descriptor
400 0491 2 OUT_STRING : BLOCK [100, BYTE], | Output string
401 0492 2 CPU_HOURS : BLOCK [2], | Elapsed CPU hours
402 0493 2 CPU_MINUTES, | Elapsed CPU minutes
403 0494 2 CPU_SECONDS, | Elapsed CPU seconds
404 0495 2 CPU_MILLS; | Elapsed CPU 10 mills
405 0496 2
406 0497 2 MAP
407 0498 2 LSTORAGE : REF VECTOR [N_OF_VALUES + 2]; | Local name for storage
408 0499 2
409 0500 2 BIND
410 0501 2 PAGEFAULTS = TSTORAGE [0],
411 0502 2 CPUTIMX = TSTORAGE [1],
412 0503 2 BUFIO = TSTORAGE [2],
413 0504 2 DIRIO = TSTORAGE [3],
414 0505 2 ELAPSED_TIME = TSTORAGE [4];
415 0506 2
416 0507 2 !+
417 0508 2 !- Declare the FAO control strings and lengths.
418 0509 2 !-
419 0510 2
420 0511 2 BIND FAO_STRING_0 = UPLIT BYTE (
421 0512 2 ' ELAPSED: !%T CPU: !UL: !2ZL: !2ZL: !2ZL BUFIO: !UL DIRIO: !UL FAULTS: !UL ');
422 L 0513 2 LITERAL FAO_LENGTH_0 = %CHARCOUNT (
423 0514 2 ' ELAPSED: !%T CPU: !UL: !2ZL: !2ZL: !2ZL BUFIO: !UL DIRIO: !UL FAULTS: !UL ');
424 0515 2 BIND FAO_STRING_1 = UPLIT BYTE (' ELAPSED TIME = !%T');
425 0516 2 LITERAL FAO_LENGTH_1 = %CHARCOUNT (' ELAPSED TIME = !%T');
426 0517 2 BIND FAO_STRING_2 = UPLIT BYTE (' CPU TIME = !UL: !2ZL: !2ZL: !2ZL');
427 0518 2 LITERAL FAO_LENGTH_2 = %CHARCOUNT (' CPU TIME = !UL: !2ZL: !2ZL: !2ZL');
428 0519 2 BIND FAO_STRING_3 = UPLIT BYTE (' BUFFERED I/O COUNT = !UL');
429 0520 2 LITERAL FAO_LENGTH_3 = %CHARCOUNT (' BUFFERED I/O COUNT = !UL');
430 0521 2 BIND FAO_STRING_4 = UPLIT BYTE (' DIRECT I/O COUNT = !UL');
431 0522 2 LITERAL FAO_LENGTH_4 = %CHARCOUNT (' DIRECT I/O COUNT = !UL');
432 0523 2 BIND FAO_STRING_5 = UPLIT BYTE (' PAGE FAULT COUNT = !UL');
433 0524 2 LITERAL FAO_LENGTH_5 = %CHARCOUNT (' PAGE FAULT COUNT = !UL');
434 0525 2
435 0526 2 !+
436 0527 2 !- Set up FAO string descriptor
437 0528 2 !-
438 0529 2
439 0530 2 FAO_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
440 0531 2 FAO_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
441 0532 2
442 0533 2 !+
443 0534 2 !- Set up out_string descriptor
444 0535 2 !-
445 0536 2 OUT_DESC [DSC$W_LENGTH] = 100;
446 0537 2 OUT_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
447 0538 2 OUT_DESC [DSC$B_CLASS] = DSC$K_CLASS_S;
448 0539 2
449 0540 2 IF (NULLPARAMETER (1)) | "handle" not specified?
450 0541 2 THEN
451 0542 2 LSTORAGE = STORAGE | Use OWN storage
452 0543 2 ELSE
```

```
453 0544 2 LSTORAGE = ..HANDLE; ! Use handle
454 0545
455 0546
456 0547 IF (.LSTORAGE [0] NEQU %ASCII'TIMR') THEN RETURN (LIB$_INVARG); ! Invalid storage block
457 0548
458 0549 STATUS = LIB$$GETJPI (TSTORAGE); ! Get values
459 0550
460 0551 IF ( NOT .STATUS) THEN RETURN (.STATUS);
461 0552
462 0553 !+
463 0554 Compute values elapsed since call to LIB$INIT_TIMER
464 0555 !-
465 0556 PAGEFAULTS = .PAGEFAULTS - .LSTORAGE [1];
466 0557 CPU_HOURS = .CPUTIMX - .LSTORAGE [2];
467 0558 BUFIO = .BUFIO - .LSTORAGE [3];
468 0559 DIRIO = .DIRIO - .LSTORAGE [4];
469 0560 SUBM (2, LSTORAGE [5], ELAPSED_TIME, ELAPSED_TIME);
470 0561 CPU_HOURS + 4 = 0; ! Clear high order part of CPU time
471 0562 EDIV (%REF (100), CPU_HOURS, CPU_HOURS, CPU_MILLS);
472 0563 EDIV (%REF (60), CPU_HOURS, CPU_HOURS, CPU_SECONDS);
473 0564 EDIV (%REF (60), CPU_HOURS, CPU_HOURS, CPU_MINUTES);
474 0565 !+
475 0566 Create out_string
476 0567 !-
477 0568 OUT_DESC [DSC$A_POINTER] = OUT_STRING;
478 0569 !+
479 0570 Now "output" the result, depending on the value of CODE.
480 0571 !-
481 0572
482 0573 IF (NULLPARAMETER (2)) THEN TEMPCODE = 0 ELSE TEMPCODE = ..CODE;
483 0574
484 0575 CASE .TEMPCODE FROM 0 TO N_OF_VALUES OF
485 0576 SET
486 0577
487 0578 [0] : ! Default case, all five values
488 0579 BEGIN
489 0580 FAO_DESC [DSC$A_POINTER] = FAO_STRING 0;
490 0581 FAO_DESC [DSC$W_LENGTH] = FAO_LENGTH 0;
491 0582 STATUS = SYSSFAO (FAO_DESC, OUT_DESC [DSC$W_LENGTH], OUT_DESC, ELAPSED_TIME, .CPU_HOURS,
492 0583 .CPU_MINUTES, .CPU_SECONDS, .CPU_MILLS, .BUFIO, .DIRIO, .PAGEFAULTS);
493 0584 END;
494 0585
495 0586 [1] : ! Elapsed time only
496 0587 BEGIN
497 0588 FAO_DESC [DSC$A_POINTER] = FAO_STRING 1;
498 0589 FAO_DESC [DSC$W_LENGTH] = FAO_LENGTH 1;
499 0590 STATUS = SYSSFAO (FAO_DESC, OUT_DESC [DSC$W_LENGTH], OUT_DESC, ELAPSED_TIME);
500 0591 END;
501 0592
502 0593 [2] : ! CPU time only
503 0594 BEGIN
504 0595 FAO_DESC [DSC$A_POINTER] = FAO_STRING 2;
505 0596 FAO_DESC [DSC$W_LENGTH] = FAO_LENGTH 2;
506 0597 STATUS = SYSSFAO (FAO_DESC, OUT_DESC [DSC$W_LENGTH], OUT_DESC, .CPU_HOURS, .CPU_MINUTES,
507 0598 .CPU_SECONDS, .CPU_MILLS);
508 0599 END;
509 0600
```

```

: 510      0601      2
: 511      0602
: 512      0603
: 513      0604
: 514      0605
: 515      0606
: 516      0607
: 517      0608
: 518      0609
: 519      0610
: 520      0611
: 521      0612
: 522      0613
: 523      0614
: 524      0615
: 525      0616
: 526      0617
: 527      0618
: 528      0619
: 529      0620
: 530      0621
: 531      0622
: 532      0623
: 533      0624
: 534      0625
: 535      0626
: 536      0627
: 537      0628
: 538      0629
: 539      0630
: 540      0631
: 541      0632
: 542      0633
: 543      0634
: 544      0635
: 545      0636
: 546      0637
: 547      0638
: 548      0639
: 549      0640      1

[3] :                                     ! Buffered I/O count only
      BEGIN
      FAO_DESC [DSC$A_POINTER] = FAO_STRING_3;
      FAO_DESC [DSC$W_LENGTH] = FAO_LENGTH_3;
      STATUS = SYSSFAO (FAO_DESC, OUT_DESC[DSC$W_LENGTH], OUT_DESC, .BUFIO);
      END;

[4] :                                     ! Direct I/O count only
      BEGIN
      FAO_DESC [DSC$A_POINTER] = FAO_STRING_4;
      FAO_DESC [DSC$W_LENGTH] = FAO_LENGTH_4;
      STATUS = SYSSFAO (FAO_DESC, OUT_DESC[DSC$W_LENGTH], OUT_DESC, .DIRIO);
      END;

[5] :                                     ! Page faults only
      BEGIN
      FAO_DESC [DSC$A_POINTER] = FAO_STRING_5;
      FAO_DESC [DSC$W_LENGTH] = FAO_LENGTH_5;
      STATUS = SYSSFAO (FAO_DESC, OUT_DESC[DSC$W_LENGTH], OUT_DESC, .PAGEFAULTS);
      END;

[OUTRANGE] :
      STATUS = LIB$_INVARG;
      TES;

IF ( NOT .STATUS) THEN RETURN (.STATUS);

IF (NULLPARAMETER (3))
THEN
  STATUS = LIB$PUT_OUTPUT (OUT_DESC)
ELSE
  IF (NULLPARAMETER (4))
  THEN
    STATUS = (.ACTION) (OUT_DESC)
  ELSE
    STATUS = (.ACTION) (OUT_DESC, .USERARG);

RETURN (.STATUS);
END;
```

```

20 20 54 25 21 20 3A 44 45 53 50 41 4C 45 20 0009E P.AAA: .ASCII \ ELAPSED: !%T CPU: !UL:!2ZL:!2ZL.!2ZL \
21 3A 4C 5A 32 21 3A 4C 55 21 20 3A 55 50 43 000AD
20 20 4C 5A 32 21 2E 4C 5A 32 000BC
52 49 44 20 20 4C 55 21 20 3A 4F 49 46 55 42 000C6 .ASCII \BUFIO: !UL DIRIO: !UL FAULTS: !UL \
53 54 4C 55 41 46 20 20 4C 55 21 20 3A 4F 49 000D5
20 4C 55 21 20 3A 000E4
3D 20 45 4D 49 54 20 44 45 53 50 41 4C 45 20 000EA P.AAB: .ASCII \ ELAPSED TIME = !%T\
54 25 21 20 000F9
4C 55 21 20 3D 20 45 4D 49 54 20 55 50 43 20 000FD P.AAC: .ASCII \ CPU TIME = !UL:!2ZL:!2ZL.!2ZL\
4C 5A 32 21 2E 4C 5A 32 21 3A 4C 5A 32 21 3A 0010C
43 20 4F 2F 49 20 44 45 52 45 46 46 55 42 20 0011B P.AAD: .ASCII \ BUFFERED I/O COUNT = !UL\
55 4F 43 20 4F 2F 49 20 54 43 45 52 49 44 20 0012A
4C 55 21 20 3D 20 54 4E 55 4F 00134 P.AAE: .ASCII \ DIRECT I/O COUNT = !UL\
4C 55 21 20 3D 20 54 4E 00143
```

55 4F 43 20 54 4C 55 41 46 20 45 47 41 50 20 0014B P.AAF: .ASCII \ PAGE FAULT COUNT = !UL\
4C 55 21 20 3D 20 54 4E 0015AFAO_STRING_0= P.AAA
FAO_STRING_1= P.AAB
FAO_STRING_2= P.AAC
FAO_STRING_3= P.AAD
FAO_STRING_4= P.AAE
FAO_STRING_5= P.AAF

56	00000000G	8F	D0	00002	.ENTRY	LIB\$SHOW TIMER, Save R2,R3,R4,R5,R6	0373
55	00000000G	00	9E	00009	MOVL	#LIB\$ INVARG, R6	
5E	FF6C	CE	9E	00010	MOVAB	SYSSFAO, R5	
76	AE 010E	8F	B0	00015	MOVAB	-148(SP), SP	
6C	AE 010E0064	8F	D0	0001B	MOVW	#270, FAO_DESC+2	0530
		6C	95	00023	MOVL	#17694820, OUT_DESC	0536
		05	13	00025	TSTB	(AP)	0540
	04	AC	D5	00027	BEQL	1\$	
		09	12	0002A	TSTL	4(AP)	
52	00000000'	EF	9E	0002C	BNEQ	2\$	
		04	11	00033	MOVAB	STORAGE, LSTORAGE	0542
52	04	BC	D0	00035	BRB	3\$	
524D4954	8F	62	D1	00039	MOVL	@HANDLE, LSTORAGE	0544
		04	13	00040	CMPL	(LSTORAGE), #1380796756	0547
50		56	D0	00042	BEQL	4\$	
		04	00045	MOVL	R6, R0		
	7C	AE	9F	00046	RET		
0000V	CF	01	FB	00049	PUSHAB	TSTORAGE	0549
	01	50	E8	0004E	CALLS	#1, LIB\$\$GETJPI	
		04	00051	BLBS	STATUS, 5\$		0551
	7C	AE	A2	C2	RET		
6E	EC	AD	A2	C3	SUBL2	4(LSTORAGE), PAGEFAULTS	0556
	F0	AD	A2	C2	SUBL3	8(LSTORAGE), CPUTIMX, CPU_HOURS	0557
	F4	AD	A2	C2	SUBL2	12(LSTORAGE), BUFIO	0558
	F8	AD	A2	C2	SUBL2	16(LSTORAGE), DIRIO	0559
	FC	AD	A2	C2	SUBL2	20(LSTORAGE), ELAPSED_TIME	0560
		18	A2	D9	SBWC	24(LSTORAGE), ELAPSED_TIME	
		04	AE	D4	CLRL	CPU_HOURS+4	0561
54	6E	6E	8F	7B	EDIV	#100, CPU_HOURS, CPU_HOURS, CPU_MILLS	0562
52	6E	6E	3C	7B	EDIV	#60, CPU_HOURS, CPU_HOURS, CPU_SECONDS	0563
53	6E	6E	3C	7B	EDIV	#60, CPU_HOURS, CPU_HOURS, CPU_MINUTES	0564
	70	AE	9E	00087	MOVAB	OUT_STRING, OUT_DESC+4	0568
		02	6C	91	CMPB	(APT), #2	0573
			05	1F	BLSSU	6\$	
		08	AC	D5	TSTL	8(AP)	
			04	12	BNEQ	7\$	
		51	D4	00096	CLRL	TEMPCODE	
		04	11	00098	BRB	8\$	
	51	08	BC	D0	MOVL	@CODE, TEMP CODE	
0069	05	00	51	CF	CASEL	TEMP CODE, #0, #5	0575
	004A	003B	0011	0009E	.WORD	10\$-9\$,-	
		0087	0078	000A2		11\$-9\$,-	
				000AA		12\$-9\$,-	
						14\$-9\$,-	
						15\$-9\$,-	
						16\$-9\$,-	

50		56	D0	000AE	MOVL	R6, STATUS	: 0623
		56	11	000B1	BRB	13\$	
78	AE	FE85	CF	9E 000B3	10\$: MOVAB	FAO_STRING_0, FAO_DESC+4	: 0580
74	AE	4C	8F	9B 000B9	MOVZBW	#76, FAO_DESC	: 0581
		7C	AE	DD 000BE	PUSHL	PAGEFAULTS	: 0583
	7E	F0	AD	7D 000C1	MOVQ	BUFIO, -(SP)	
			14	BB 000C5	PUSHR	#*M<R2,R4>	
			53	DD 000C7	PUSHL	CPU_MINUTES	
		18	AE	DD 000C9	PUSHL	CPU_HOURS	: 0582
		F8	AD	9F 000CC	PUSHAB	ELAPSED_TIME	
		D8	AD	9F 000CF	PUSHAB	OUT_DESC	
		D8	AD	9F 00CD2	PUSHAB	OUT_DESC	
		E0	AD	9F 000D5	PUSHAB	FAO_DESC	
65			0B	FB 000D8	CALLS	#11, SYSS\$FAO	
			65	11 000DB	BRB	18\$	: 0575
78	AE	FEA7	CF	9E 000DD	11\$: MOVAB	FAO_STRING_1, FAO_DESC+4	: 0588
74	AE		13	B0 000E3	MOVW	#19, FAO_DESC	: 0589
		F8	AD	9F 000E7	PUSHAB	ELAPSED_TIME	: 0590
			4A	11 000EA	BRB	17\$	
78	AE	FEAB	CF	9E 000EC	12\$: MOVAB	FAO_STRING_2, FAO_DESC+4	: 0595
74	AE		1E	B0 000F2	MOVW	#30, FAO_DESC	: 0596
			14	BB 000F6	PUSHR	#*M<R2,R4>	: 0598
			53	DD 000F8	PUSHL	CPU_MINUTES	: 0597
		0C	AE	DD 000FA	PUSHL	CPU_HOURS	
		7C	AE	9F 000FD	PUSHAB	OUT_DESC	
		D8	AD	9F 00100	PUSHAB	OUT_DESC	
		E0	AD	9F 00103	PUSHAB	FAO_DESC	
65			07	FB 00106	CALLS	#7, SYSS\$FAO	
			37	11 00109	13\$: BRB	18\$	: 0575
78	AE	FEAA	CF	9E 0010B	14\$: MOVAB	FAO_STRING_3, FAO_DESC+4	: 0603
74	AE		19	B0 00111	MOVW	#25, FAO_DESC	: 0604
		F0	AD	DD 00115	PUSHL	BUFIO	: 0605
			1C	11 00118	BRB	17\$	
78	AE	FEB4	CF	9E 0011A	15\$: MOVAB	FAO_STRING_4, FAO_DESC+4	: 0610
74	AE		17	B0 00120	MOVW	#23, FAO_DESC	: 0611
		F4	AD	DD 00124	PUSHL	DIRIO	: 0612
			0D	11 00127	BRB	17\$	
78	AE	FEBC	CF	9E 00129	16\$: MOVAB	FAO_STRING_5, FAO_DESC+4	: 0617
74	AE		17	B0 0012F	MOVW	#23, FAO_DESC	: 0618
		7C	AE	DD 00133	PUSHL	PAGEFAULTS	: 0619
		70	AE	9F 00136	17\$: PUSHAB	OUT_DESC	
		74	AE	9F 00139	PUSHAB	OUT_DESC	
		E0	AD	9F 0013C	PUSHAB	FAO_DESC	
65			04	FB 0013F	CALLS	#4, SYSS\$FAO	
31			50	E9 00142	18\$: BLBC	STATUS, 23\$	: 0626
03			6C	91 00145	CMPB	(AP), #3	: 0628
			05	1F 00148	BLSSU	19\$	
		0C	AC	D5 0014A	TSTL	12(AP)	
			0B	12 0014D	BNEQ	20\$	
00000000G	00	6C	AE	9F 0014F	19\$: PUSHAB	OUT_DESC	: 0630
			01	FB 00152	CALLS	#1, LIB\$PUT_OUTPUT	
			04	00159	RET		
	04		6C	91 0015A	20\$: CMPB	(AP), #4	: 0633
			05	1F 0015D	BLSSU	21\$	
		10	AC	D5 0015F	TSTL	16(AP)	
			08	12 00162	BNEQ	22\$	
		6C	AE	9F 00164	21\$: PUSHAB	OUT_DESC	: 0635

16-Sep-1984 01:18:20 VAX-11 BLISS-32 V4.0-742
14-Sep-1984 12:39:33 [LIBRTL.SRC]LIBTIMER.B32;1

Page 16
(4)

0C	BC		01	FB	00167		CALLS	#1, @ACTION
				04	0016B		RET	
		10	AC	DD	0016C	22\$:	PUSHL	USERARG
		70	AE	9F	0016F		PUSHAB	OUT_DESC
0C	BC		02	FB	00172		CALLS	#2, @ACTION
				04	00176	23\$:	RET	

0637
0640

: 550 0641 1

```
552 0642 1 GLOBAL ROUTINE LIB$STAT_TIMER (
553 0643 1     CODE
554 0644 1     VALUE
555 0645 1     HANDLE
556 0646 1 ) =
557 0647 1
558 0648 1 ++
559 0649 1 FUNCTIONAL DESCRIPTION:
560 0650 1
561 0651 1     LIB$STAT_TIMER returns to its caller one of five available
562 0652 1     statistics. Unlike LIB$SHOW_TIMER which formats the values
563 0653 1     for output, LIB$STAT_TIMER returns the value to a location
564 0654 1     specified as a parameter.
565 0655 1
566 0656 1     Only one of the five statistics can be returned by one call
567 0657 1     to LIB$STAT_TIMER. "code" must be one of the five values
568 0658 1     described for LIB$SHOW_TIMER. A "code" of zero is invalid.
569 0659 1
570 0660 1     It is important to note that elapsed time (LIB$K_TMRELP) is
571 0661 1     returned in the system quadword format. Therefore the receiving
572 0662 1     area should be 8 bytes long. All other values are longwords.
573 0663 1
574 0664 1 CALLING SEQUENCE:
575 0665 1
576 0666 1     status.wlc.v = LIB$STAT_TIMER (code.rl, value.wx [, handle.rl])
577 0667 1
578 0668 1 FORMAL PARAMETERS:
579 0669 1
580 0670 1     code - Specifies which statistic is to be returned. Allowed
581 0671 1           values are:
582 0672 1             LIB$K_TMRELP = 1 = Elapsed time - system quadword format
583 0673 1             LIB$K_TMRCPU = 2 = CPU time - .01 second units
584 0674 1             LIB$K_TMRBUF = 3 = Buffered I/O
585 0675 1             LIB$K_TMRDIR = 4 = Direct I/O
586 0676 1             LIB$K_TMRFLT = 5 = Page faults
587 0677 1           Note that it is invalid to omit "code": or to give
588 0678 1           a "code" of zero.
589 0679 1
590 0680 1     value - The location to store the result. All values are
591 0681 1             longword integers except elapsed time, which is a
592 0682 1             quadword. See VAX/VMS System Services Reference Manual
593 0683 1             for more details on the system quadword time format.
594 0684 1
595 0685 1     handle - (Optional) If specified, it must be the value returned
596 0686 1               by a previous call to LIB$INIT_TIMER. Otherwise, OWN
597 0687 1               storage will be used.
598 0688 1
599 0689 1
600 0690 1 IMPLICIT INPUTS:
601 0691 1
602 0692 1     It is assumed that LIB$INIT_TIMER has previously been called and
603 0693 1     that the "handle" argument to LIB$INIT_TIMER is the same as in
604 0694 1     the call to LIB$STAT_TIMER.
605 0695 1
606 0696 1 IMPLICIT OUTPUTS:
607 0697 1
608 0698 1
```

```

609 0699 1 | NONE
610 0700 1 |
611 0701 1 | ROUTINE VALUE:
612 0702 1 | COMPLETION CODES:
613 0703 1 |
614 0704 1 |     $$$ NORMAL      - Successful completion
615 0705 1 |     LIB$_INVARG     - Invalid argument. Either "code" or "handle"
616 0706 1 |                     is invalid.
617 0707 1 |
618 0708 1 | SIDE EFFECTS:
619 0709 1 |
620 0710 1 |     NONE
621 0711 1 |
622 0712 1 | --
623 0713 1 |
624 0714 2 | BEGIN
625 0715 2 |
626 0716 2 | BUILTIN
627 0717 2 |     SUBM,
628 0718 2 |     NULLPARAMETER;
629 0719 2 |
630 0720 2 | LOCAL
631 0721 2 |     STATUS,                ! Returned status condition
632 0722 2 |     LSTORAGE,              ! Will contain address of storage block
633 0723 2 |     TSTORAGE : VECTOR [N_OF_VALUES + 1], ! Gets current times/counts
634 0724 2 |     CPU_HOURS;             ! Elapsed CPU time
635 0725 2 |
636 0726 2 | MAP
637 0727 2 |     LSTORAGE : REF VECTOR [N_OF_VALUES + 2];      ! Local name for storage
638 0728 2 |
639 0729 2 | BIND
640 0730 2 |     PAGEFAULTS = TSTORAGE [0],
641 0731 2 |     CPUTIMX = TSTORAGE [1],
642 0732 2 |     BUFIO = TSTORAGE [2],
643 0733 2 |     DIRIO = TSTORAGE [3],
644 0734 2 |     ELAPSED_TIME = TSTORAGE [4];
645 0735 2 |
646 0736 3 | IF (NULLPARAMETER (3))      ! If "handle" omitted
647 0737 2 | THEN
648 0738 2 |     LSTORAGE = STORAGE      ! Use OWN storage
649 0739 2 | ELSE
650 0740 2 |     LSTORAGE = ..HANDLE;    ! Use handle
651 0741 2 |
652 0742 2 | IF (.LSTORAGE [0] NEQU %ASCII'TIMR') THEN RETURN (LIB$_INVARG);    ! Not a valid block
653 0743 2 |
654 0744 2 | STATUS = LIB$$GETJPI (TSTORAGE);    ! Get values
655 0745 2 |
656 0746 2 | IF ( NOT .STATUS) THEN RETURN (.STATUS);
657 0747 2 |
658 0748 2 | !+
659 0749 2 |     Compute all values
660 0750 2 | !-
661 0751 2 |     PAGEFAULTS = .PAGEFAULTS - .LSTORAGE [1];
662 0752 2 |     CPU_HOURS = .CPUTIMX - .LSTORAGE [2];
663 0753 2 |     BUFIO = .BUFIO - .LSTORAGE [3];
664 0754 2 |     DIRIO = .DIRIO - .LSTORAGE [4];
665 0755 2 |     SUBM (2, LSTORAGE [5], ELAPSED_TIME, ELAPSED_TIME);
```

```

: 666      0756 2 1+
: 667      0757 2 1+
: 668      0758 2 1+
: 669      0759 2 1+
: 670      0760 2 1+
: 671      0761 2 1+
: 672      0762 2 1+
: 673      0763 2 1+
: 674      0764 2 1+
: 675      0765 2 1+
: 676      0766 2 1+
: 677      0767 2 1+
: 678      0768 2 1+
: 679      0769 2 1+
: 680      0770 2 1+
: 681      0771 2 1+
: 682      0772 2 1+
: 683      0773 2 1+
: 684      0774 2 1+
: 685      0775 2 1+
: 686      0776 2 1+
: 687      0777 2 1+
: 688      0778 2 1+
: 689      0779 2 1+
: 690      0780 2 1+
: 691      0781 2 1+
: 692      0782 2 1+
: 693      0783 2 1+
: 694      0784 2 1+
: 695      0785 2 1+
: 696      0786 1 1+

Now return appropriate statistic determined by CODE

CASE CODE FROM 1 TO N_OF_VALUES OF
SET
[1] :                               ! Elapsed time
      BEGIN
      .VALUE = .ELAPSED_TIME;       ! quadword format
      (.VALUE) + 4 = .(ELAPSED_TIME + 4);
      END;
[2] :                               ! CPU time
      .VALUE = .CPU_HOURS;
[3] :                               ! Buffered I/O count
      .VALUE = .BUFIO;
[4] :                               ! Direct I/O count
      .VALUE = .DIRIO;
[5] :                               ! Page fault count
      .VALUE = .PAGEFAULTS;
[OUTRANGE] :
      RETURN (LIB$_INVARG);
TES;
RETURN (SS$_NORMAL);
END;                               ! Exit
                                   ! End of routine LIB$STAT_TIMER
```

```

                                0004 00000
                                18 C2 00002
                                6C 91 00005
                                05 1F 00008
                                OC AC D5 0000A
                                09 12 0000D
                                52 00000000' EF 9E 0000F 1$:
                                04 11 00016
                                52 OC BC D0 00018 2$:
524D4954 8F 62 D1 0001C 3$:
                                37 12 00023
                                5E DD 00025
                                01 FB 00027
                                50 E9 0002C
                                04 A2 C2 0002F
51 04 AE 08 A2 C3 00033
08 AE 0C A2 C2 00039
OC AE 10 A2 C2 0003E
10 AE 14 A2 C2 00043
14 AE 18 A2 D9 00048
04 01 04 BC CF 0004D

.ENTRY LIB$STAT_TIMER, Save R2
: 0642
SUBL2 #24, SP
: 0736
CMPB (AP), #3
BLSSU 1$
TSTL 12(AP)
BNEQ 2$
MOVAB STORAGE, LSTORAGE
BRB 3$
: 0738
MOVL @HANDLE, LSTORAGE
: 0740
CMPL (LSTORAGE), #1380796756
: 0742
BNEQ 5$
PUSHL SP
: 0744
CALLS #1, LIB$$GETJPI
BLBC STATUS, 12$
: 0746
SUBL2 4(LSTORAGE), PAGEFAULTS
: 0751
SUBL3 8(LSTORAGE), CPUTIMX, CPU_HOURS
: 0752
SUBL2 12(LSTORAGE), BUFIO
: 0753
SUBL2 16(LSTORAGE), DIRIO
: 0754
SUBL2 20(LSTORAGE), ELAPSED_TIME
: 0755
SBWC 24(LSTORAGE), ELAPSED_TIME
CASEL @CODE, #1, #4
: 0760
```

LIB\$TIMER
1-011

G 10
16-Sep-1984 01:18:20
14-Sep-1984 12:39:33

VAX-11 Bliss-32 V4.0-742
[LIBRTL.SRC]LIBTIMER.B32;1

Page 20
(5)

0029

0022

001C

0012
0030

00052 4\$:
0005A

.WORD

6\$-4\$,-
7\$-4\$,-
8\$-4\$,-
9\$-4\$,-
10\$-4\$

50 00000000G

8F

D0 0005C 5\$:

MOVL

#LIB\$_INVARG, R0

0782

50

08

AC

D0 00063

RET

60

10

AE

D0 00064 6\$:

MOVL

VALUE, R0

0765

08 BC

18

7D 00068

MOVQ

ELAPSED_TIME, (R0)

0760

08 BC

08

AE

D0 0006C 7\$:

BRB

CPU_HOURS, @VALUE

0770

08 BC

08

AE

D0 00072 8\$:

BRB

BUFIO, @VALUE

0773

08 BC

0C

AE

D0 00074 9\$:

BRB

DIRIO, @VALUE

0776

08 BC

04

D0 00078 10\$:

BRB

PAGEFAULTS, @VALUE

0779

50

6E

D0 00080 11\$:

MOVL

#1, R0

0785

04 00086 12\$:

RET

0786

; Routine Size: 138 bytes, Routine Base: _LIB\$CODE + 02D9

; 697 0787 1

```
699 0788 1 GLOBAL ROUTINE LIB$FREE_TIMER (
700 0789 1     HANDLE
701 0790 1 ) =
702 0791 1
703 0792 1 ++
704 0793 1 FUNCTIONAL DESCRIPTION:
705 0794 1
706 0795 1     Frees the storage allocated by LIB$INIT_TIMER. If the block
707 0796 1     referred to by "handle" was not allocated there, an error is
708 0797 1     returned.
709 0798 1
710 0799 1 CALLING SEQUENCE:
711 0800 1
712 0801 1     status.wlc.v = LIB$FREE_TIMER (handle.ml)
713 0802 1
714 0803 1
715 0804 1 FORMAL PARAMETERS:
716 0805 1
717 0806 1     handle - This must be the same value as returned by a previous
718 0807 1             call to LIB$INIT_TIMER. On a successful return, "handle"
719 0808 1             is set to zero.
720 0809 1
721 0810 1 IMPLICIT INPUTS:
722 0811 1
723 0812 1     It is assumed that "handle" is the same value returned by a
724 0813 1     previous call to LIB$INIT_TIMER.
725 0814 1
726 0815 1 IMPLICIT OUTPUTS:
727 0816 1
728 0817 1     NONE
729 0818 1
730 0819 1 ROUTINE VALUE:
731 0820 1 COMPLETION CODES:
732 0821 1
733 0822 1     SSS NORMAL      - Successful completion
734 0823 1     LIB$_INVARG     - Invalid argument. "handle" is invalid.
735 0824 1     LIB$_BADBLOADR  - "handle" is invalid.
736 0825 1
737 0826 1 SIDE EFFECTS:
738 0827 1
739 0828 1     The storage allocated by LIB$INIT_TIMER is freed.
740 0829 1
741 0830 1 --
742 0831 1
743 0832 2 BEGIN
744 0833 2
745 0834 2 BUILTIN
746 0835 2     NULLPARAMETER;
747 0836 2
748 0837 2 LOCAL
749 0838 2     STATUS;
750 0839 2
751 0840 2 ++
752 0841 2     Test to make sure block was
753 0842 2     initialized by LIB$INIT_TIMER.
754 0843 2 --
755 0844 2
```

```
: 756      0845 3      IF ((NULLPARAMETER (1)) OR (...HANDLE NEQU %ASCII'TIMR'))
: 757      0846      THEN
: 758      0847          RETURN (LIB$_INVARG);
: 759      0848
: 760      0849      +
: 761      0850          Now, free the storage
: 762      0851      -
: 763      0852          STATUS = LIB$FREE_VM (%REF (BLOCKSIZE), .HANDLE);
: 764      0853
: 765      0854          IF ( NOT .STATUS) THEN RETURN (.STATUS);
: 766      0855
: 767      0856          .HANDLE = 0;
: 768      0857          RETURN (.STATUS);
: 769      0858 1      END;
```

```
! Reset handle
! Must be $$ NORMAL
! End of routine LIB$FREE_TIMER
```

			0000 00000	.ENTRY	LIB\$FREE_TIMER, Save nothing	: 0788
	5E		04 C2 00002	SUBL2	#4, SP	
			6C 95 00005	TSTB	(AP)	: 0845
			12 13 00007	BEQL	1\$	
		04	AC D5 00009	TSTL	4(AP)	
			0D 13 0000C	BEQL	1\$	
	50	04	BC D0 0000E	MOVL	@HANDLE, R0	
524D4954	8F		60 D1 00012	CMPL	(R0), #1380796756	
			08 13 00019	BEQL	2\$	
	50	00000000G	8F D0 0001B 1\$:	MOVL	#LIB\$_INVARG, R0	: 0847
			04 00022	RET		
		04	AC DD 00023 2\$:	PUSHL	HANDLE	: 0852
	04	AE	1C D0 00026	MOVL	#28, 4(SP)	
		04	AE 9F 0002A	PUSHAB	4(SP)	
00000000G	00		02 FB 0002D	CALLS	#2, LIB\$FREE_VM	
	03		50 E9 00034	BLBC	STATUS, 3\$	: 0854
		04	BC D4 00037	CLRL	@HANDLE	: 0856
			04 0003A 3\$:	RET		: 0858

; Routine Size: 59 bytes, Routine Base: _LIB\$CODE + 0363

; 770 0859 1

```

: 772 0860 1 ROUTINE LIB$$GETJPI (
: 773 0861 1     VALUES
: 774 0862 1     ) =
: 775 0863 1
: 776 0864 1 ++
: 777 0865 1 FUNCTIONAL DESCRIPTION:
: 778 0866 1
: 779 0867 1     Calls the system services SYS$GETJPI and SYS$GETTIM and places the times
: 780 0868 1     and counts in the block pointed to by VALUES. The particular
: 781 0869 1     times and counts obtained are determined by the codes placed
: 782 0870 1     in JPIPARAMS, a block of OWN storage. This block is never
: 783 0871 1     altered; instead it is copied to the stack and changed there,
: 784 0872 1     thus this routine is reentrant.
: 785 0873 1
: 786 0874 1 FORMAL PARAMETERS:
: 787 0875 1
: 788 0876 1     VALUES.ml.ra - Points to a block of N_OF_VALUES+1 longwords
: 789 0877 1     into which are placed the times and counts.
: 790 0878 1     The current time of day is placed in the
: 791 0879 1     last two longwords of the block.
: 792 0880 1
: 793 0881 1 IMPLICIT INPUTS:
: 794 0882 1
: 795 0883 1     NONE
: 796 0884 1
: 797 0885 1 IMPLICIT OUTPUTS:
: 798 0886 1
: 799 0887 1     NONE
: 800 0888 1
: 801 0889 1 ROUTINE VALUE:
: 802 0890 1 COMPLETION CODES:
: 803 0891 1
: 804 0892 1     SSS_NORMAL - Successful completion
: 805 0893 1
: 806 0894 1     Any other completion code is an error returned by SYS$GETJPI or SYS$GETTIM.
: 807 0895 1
: 808 0896 1 SIDE EFFECTS:
: 809 0897 1
: 810 0898 1     NONE
: 811 0899 1
: 812 0900 1 --
: 813 0901 1
: 814 0902 2 BEGIN
: 815 0903 2
: 816 0904 2 MAP
: 817 0905 2     VALUES : REF VECTOR [N_OF_VALUES + 1];
: 818 0906 2
: 819 0907 2 LOCAL
: 820 0908 2     STATUS, ! Returned status from services
: 821 0909 2     EFN, ! Event flag number
: 822 0910 2     JPILIST : VECTOR [LISTSIZE]; ! Temporary GETJPI argument list
: 823 0911 2
: 824 0912 2     CH$MOVE (LISTSIZE*%UPVAL, JPIPARAMS, JPILIST); ! Move argument list to temporary
: 825 0913 2
: 826 0914 2 ++
: 827 0915 2 ! Place in the temporary argument list the addresses of the storage
: 828 0916 2 ! block locations which are to receive the times and counts.
```

```
0917 2 !-
0918
0919 INCR I FROM 0 TO N_OF_VALUES - 2 DO
0920 BEGIN
0921 JPILIST [((.I)*3) + 1] = VALUES [.I];
0922 END;
0923
0924 !+
0925 Allocate an event flag number
0926 !-
0927
0928 STATUS = LIB$GET_EF (EFN);
0929 IF (NOT .STATUS) THEN RETURN .STATUS;
0930
0931 STATUS = $GETJPIW (EFN=.EFN, ITMLST=JPILIST); ! Get times and counts
0932 LIB$FREE_EF (EFN);
0933 IF (NOT .STATUS) THEN RETURN (.STATUS);
0934
0935 STATUS = LIB$FREE_EF (EFN);
0936 RETURN ($GETTIM (TIMADR=VALUES [N_OF_VALUES - 1])); ! Get elapsed time
0937 END; ! End of routine LIB$$GETJPI
```

.EXTRN SYSS\$GETJPIW, SYSS\$GETTIM

```
007C 00000 LIB$$GETJPI:
56 00000000G 00 9E 00002 .WORD Save R2,R3,R4,R5,R6 0860
5E 38 C2 00009 MOVAB LIB$FREE_EF, R6
04 AE FC51 CF 34 28 0000C SUBL2 #56, SP 0912
51 50 D4 00013 MOV C3 #52, JPIPARAMS, JPILIST 0921
F1 08 AE41 04 BC40 DE 00019 1$: MULL3 #3, I, R1
50 03 C5 00015 MOVAL @VALUES[I], JPILIST+4[R1]
00000000G 00 01 FB 00026 AOBLEQ #3, I, 1$ 0919
52 50 D0 0002D PUSH SP 0928
1E 52 E9 00030 CALLS #1, LIB$GET_EF
7E 7C 00033 MOV R0, STATUS 0929
7E D4 00035 BLBC STATUS, 2$ 0931
10 AE 9F 00037 CLRQ -(SP)
7E 7C 0003A CLRQ -(SP)
18 AE DD 0003C PUSHAB JPILIST
00000000G 00 07 FB 0003F CALLS #7, SYSS$GETJPIW
52 50 D0 00046 MOV R0, STATUS 0932
66 01 FB 0004B PUSH SP
04 52 E8 0004E CALLS #1, LIB$FREE_EF 0933
50 52 D0 00051 MOV STATUS, R0
5E DD 00055 2$: RET 0935
01 FB 00057 3$: PUSH SP
50 D0 0005A CALLS #1, LIB$FREE_EF
7E 04 AC 10 C1 0005D MOV R0, STATUS 0936
00000000G 00 01 FB 00062 ADDL3 #16, VALUES, -(SP)
04 00069 CALLS #1, SYSS$GETTIM 0937
RET
```

; Routine Size: 106 bytes, Routine Base: _LIB\$CODE + 039E

; 850 0938 1 END !End of module
; 851 0939 1
; 852 0940 0 ELUDOM

PSECT SUMMARY

Name	Bytes	Attributes
_LIB\$DATA	32	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, PIC, ALIGN(2)
_LIB\$CODE	1032	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]STARLET.L 32;1	9776	15	0	581	00:00.7

COMMAND QUALIFIERS

; BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LISS:LIB\$TIMER/OBJ=OBJ\$:LIB\$TIMER MSRC\$:LIB\$TIMER/UPDATE=(ENHS:LIB\$TIMER)

; Size: 784 code + 280 data bytes
; Run Time: 00:12.2
; Elapsed Time: 00:55.8
; Lines/CPU Min: 4630
; Lexemes/CPU-Min: 24428
; Memory Used: 145 pages
; Compilation Complete

0210 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

