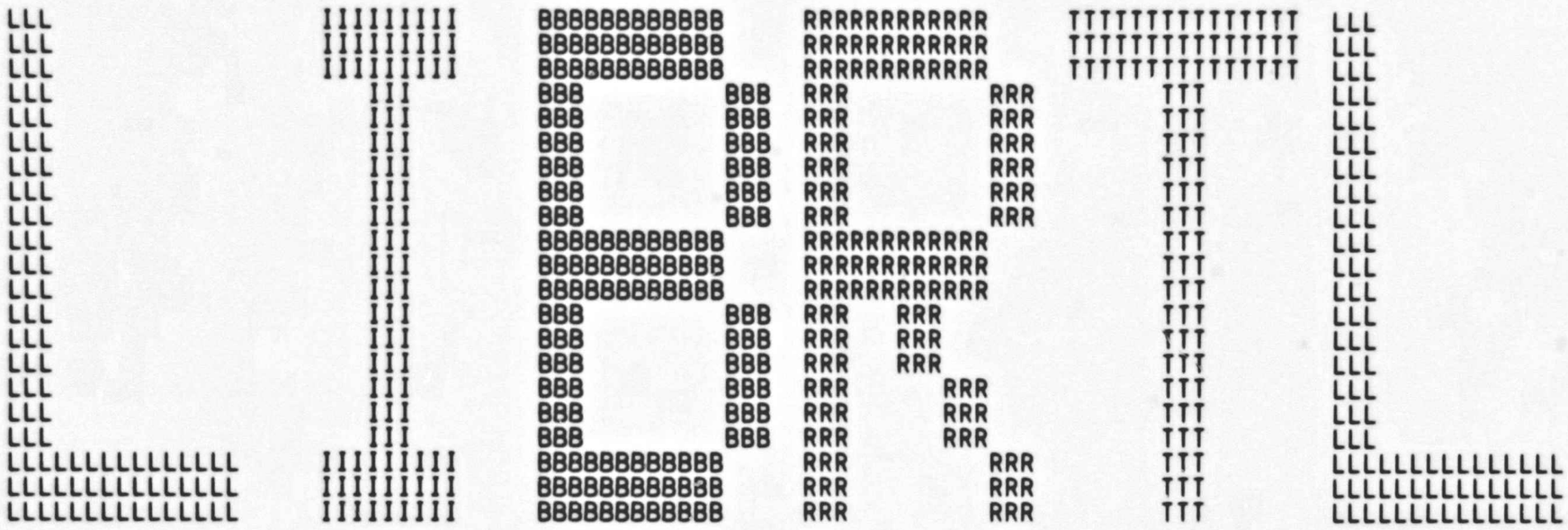
LLL IIIIIIIII BBBBBBBBBBB RRRRRRRRRRR TTTTTTTTTTTTTTT LLL
LLL IIIIIIIII BBBBBBBBBBBB RRRRRRRRRRRR TTTTTTTTTTTTTTT LLL
LLL IIIIIIIII BBBBBBBBBBBB RRRRRRRRRRR TTTTTTTTTTTTTTT LLL
LLL III BBB BBB RRR RRR TTT LLL
LLL III BBB BBB RRR RRR TTT LLL
LLL III BBB BBB RRR RRR TTT LLL
LLL III BBB BBB RRR RRR TTT LLL
LLL III BBB BBB RRR RRR TTT LLL
LLL III BBB BBB RRR RRR TTT LLL
LLL III BBBBBBBBBBB RRRRRRRRRRR TTT LLL
LLL III BBBBBBBBBBBB RRRRRRRRRRRR TTT LLL
LLL III BBBBBBBBBBBB RRRRRRRRRRRR TTT LLL
LLL III BBB BBB RRR RRR TTT LLL
LLL III BBB BBB RRR RRR TTT LLL
LLL III BBB BBB RRR RRR TTT LLL
LLL III BBB BBB RRR RRR TTT LLL
LLL III BBB BBB RRR RRR TTT LLL
LLLLLLLLLLLLLLL IIIIIIIII BBBBBBBBBBBB RRR RRR TTT LLLLLLLLLLLLLLL
LLLLLLLLLLLLLLL IIIIIIIII BBBBBBBBBBBB RRR RRR TTT LLLLLLLLLLLLLLL
LLLLLLLLLLLLLLL IIIIIIIII BBBBBBBBBBBB RRR RRR TTT LLLLLLLLLLLLLLL

```
LL             IIIIII    BBBBBBBB      SSSSSSSS    IIIIII      GGGGGGGG  NN      NN    AAAAAA     LL
LL             IIIIII    BBBBBBBB      SSSSSSSS    IIIIII      GGGGGGGG  NN      NN    AAAAAA     LL
LL               II      BB      BB    SS            II      GG          NN      NN   AA    AA    LL
LL               II      BB      BB    SS            II      GG          NN      NN   AA    AA    LL
LL               II      BB      BB    SS            II      GG          NNNN    NN   AA    AA    LL
LL               II      BB      BB    SS            II      GG          NNNN    NN   AA    AA    LL
LL               II      BBBBBBBB        SSSSSS      II      GG          NN  NN  NN   AA    AA    LL
LL               II      BBBBBBBB        SSSSSS      II      GG          NN  NN  NN   AA    AA    LL
LL               II      BB      BB            SS    II      GG  GGGGGG  NN    NNNN   AAAAAAAAAA  LL
LL               II      BB      BB            SS    II      GG  GGGGGG  NN    NNNN   AAAAAAAAAA  LL
LL               II      BB      BB            SS    II      GG      GG  NN      NN   AA      AA  LL                  ....
LL               II      BB      BB            SS    II      GG      GG  NN      NN   AA      AA  LL                  ....
LLLLLLLLLL     IIIIII    BBBBBBBB      SSSSSSSS    IIIIII      GGGGGG    NN      NN   AA      AA  LLLLLLLLLL          ....
LLLLLLLLLL     IIIIII    BBBBBBBB      SSSSSSSS    IIIIII      GGGGGG    NN      NN   AA      AA  LLLLLLLLLL          ....

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
LLLLLLLLLL     IIIIII    SSSSSSSS
LLLLLLLLLL     IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          .TITLE  LIB$SIGNAL - Condition Handling Facility SIGNAL and STOP
0000     2          .IDENT  /1-018/                 ; File: LIBSIGNAL.MAR  Edit: ACG0183
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28
0000    29 ;++
0000    30 ; FACILITY:     Condition Handling
0000    31 ;
0000    32 ; ABSTRACT:
0000    33 ;
0000    34 ;       The Condition Handling Facility supports the exception
0000    35 ;       handling mechanisms needed by each of the common languages.
0000    36 ;       It provides the programmer with some control over fixup,
0000    37 ;       reporting, and flow of control on errors.  It provides
0000    38 ;       subsystem and application writers with the ability to
0000    39 ;       override system messages in order to give a more suitable
0000    40 ;       application oriented interface.
0000    41 ;
0000    42 ;       The CHF includes procedures to allow higher level language
0000    43 ;       users to change the hardware enables, and to establish and
0000    44 ;       revert condition handlers. This module includes procedures
0000    45 ;       to signal exceptions (LIB$SIGNAL and LIB$STOP).  The
0000    46 ;       facility also includes a procedure to unwind the stack
0000    47 ;       from a handler to its establisher (SYS$SET_UNWIND).
0000    48 ;
0000    49 ;       To understand CHF more fully, refer to its functional
0000    50 ;       specification and to the STARLET exception routine (EXCEPTION).
0000    51 ;
0000    52 ;
0000    53 ; ENVIRONMENT:  Any access mode--normally user mode
0000    54 ;                AST reentrant
0000    55 ;
0000    56 ; AUTHOR: Peter F. Conklin, CREATION DATE: 12-Nov-76
```

```
0000     58          .SBTTL  Modification History
0000     59 ;
0000     60 ; MODIFIED BY:
0000     61 ;
0000     62 ;  Peter F. Conklin, 27-Jan-78 VERSION 01
0000     63 ; 01     -  Original, based on CHF Rev 4 spec.
0000     64 ; 02     -  Changed to Rev 5 spec.
0000     65 ; 03     -  Exit with "no handler" or "access violation" if bad stack.
0000     66 ; 04     -  Copy signal args; add PSL and PC.
0000     67 ; 05     -  Reformat stack to mimic EXCEPTION exactly.
0000     68 ; 06     -  Change to PSECT LIB$CODE
0000     69 ; 07     -  Use $GETJPI, SYS$EXCMSG, SS$_NOHANDLER
0000     70 ;               support last chance handler,
0000     71 ;               STOP forces severe at every handler call.
0000     72 ; 08     - Change name to SYS$EXCMSG.  TNH 24-Jan-78
0000     73 ; 09     - Remember EXCVEC address.
0000     74 ; 10     - Continue from STOP now vectors to a Panic EXIT.
0000     75 ; 11     - Change name of OWN PSECT to LIB$DATA.
0000     76 ; 12     - Use G^ addressing in call to SYS$UNWIND.  JMT 28-Feb-78
0000     77 ; 13     - Make OWN be PIC.  TNH 27-June-78
0000     78 ; 1-014 - Reformatted version number to have three digits in the
0000     79 ;          edit number field.  JBS 16-NOV-78
0000     80 ; 1-015 - Add "_" to the PSECT directives.  JBS 21-DEC-78
0000     81 ; 1-016 - Make sure that SP stays below needed information.
0000     82 ;          SPR 11-24926  SBL 13-July-1979
0000     83 ; 1-017 - Remove restriction that the stack frame must be in P1 space.
0000     84 ;          SBL 22-May-1980
0000     85 ; 1-018 - ACG0183: Andrew C. Goldstein  30-Dec-1980  15:44
0000     86 ;          Complete rewrite, to correct stack management bugs in
0000     87 ;          building of signal vectors, and to use SYS$SRCHANDLER entry
0000     88 ;          to the EXEC's handler search logic, eliminating duplicated
0000     89 ;          condition handler search algorithms.
0000     90 ;--
```

```
                        92          .SBTTL  DECLARATIONS
                        93
                        94 ;
                        95 ; EXTERNAL REFERENCES:
                        96 ;
                        97          .DSABL  GBL
                        98
                        99          .EXTRN  SYS$SRCHANDLER
                       100
                       101 ;
                       102 ; INCLUDE FILES:
                       103 ;
                       104          $LIBDEF                                ;Library status defs
                       105          $SFDEF                                 ;Stack frame offsets
                       106
                       107 ;
                       108 ; MACROS:
                       109 ;
                       110
                       111 ;
                       112 ; This macro defines the formals to a procedure
                       113 ;
                       114
                       115          .MACRO  $FORMAL LIST
                       116 $$FORMAL=0
                       117          .IRP    L,<LIST>
                       118 $$FORMAL=$$FORMAL+4
                       119 L=$$FORMAL
                       120          .ENDM
                       121          .ENDM   $FORMAL
                       122
                       123 ;
                       124 ; EQUATED SYMBOLS:
                       125 ;
                       126 ;          NONE
                       127 ;
                       128 ; OWN STORAGE:
                       129 ;
                       130 ;          NONE
                       131 ;
                       132
                       133          .PSECT  _LIB$CODE,PIC,SHR,NOWRT,LONG
```

```
                     0000   135          .SBTTL  LIB$STOP -  Stop execution via signalling
                     0000   136 ;++
                     0000   137 ; FUNCTIONAL DESCRIPTION:
                     0000   138 ;
                     0000   139 ;       This procedure is called whenever it is impossible
                     0000   140 ;       to continue execution and no recovery is possible.
                     0000   141 ;       It signals the exception. It always forces the
                     0000   142 ;       severity code of severe_error on each call to
                     0000   143 ;       a handler. Handler requests to continue are
                     0000   144 ;       treated as an error and produce a panic exit.
                     0000   145 ;       This procedure is guaranteed to never return.
                     0000   146 ;
                     0000   147 ;
                     0000   148 ; CALLING SEQUENCE:
                     0000   149 ;
                     0000   150 ;       CALL LIB$STOP (
                     0000   151 ;
                     0000   152 ;       CONDITON_VALUE.rlc.v,   standard signal name
                     0000   153 ;
                     0000   154 ;       {ARGS.rl.v})            additional FAO parameters for message
                     0000   155 ;                               (stop adds PC and PSL to end)
                     0000   156 ;
                     0000   157 ; INPUT PARAMETERS:
                     0000   158 ;
                     0000   159 ;       NONE
                     0000   160 ;
                     0000   161 ; IMPLICIT INPUTS:
                     0000   162 ;
                     0000   163 ;       NONE
                     0000   164 ;
                     0000   165 ; OUTPUT PARAMETERS:
                     0000   166 ;
                     0000   167 ;       NONE
                     0000   168 ;
                     0000   169 ; IMPLICIT OUTPUTS:
                     0000   170 ;
                     0000   171 ;       NONE
                     0000   172 ;
                     0000   173 ; COMPLETION CODES:
                     0000   174 ;
                     0000   175 ;       NONE
                     0000   176 ;
                     0000   177 ; SIDE EFFECTS:
                     0000   178 ;
                     0000   179 ;       Never returns
                     0000   180 ;
                     0000   181 ;--
                     0000   182
                     0000   183          $FORMAL <-
                     0000   184 CONDITION_VALUE-                 ;signal code
                     0000   185          >                      ; {parameters}
                     0000   186
                0000 0000   187          .ENTRY  LIB$STOP,0              ;No registers (assumed below)
             02   DD 0002   188          PUSHL   #2                      ;Set code for STOP
             0E   11 0004   189          BRB     SIGNAL                  ;Go do the signallling
```

```
                     0006   191          .SBTTL  LIB$SIGNAL -  Signal Exceptional Condition
                     0006   192 ;++
                     0006   193 ; FUNCTIONAL DESCRIPTION:
                     0006   194 ;
                     0006   195 ;       This procedure is called whenever it is necessary
                     0006   196 ;       to indicate an exceptional condition and the procedure
                     0006   197 ;       can not return a status code. If a handler returns
                     0006   198 ;       with a continue code, LIB$SIGNAL returns with
                     0006   199 ;       all registers including R0 and R1 preserved.  Thus,
                     0006   200 ;       LIB$SIGNAL can also be used to plant performance and
                     0006   201 ;       debugging traps in any code.  If no handler is found,
                     0006   202 ;       or all resignal, a catch-all handler is CALLed.
                     0006   203 ;
                     0006   204 ; CALLING SEQUENCE:
                     0006   205 ;
                     0006   206 ;       CALL LIB$SIGNAL (
                     0006   207 ;
                     0006   208 ;       CONDITION_VALUE.rlc.v,  standard signal name
                     0006   209 ;
                     0006   210 ;       {ARGS.rl.v})            additional FAO parameters for message
                     0006   211 ;                               (signal adds PC and PSL to end)
                     0006   212 ;
                     0006   213 ; INPUT PARAMETERS:
                     0006   214 ;
                     0006   215 ;       NONE
                     0006   216 ;
                     0006   217 ; IMPLICIT INPUTS:
                     0006   218 ;
                     0006   219 ;       NONE
                     0006   220 ;
                     0006   221 ; OUTPUT PARAMETERS:
                     0006   222 ;
                     0006   223 ;       NONE
                     0006   224 ;
                     0006   225 ; IMPLICIT OUTPUTS:
                     0006   226 ;
                     0006   227 ;       NONE
                     0006   228 ;
                     0006   229 ; COMPLETION CODES:
                     0006   230 ;
                     0006   231 ;       NONE
                     0006   232 ;
                     0006   233 ; SIDE EFFECTS:
                     0006   234 ;
                     0006   235 ;       If a handler unwinds, then control will not return.
                     0006   236 ;       A handler could also modify R0/R1 and change the
                     0006   237 ;       flow of control.  If neither is done, then all
                     0006   238 ;       registers are preserved.
                     0006   239 ;
                     0006   240 ;--
                     0006   241
                0000 0006   242          .ENTRY  LIB$SIGNAL,0            ;No registers (assumed below)
             01   DD 0008   243          PUSHL   #1                      ;Set code for SIGNAL
             08   11 000A   244          BRB     SIGNAL                  ;go do the signalling
```

```
                        000C   246          .SBTTL  SIGNAL -  Internal Routine to Signal Exceptions
                        000C   247 ;++
                        000C   248 ; FUNCTIONAL DESCRIPTION:
                        000C   249 ;
                        000C   250 ;       This routine is used by LIB$STOP and LIB$SIGNAL to do
                        000C   251 ;       the actual exception signaling. It converts the call frame
                        000C   252 ;       and argument list into mechanism and signal vectors.
                        000C   253 ;       It then jumps to the EXEC's common condition handler
                        000C   254 ;       search routine.
                        000C   255 ;
                        000C   256 ;
                        000C   257 ; CALLING SEQUENCE:
                        000C   258 ;
                        000C   259 ;       PUSHL   #code           ;1=SIGNAL, 2=STOP
                        000C   260 ;       BR      SIGNAL
                        000C   261 ;
                        000C   262 ; INPUT PARAMETERS:
                        000C   263 ;
                        000C   264 ;       AP points to the arg list
                        000C   265 ;
                        000C   266 ; IMPLICIT INPUTS:
                        000C   267 ;
                        000C   268 ;       NONE
                        000C   269 ;
                        000C   270 ; OUTPUT PARAMETERS:
                        000C   271 ;
                        000C   272 ;       NONE
                        000C   273 ;
                        000C   274 ; IMPLICIT OUTPUTS:
                        000C   275 ;
                        000C   276 ;       NONE
                        000C   277 ;
                        000C   278 ; COMPLETION CODES:
                        000C   279 ;
                        000C   280 ;       NONE
                        000C   281 ;
                        000C   282 ; SIDE EFFECTS:
                        000C   283 ;
                        000C   284 ;       If a handler unwinds, then control will not return.
                        000C   285 ;       A handler could also modify R0/R1 and change the flow
                        000C   286 ;       of control. If neither is done, then all registers
                        000C   287 ;       are preserved.
                        000C   288 ;
                        000C   289 ;--
                        000C   290
                        000C   291 ;
                        000C   292 ; The following is a dummy signal argument list that is
                        000C   293 ; used if the call has no arguments. This dummy list
                        000C   294 ; has just one argument, in particular the signal code.
                        000C   295 ;
                        000C   296
                        000C   297 DUMMY_SIG_ARG:
0015825C 00000001       000C   298          .LONG   1,LIB$_SIGNO_ARG        ;dummy arg list
```

```
0014   300 ;+
0014   301 ; The code for the signal routine is straightforward
0014   302 ; and consists of the following three parts:
0014   303 ;
0014   304 ;       evaporate the call stack
0014   305 ;       format the handler argument list
0014   306 ;       jump to common handler search routine
0014   307 ;
0014   308 ; The only peculiar algorithm has to do with evaporating
0014   309 ; the call frame. This is needed so that the format of
0014   310 ; the stack is the same whether the signal was an
0014   311 ; explicit call to signal or was a hardware detected
0014   312 ; exception. The latter has no frame because the hardware
0014   313 ; pushes the PSL, the PC, and some arguments. The exception
0014   314 ; handler in the system adds a reason code to identify the
0014   315 ; exception and then goes into a duplicate of this code.
0014   316 ;
0014   317 ; The algorithm to evaporate the frame is different for
0014   318 ; the cases of being called by CALLG or CALLS. In the latter
0014   319 ; case, the arguments also must be evaporated (by incorporation
0014   320 ; into the signal vector). In the former case, the argument
0014   321 ; list must be copied in order to allow
0014   322 ; the handler to alter the severity and resignal. At the
0014   323 ; same time, the caller's PSL and PC are appended to
0014   324 ; the list for consistency with the hardware detected
0014   325 ; exceptions. In both cases, the only trick in the algorithm
0014   326 ; is to ensure that SP stays below any information on the
0014   327 ; stack, and to allow for various number of parameters
0014   328 ; including the 0 to 3 byte stack alignment on call.
0014   329 ;-
```

```
                                0014   331 SIGNAL:
                03      BB      0014   332         PUSHR   #^M<R0,R1>                   ;save user's R1'R0
3F 06 AD        0D      E1      0016   333         BBC     #SF$V_CALLS,SF$W_SAVE_MASK(FP),40$
                                001B   334                                             ;go handle CALLG call
                                001B   335
                                001B   336 ;
                                001B   337 ; Here on a CALLS with the stack:
                                001B   338 ;
                                001B   339 ;       (SP) = -12(FP) = caller's R0
                                001B   340 ;              -08(FP) = caller's R1
                                001B   341 ;              -04(FP) = code for LIB$STOP or LIB$SIGNAL
                                001B   342 ;               00(FP) = 0 (this incarnation's handler)
                                001B   343 ;               04(FP) = CALL frame mask and caller's PSW
                                001B   344 ;               08(FP) = caller's AP
                                001B   345 ;               12(FP) = caller's FP
                                001B   346 ;               16(FP) = caller's PC
                                001B   347 ;                 ...  = 0 to 3 bytes of alignment filler
                                001B   348 ;               00(AP) = number of arguments
                                001B   349 ;               04(AP)++ arguments to SIGNAL
                                001B   350 ;
                                001B   351
          08    AD      DD      001B   352         PUSHL   SF$L_SAVE_AP(FP)            ;save caller's AP
                7E      DC      001E   353         MOVPSL  -(SP)                       ;save PSL
       6E       04      AD      B0     0020   354         MOVW    SF$W_SAVE_PSW(FP),(SP)  ;update with caller's PSW
                10      AD      DD     0024   355         PUSHL   SF$L_SAVE_PC(FP)        ;save caller's PC
       5D       0C      AD      D0     0027   356         MOVL    SF$L_SAVE_FP(FP),FP     ;restore caller's FP
                                002B   357                                             ;the stack frame is now officially gone
          51    6C      9A      002B   358 5$:     MOVZBL  (AP),R1                     ;get number of args
                09      12      002E   359         BNEQU   10$                         ;branch if some
       6C     DD      AF      D0     0030   360         MOVL    DUMMY_SIG_ARG+4,(AP)    ;use dummy if none
             7C        01      D0     0034   361         MOVL    #1,-(AP)                ;make room and count as one arg
                       F2      11     0037   362         BRB     5$                      ;proceed
                                0039   363
          50    F8      AC      9E     0039   364 10$:    MOVAB   -8(AP),R0               ;point to top of signal vector
    6C    51            02      C1     003D   365         ADDL3   #2,R1,(AP)              ;add PC & PSL to argument count
      F4  AC            8C      D0     0041   366 20$:    MOVL    (AP)+,-12(AP)           ;copy args into signal vector
                F9      51      F4     0045   367         SOBGEQ  R1,20$
                                0048   368
             7C         8E      7D     0048   369         MOVQ    (SP)+,-(AP)             ;put away PSL'PC pair
             5C         8E      D0     004B   370         MOVL    (SP)+,AP                ;restore caller's AP
          70      08    AE      D0     004E   371         MOVL    8(SP),-(R0)             ;store code for SIGNAL/STOP
             70         6E      7D     0052   372         MOVQ    (SP),-(R0)              ;store caller's R1'R0
             5E         50      D0     0055   373         MOVL    R0,SP                   ;clean off unused space
                        4E      11     0058   374         BRB     80$                     ;go join common code
```

```
                                005A   376 ;
                                005A   377 ; Here on a CALLG
                                005A   378 ;
                                005A   379 ;
                                005A   380 ; At this point the stack is:
                                005A   381 ;
                                005A   382 ;       (SP) = -12(FP) = caller's R0
                                005A   383 ;              -08(FP) = caller's R1
                                005A   384 ;              -04(FP) = code for LIB$STOP or LIB$SIGNAL
                                005A   385 ;               00(FP) = 0 (this incarnation's handler)
                                005A   386 ;               04(FP) = CALL frame mask and caller's PSW
                                005A   387 ;               08(FP) = caller's AP
                                005A   388 ;               12(FP) = caller's FP
                                005A   389 ;               16(FP) = caller's PC
                                005A   390 ;                 ...    0 to 3 bytes of alignment filler
                                005A   391 ;
                                005A   392 ; NOTE: In the computation below, SP is set two long words lower
                                005A   393 ; then seems obviously needed. The two longword bias is necessary to
                                005A   394 ; guarantee that SP is not being set above the current location of
                                005A   395 ; the saved R0'R1 for the case of the smallest possible argument list.
                                005A   396 ;
                                005A   397
          50    5D    D0        005A   398 40$:    MOVL    FP,R0                   ;point to current frame
                5D    10        005D   399         BSBB    OLD_SP                  ;calculate pre-CALL SP into R0
          51    6C    9A        005F   400 50$:    MOVZBL  (AP),R1                 ;get arg count
                06    12        0062   401         BNEQU   60$                     ;proceed if not empty
       5C    A5   AF    9E     0064   402         MOVAB   DUMMY_SIG_ARG,AP        ;use default if empty
                F5    11        0068   403         BRB     50$                     ;and try again
                                006A   404
          51    51    CE        006A   405 60$:    MNEGL   R1,R1                   ;compute variable part of vector
       5E    E0 A041    DE     006D   406         MOVAL   -32(R0)[R1],SP          ;set SP to final value
          51    FC AD    D0     0072   407         MOVL    -4(FP),R1               ;save code for SIGNAL/STOP
    08 AE    F4 AD    7D     0076   408         MOVQ    -12(FP),8(SP)           ;put R1'R0 in correct place
       10 AE    51    D0     007B   409         MOVL    R1,16(SP)               ;store code in vector
    04 AE    08 AD    D0     007F   410         MOVL    SF$L_SAVE_AP(FP),4(SP)  ;save caller's AP for a moment
                6E    DC     0084   411         MOVPSL  (SP)                    ;save PSL for a moment
          6E    04 AD    B0     0086   412         MOVW    SF$W_SAVE_PSW(FP),(SP)  ;change PSL to caller's
                10 AD    DD     008A   413         PUSHL   SF$L_SAVE_PC(FP)        ;save caller's PC for a moment
          5D    0C AD    D0     008D   414         MOVL    SF$L_SAVE_FP(FP),FP     ;shift FP back to caller's FP
```

```
                        0091   416 ;
                        0091   417 ; At this point, the extra frame created by the CALL to us
                        0091   418 ; has been removed. We can now proceed to build the signal
                        0091   419 ; and mechanism vectors and the final arg list.
                        0091   420 ;
                        0091   421
      70    8E    7D    0091   422          MOVQ    (SP)+,-(R0)             ;move caller's PSL'PC to vector
      51    6C    9A    0094   423          MOVZBL  (AP),R1                 ;get arg count
   70    6C41    D0    0097   424 70$:      MOVL    (AP)[R1],-(R0)          ;copy args to signal vector
     F9  51    F5    009B   425          SOBGTR  R1,70$                  ; top down
      51    6C    9A    009E   426          MOVZBL  (AP),R1                 ;get arg count
70    51    02    C1    00A1   427          ADDL3   #2,R1,-(R0)             ;signal count is 2+arg count
      5C    8E    D0    00A5   428          MOVL    (SP)+,AP                ;change AP to caller's AP
                        00A8   429                                          ;CALLS entry joins here
      7E    03    CE    00A8   430 80$:      MNEGL   #3,-(SP)                ;initialize depth
            5D    DD    00AB   431          PUSHL   FP                      ;initialize frame
            04    DD    00AD   432          PUSHL   #4                      ;set mechanism vector length
            6E    9F    00AF   433          PUSHAB  (SP)                    ;2nd arg is mechanism vector
         1C AE    9F    00B1   434          PUSHAB  28(SP)                  ;1st arg is signal vector
            02    DD    00B4   435          PUSHL   #2                      ;two arguments
                        00B6   436
                        00B6   437 ;
                        00B6   438 ;  At this point the stack is all set for a call to any handler:
                        00B6   439 ;
                        00B6   440 ;       00(SP) = 2
                        00B6   441 ;       04(SP) = signal vector address          ( 36(SP) )
                        00B6   442 ;       08(SP) = mechanism vector address       ( 12(SP) )
                        00B6   443 ;       12(SP) = mechanism vector length (4)    \
                        00B6   444 ;       16(SP) = mechanism vector frame (FP)     |
                        00B6   445 ;       20(SP) = mechanism vector depth (-3)     | mechanism
                        00B6   446 ;       24(SP) = mechanism vector caller's R0    |
                        00B6   447 ;       28(SP) = mechanism vector caller's R1   /
                        00B6   448 ;       32(SP) = code for LIB$STOP or LIB$SIGNAL
                        00B6   449 ;       36(SP) = 2+number of caller's args      \
                        00B6   450 ;       40(SP)++ copy of caller's signal args   | signal
                        00B6   451 ;         ...    caller's return PC             |   args
                        00B6   452 ;         ...    caller's PSL                  /
                        00B6   453 ;
                        00B6   454 ;       The next higher location on the stack is the
                        00B6   455 ;       value of the caller's SP just before the CALL.
                        00B6   456 ;
                        00B6   457 ; AP and FP have been restored to the caller's values.
                        00B6   458 ;
                        00B6   459
00000000'GF   17        00B6   460          JMP     G^SYS$SRCHANDLER        ;go find a handler
```

```
                        00BC    462          .SBTTL  OLD_SP -  Internal Routine to Calculate Old SP
                        00BC    463 ;++
                        00BC    464 ; FUNCTIONAL DESCRIPTION:
                        00BC    465 ;
                        00BC    466 ;       This routine is called to calculate what SP was before
                        00BC    467 ;       a particular CALLG that resulted in a specific stack
                        00BC    468 ;       frame.  RESTRICTION: CALLS not handled.
                        00BC    469 ;
                        00BC    470 ; CALLING SEQUENCE:
                        00BC    471 ;
                        00BC    472 ;       NONE
                        00BC    473 ;
                        00BC    474 ; INPUT PARAMETERS:
                        00BC    475 ;
                        00BC    476 ;       R0 = address of stack frame in question
                        00BC    477 ;
                        00BC    478 ; IMPLICIT INPUTS:
                        00BC    479 ;
                        00BC    480 ;       NONE
                        00BC    481 ;
                        00BC    482 ; OUTPUT PARAMETERS:
                        00BC    483 ;
                        00BC    484 ;       R0 = value of SP before CALL in question
                        00BC    485 ;
                        00BC    486 ; IMPLICIT OUTPUTS:
                        00BC    487 ;
                        00BC    488 ;       NONE
                        00BC    489 ;
                        00BC    490 ; COMPLETION CODES:
                        00BC    491 ;
                        00BC    492 ;       NONE
                        00BC    493 ;
                        00BC    494 ; SIDE EFFECTS:
                        00BC    495 ;
                        00BC    496 ;       R1 is clobbered
                        00BC    497 ;
                        00BC    498 ;--
                        00BC    499
                        00BC    500
                        00BC    501
                        00BC    502 OLD_SP:
            02   0E     EF  00BC    503          EXTZV   #SF$V_STACKOFFS,#SF$S_STACKOFFS,-
       7E     06 A0         00BF    504                  SF$W_SAVE_MASK(R0),-(SP) ;get stack offset
51  06 A0    0C   00     EF  00C2    505          EXTZV   #SF$V_SAVE_MASK,#SF$S_SAVE_MASK,SF$W_SAVE_MASK(R0),R1
                           00C8    506                                                  ;get register mask
            50   14     C0  00C8    507          ADDL2   #SF$L_SAVE_REGS,R0           ;standard frame
            50   8E     C0  00CB    508          ADDL2   (SP)+,R0                     ;SP correction
              03 51     E9  00CE    509 10$:     BLBC    R1,20$                       ;if register bit set,
            50   04     C0  00D1    510          ADDL2   #4,R0                        ; count the register
      51    51   FF 8F   78  00D4    511 20$:     ASHL    #-1,R1,R1                    ;discard bit
                     F3   12  00D9    512          BNEQU   10$                          ;loop until all done
                          05  00DB    513          RSB                                  ;return
                              00DC    514
                              00DC    515
                              00DC    516
                              00DC    517          .END
```

```
$$FORMAL          =  00000004
DUMMY_SIG_ARG        0000000C R      02
LIB$SIGNAL           00000006 RG     02
LIB$STOP             00000000 RG     02
LIB$_SIGNO_ARG    =  0015825C
OLD_SP               000000BC R      02
SF$C_SAVE_AP      =  00000008
SF$L_SAVE_FP      =  0000000C
SF$L_SAVE_PC      =  00000010
SF$L_SAVE_REGS    =  00000014
SF$S_SAVE_MASK    =  0000000C
SF$S_STACKOFFS    =  00000002
SF$V_CALLS        =  0000000D
SF$V_SAVE_MASK    =  00000000
SF$V_STACKOFFS    =  0000000E
SF$W_SAVE_MASK    =  00000006
SF$W_SAVE_PSW     =  00000004
SIGNAL               00000014 R      02
SYS$SRCHANDLER       ********   X    00
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
_LIB$CODE	000000DC (220.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	30	00:00:00.02	00:00:01.15
Command processing	109	00:00:00.31	00:00:02.08
Pass 1	140	00:00:01.54	00:00:05.53
Symbol table sort	0	00:00:00.07	00:00:00.07
Pass 2	94	00:00:00.70	00:00:02.87
Symbol table output	3	00:00:00.02	00:00:00.02
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	380	00:00:02.69	00:00:11.74

```
The working set limit was 1050 pages.
12999 bytes (26 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 124 non-local and 10 local symbols.
517 source lines were read in Pass 1, producing 16 object records in Pass 2.
10 pages of virtual memory were used to define 9 macros.
```

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

Macro library name                    Macros defined
------------------                    --------------
_$255$DUA28:[SYSLIB]STARLET.MLB;2                5

268 GETS were required to define 5 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS$:LIBSIGNAL/OBJ=OBJ$:LIBSIGNAL MSRC$:LIBSIGNAL/UPDATE=(ENH$:LIBSIGNAL)
```

0209 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

LIBPOLYG
LIS

LIBREMQHI
LIS

LIBRENAME
LIS

LIBSCANC
LIS

LIBSIGSTO
LIS

LIBROOBJ
LIS

LIBRUNPRO
LIS

LIBSIGNAL
LIS

LIBPUTOUT
LIS

LIBREMQTI
LIS

LIBSIGRET
LIS

LIBSIMTRA
LIS

LIBSCOPY
LIS

LIBPOLYH
LIS

LIBREVERT
LIS