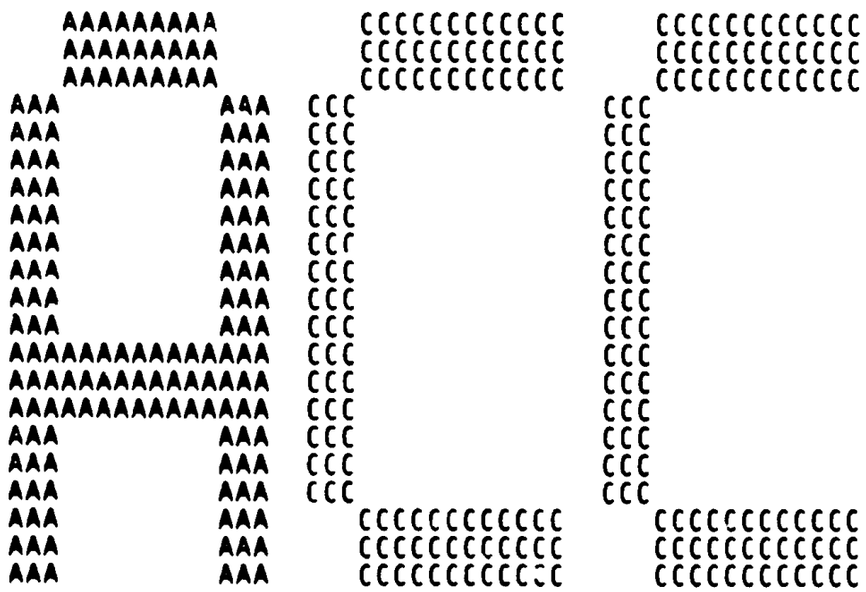
TITLE1**AH-Bi13A-BE
TITLE2**VAX/VMS V4.0
FICHENUMBER**1



FILEID**UTILITY

```
UU      UU  TTTTTTTTTT    IIIIII    LL          IIIIII    TTTTTTTTTT  YY      YY
UU      UU  TTTTTTTTTT    IIIIII    LL          IIIIII    TTTTTTTTTT  YY      YY
UU      UU      TT          II      LL            II          TT      YY      YY
UU      UU      TT          II      LL            II          TT      YY      YY
UU      UU      TT          II      LL            II          TT        YY  YY
UU      UU      TT          II      LL            II          TT        YY  YY
UU      UU      TT          II      LL            II          TT          YY
UU      UU      TT          II      LL            II          TT          YY
UU      UU      TT          II      LL            II          TT          YY
UU      UU      TT          II      LL            II          TT          YY
UU      UU      TT          II      LL            II          TT          YY          ....
UU      UU      TT          II      LL            II          TT          YY          ....
UUUUUUUUUU      TT      IIIIIII     LLLLLLLLLL    IIIIIII     TT          YY          ....
UUUUUUUUUU      TT      IIIIIII     LLLLLLLLLL    IIIIIII     TT          YY          ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
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
```

```
;     1          0001  0 MODULE
;     2          0002  0 UTILITY (IDENT = 'V04-000') =
;     3          0003  1 BEGIN
;     4          0004  1
;     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 ! FACILITY:  ACC, Account file dumper
;    31          0031  1 !
;    32          0032  1 ! ABSTRACT:
;    33          0033  1 !
;    34          0034  1 !       This module contains utility routines used by the ACC facility.
;    35          0035  1 !
;    36          0036  1 ! ENVIRONMENT:
;    37          0037  1 !
;    38          0038  1 !       VAX/VMS operating system. unprivileged user mode,
;    39          0039  1 !
;    40          0040  1 ! AUTHOR:  Greg Robert and Steve Forgey, January 1982
;    41          0041  1 !
;    42          0042  1 ! Modified by:
;    43          0043  1 !
;    44          0044  1 !       V03-003 MHB0159         Mark Bramhall           11-May-1984
;    45          0045  1 !               Remove special -2 and -4 status code checks.
;    46          0046  1 !
;    47          0047  1 !       V03-002 DAS0001         David Solomon           12-Jan-1984
;    48          0048  1 !               Get ACCDEF.REQ from SRC$, not MSRC$.
;    49          0049  1 !
;    50          0050  1 !       V03-001 SPF0079         Steve Forgey    05-Feb-1982
;    51          0051  1 !               Don't signal "end of file" errors.
;    52          0052  1 !
;    53          0053  1 !--
;    54          0054  1
;    55          0055  1 !----------------------------------------------------------------------------
;    56          0056  1 !                                                                           !
;    57          0057  1 !                                INCLUDE  FILES                             !
```

```
;    58          0058  1 !                                                                               !
;    59          0059  1 !-------------------------------------------------------------------------------!
;    60          0060  1
;    61          0061  1 SWITCHES LIST (REQUIRE);
;    62          0062  1
;    63          0063  1 REQUIRE 'SRC$:ACCDEF';          ! Command ACC definitions
```

```
;  R0064  1     !MODULE ACCDEF( IDENT = 'V04-000' ) =
;  R0065  1
;  R0066  1     !
;  R0067  1     !*****************************************************************************
;  R0068  1     !*                                                                           *
;  R0069  1     !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;  R0070  1     !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
;  R0071  1     !*  ALL RIGHTS RESERVED.                                                     *
;  R0072  1     !*                                                                           *
;  R0073  1     !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
;  R0074  1     !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
;  R0075  1     !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
;  R0076  1     !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
;  R0077  1     !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
;  R0078  1     !*  TRANSFERRED.                                                             *
;  R0079  1     !*                                                                           *
;  R0080  1     !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
;  R0081  1     !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
;  R0082  1     !*  CORPORATION.                                                             *
;  R0083  1     !*                                                                           *
;  R0084  1     !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
;  R0085  1     !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;  R0086  1     !*                                                                           *
;  R0087  1     !*                                                                           *
;  R0088  1     !*****************************************************************************
;  R0089  1
;  R0090  1     !++
;  R0091  1     ! FACILITY:  ACC, Account file dumper
;  R0092  1     !
;  R0093  1     ! ABSTRACT:
;  R0094  1     !
;  R0095  1     !       This file contains definitions of general applicability to
;  R0096  1     !       the accounting facility
;  R0097  1     !
;  R0098  1     ! ENVIRONMENT:
;  R0099  1     !
;  R0100  1     !       VAX/VMS operating system. unprivileged user mode,
;  R0101  1     !
;  R0102  1     ! AUTHOR:  Greg Robert and Steve Forgey, January 1982
;  R0103  1     !
;  R0104  1     ! Modified by:
;  R0105  1     !
;  R0106  1     !       V03-002 TMH0002         Tim Halvorsen   14-Apr-1984
;  R0107  1     !               Remove MOVE_QUAD macro, which now exists in UTILDEF.
;  R0108  1     !
;  R0109  1     !       V03-001 DAS0001         David Solomon   03-Jan-1984
;  R0110  1     !               Remove two unused shared message definitions (CREATED and
;  R0111  1     !               EXISTS). Change declaration of message codes from ACC_ to ACC$_.
;  R0112  1     !               Add literal for summation table index. Add ACC$_INVACCREC.
;  R0113  1     !               Allow error code LIB$_INPSTRTRU on call to SCR$GET_SCREEN in
;  R0114  1     !               GET_REPLY macro.
;  R0115  1     !--
;  R0116  1     !
;  R0117  1     !+-------------------------------------------------------------------------+
;  R0118  1     !|                                                                         |
;  R0119  1     !|                              PROGRAM CONTROL                            |
;  R0120  1     !|                                                                         |
```

```
;  R0121  1     !----------------------------------------------------------------------+
;  R0122  1     SWITCHES
;  R0123  1             ADDRESSING_MODE(
;  R0124  1                 EXTERNAL=GENERAL,
;  R0125  1                 NONEXTERNAL=WORD_RELATIVE);
;  R0126  1
;  R0127  1
;  R0128  1     PSECT
;  R0129  1             CODE=           CODE,
;  R0130  1
;  R0131  1             PLIT=           CODE,
;  R0132  1             OWN=            DATA(ADDRESSING_MODE(LONG_RELATIVE)),
;  R0133  1             GLOBAL=         DATA;
;  R0134  1
;  R0135  1
;  R0136  1     !----------------------------------------------------------------------+
;  R0137  1     !                                                                      !
;  R0138  1     !                              INCLUDE  FILES                          !
;  R0139  1     !                                                                      !
;  R0140  1     !----------------------------------------------------------------------+
;  R0141  1
;  R0142  1     LIBRARY 'SYS$LIBRARY:STARLET';  ! VAX/VMS common definitions
;  R0143  1
;  R0144  1     REQUIRE 'SHRLIB$:UTILDEF';      ! Common VMS/DEVELOPMENT macros
```

```
;  R0145  1      !---
;  R0146  1
;  R0147  1      !       Commonly used definitions for VMS modules written in BLISS
;  R0148  1
;  R0149  1      ! Version:      'V04-000'
;  R0150  1
;  R0151  1      !****************************************************************************
;  R0152  1      !*                                                                          *
;  R0153  1      !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
;  R0154  1      !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
;  R0155  1      !*  ALL RIGHTS RESERVED.                                                    *
;  R0156  1      !*                                                                          *
;  R0157  1      !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;  R0158  1      !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;  R0159  1      !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;  R0160  1      !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;  R0161  1      !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;  R0162  1      !*  TRANSFERRED.                                                            *
;  R0163  1      !*                                                                          *
;  R0164  1      !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;  R0165  1      !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;  R0166  1      !*  CORPORATION.                                                            *
;  R0167  1      !*                                                                          *
;  R0168  1      !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;  R0169  1      !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
;  R0170  1      !*                                                                          *
;  R0171  1      !*                                                                          *
;  R0172  1      !****************************************************************************
;  R0173  1
;  R0174  1      !++
;  R0175  1      ! ABSTRACT:
;  R0176  1      !
;  R0177  1      !       This is the common require file for any module written
;  R0178  1      !       in BLISS
;  R0179  1      !
;  R0180  1      ! ENVIRONMENT:
;  R0181  1      !
;  R0182  1      !       VAX/VMS operating system.
;  R0183  1      !
;  R0184  1      ! AUTHOR:  Tim Halvorsen,  Feb 1980
;  R0185  1      !
;  R0186  1      ! MODIFIED BY:
;  R0187  1      !
;  R0188  1      !       V03-001 MHB0127         Mark Bramhall           5-Apr-1984
;  R0189  1      !               Added the MOVE_QUAD macro.
;  R0190  1      !----
```

```
;  R0191  1      !
;  R0192  1      !       Equated symbols
;  R0193  1      !
;  R0194  1
;  R0195  1      LITERAL
;  R0196  1          true        = 1,                    ! boolean true
;  R0197  1          false       = 0,                    ! boolean false
;  R0198  1          ok          = 1,                    ! success return code
;  R0199  1          error       = 2,                    ! error return code
;  R0200  1          quad        = 8;                    ! quadword allocation definition
;  R0201  1
;  R0202  1      !
;  R0203  1      !       Define structure type for VMS structures
;  R0204  1      !
;  R0205  1
;  R0206  1      STRUCTURE
;  R0207  1          bblock [o, p, s, e; n] =
;  R0208  1                      [n]
;  R0209  1                      (bblock+o)<p,s,e>;
;  R0210  1
;  R0211  1      MACRO
; MR0212  1          move_quad (src, dst) =      ! Move a quadword
; MR0213  1              BEGIN
; MR0214  1              (dst)+0 = .(src)<0, 32>;
; MR0215  1              (dst)+4 = .(src)<32,32>;
;  R0216  1              END%;
;  R0217  1
;  R0218  1      MACRO
; MR0219  1          descriptor [] =             ! Generate a static string descriptor
; MR0220  1              UPLIT (%CHARCOUNT (%STRING (%REMAINING)),
;  R0221  1                     UPLIT BYTE (%STRING (%REMAINING))) %;
;  R0222  1
;  R0223  1      MACRO
; MR0224  1          own_desciptor [] =          ! Generate the actual static string descriptor
; MR0225  1              BBLOCK [8] INITIAL(%CHARCOUNT(%STRING(%REMAINING)),
;  R0226  1                                 UPLIT BYTE (%STRING(%REMAINING))) %;
;  R0227  1
;  R0228  1      MACRO
; MR0229  1          return_if_error(command) =
; MR0230  1              BEGIN
; MR0231  1              LOCAL
; MR0232  1                  status;
; MR0233  1
; MR0234  1              status = command;
; MR0235  1              IF NOT .status
; MR0236  1              THEN
; MR0237  1                  RETURN .status;
;  R0238  1              END%;
;  R0239  1
;  R0240  1      MACRO
; MR0241  1          signal_if_error(command) =
; MR0242  1              BEGIN
; MR0243  1              LOCAL
; MR0244  1                  status;
; MR0245  1
; MR0246  1              status = command;
; MR0247  1              IF NOT .status
```

```
;MR0248  1              THEN
;MR0249  1                  BEGIN
;MR0250  1                  SIGNAL(.status);
;MR0251  1                  RETURN .status;
;MR0252  1                  END;
; R0253  1              END%;
; R0254  1
; R0255  1      !
; R0256  1      ! Macro to implement a function (f) of the message severity level that
; R0257  1      ! maps the various severity levels such that arithmetic comparisions of the
; R0258  1      ! mapped values (  f(severity) )  yield a order of precedence that is
; R0259  1      ! intuivitvely acceptable:
; R0260  1      !
; R0261  1      !
; R0262  1      !               ERROR NAME      OLDVAL      NEWVAL
; R0263  1      !
; R0264  1      !               F(SUCCESS)        1    -->   0
; R0265  1      !               F(INFORMATIONAL)  3    -->   2
; R0266  1      !               F(WARNING)        0    -->   3
; R0267  1      !               F(ERROR)          2    -->   5
; R0268  1      !               F(SEVERE_ERROR)   4    -->   7
; R0269  1      !
; R0270  1
; R0271  1      MACRO
;MR0272  1          severity_level (status) =
;MR0273  1              BEGIN
;MR0274  1              LOCAL code: BBLOCK [LONG];
;MR0275  1              code = status;
;MR0276  1              .code [sts$v_severity] - (4 * .code [sts$v_success]) + 3
; R0277  1              END%;
; R0278  1
; R0279  1      MACRO
;MR0280  1          cli$external(prefix) =
;MR0281  1              %IF %DECLARED(%QUOTE %QUOTE cli$prefix)
;MR0282  1                      %THEN UNDECLARE %QUOTE %QUOTE cli$prefix; %FI
;MR0283  1              MACRO cli$prefix = prefix %QUOTE %;
;MR0284  1              EXTERNAL LITERAL
; R0285  1                  cli$external_loop(%REMAINING)%,
; R0286  1
;MR0287  1              cli$external_loop[name] =
; R0288  1              %NAME(cli$prefix,name): UNSIGNED(8)%;
; R0289  1
; R0290  1      MACRO
;MR0291  1          $external_literal(symbol) =
;MR0292  1              BEGIN
;MR0293  1              %IF NOT %DECLARED(symbol) %THEN EXTERNAL LITERAL symbol
;MR0294  1              %IF %LENGTH GTR 1 %THEN : %REMAINING %FI; %FI
;MR0295  1              symbol
; R0296  1              END%;
; R0297  1
; R0298  1      MACRO
;MR0299  1          $fab_dev(dev_bit) =         ! Access FAB$L_DEV bits of FAB block
;MR0300  1              $BYTEOFFSET(fab$l_dev),
; R0301  1              $BITPOSITION(%NAME('dev$v_',dev_bit)),1,0%;
; R0302  1
; R0303  1      !
; R0304  1      ! $SHR_MESSAGES - a macro which defines facility-specific message codes
```

```
;  R0305  1        !       which are based on the system-wide shared message codes.
;  R0306  1        !
;  R0307  1        !       $SHR_MESSAGES( name, code, (msg,severity), ... )
;  R0308  1        !
;  R0309  1        !       where:
;  R0310  1        !           "name" is the name of the facility (e.g., COPY)
;  R0311  1        !           "code" is the corresponding facility code (e.g., 103)
;  R0312  1        !           "msg" is the name of the shared message (e.g., BEGIN)
;  R0313  1        !           "severity" is the desired message severity (e.g., 1, 0, 2)
;  R0314  1        !
;  R0315  1
;  R0316  1        MACRO
; MR0317  1            $SHR_MESSAGES( FACILITY_NAME, FACILITY_CODE ) =
; MR0318  1                %LITERAL
;  R0319  1                SHR$MSG_IDS( FACILITY_NAME, FACILITY_CODE, %REMAINING ); %,
;  R0320  1
; MR0321  1            SHR$MSG_IDS( FACILITY_NAME, FACILITY_CODE) [ VALUE ] =
;  R0322  1                SHR$MSG_CALC( FACILITY_NAME, FACILITY_CODE, %REMOVE(VALUE) ) %,
;  R0323  1
; MR0324  1            SHR$MSG_CALC( FACILITY_NAME, FACILITY_CODE, MSG_ID, SEVERITY ) =
; MR0325  1                %NAME(FACILITY_NAME,'$_',MSG_ID) = %NAME('SHR$_',MSG_ID) + FACILITY_CODE*65536 +
; MR0326  1                    %IF %DECLARED(%NAME('STS$K_',SEVERITY))
; MR0327  1                        %THEN %NAME('STS$K_',SEVERITY)
;  R0328  1                        %ELSE SEVERITY %FI %;
```

```
;  R0329  1
;  R0330  1     LIBRARY 'SYS$LIBRARY:TPAMAC';    ! TPARSE macros
```

```
;  R0331  1     !----------------------------------------------------------------------+
;  R0332  1     !                                                                      |
;  R0333  1     !                       DEFINE  EXTERNAL  REFERENCES                   |
;  R0334  1     !                                                                      |
;  R0335  1     !----------------------------------------------------------------------+
;  R0336  1
;  R0337  1     EXTERNAL LITERAL
;  R0338  1
;  R0339  1             ACC$_INVACCREC,                 ! Invalid accounting record format
;  R0340  1             ACC$_TOTAL,                     ! Totals selected/rejected message
;  R0341  1             ACC$_MERGE,                     ! "Merge phase beginning" message
;  R0342  1             ACC$_INPUT,                     ! File name/number message
;  R0343  1             ACC$_TITLETRUNC;                ! Title truncation warning
;  R0344  1
;  R0345  1
;  R0346  1     EXTERNAL ROUTINE
;  R0347  1
;  R0348  1             ADD_SYMBOL,                     ! Add a symbol to a symbol table
;  R0349  1             ALLOCATE,                       ! Gets virtual memory
;  R0350  1             FIND_WATERMARK,                 ! Determine high watermarks
;  R0351  1             HANDLER,                        ! Local signal processor
;  R0352  1             LIB$ADDX,                       ! Extended binary addition
;  R0353  1             LIB$CVT_DTB,                    ! Converts decimal to binary
;  R0354  1             LIB$CVT_HTB,                    ! Converts hex to binary
;  R0355  1             LIB$CVT_TIME,                   ! Converts ascii to binary time
;  R0356  1             LIB$DAY,                        ! Gets days since epoch
;  R0357  1             LIB$FILE_SCAN,                  ! Does wildcarding and stickiness
;  R0358  1             LIB$ICHAR,                      ! Converts charater to integer
;  R0359  1             LIB$LOOKUP_KEY,                 ! Searches keyword lists
;  R0360  1             LIB$SUBX,                       ! Extended precision subtract
;  R0361  1             LIB$SYS_ASCTIM,                 ! Converts binary time to ascii
;  R0362  1             LIB$SYS_FAO,                    ! Formatted ascii output
;  R0363  1             LIB$SYS_FACL,                   ! Library FAO routine
;  R0364  1             LIB$TPARSE,                     ! Library parse routine
;  R0365  1             LOG_FILENAME,                   ! Signals filenames and error messages
;  R0366  1             LOOKUP_SYMBOL,                  ! Lookup a symbol in a symbol table
;  R0367  1             MAP_QUALIFIERS,                 ! Establish qualifer maps
;  R0368  1             PARSE_OUTPUT_FILES,             ! Sets up output fabs, rabs
;  R0369  1             RELEASE_TO_SORT,                ! Build keys and release to sort
;  R0370  1             SCAN_SYMBOLS,                   ! Call action routine for every symbol
;  R0371  1             SHOW_RECORD,                    ! Dispatch to output routines
;  R0372  1             SOR$END_SORT,                   ! Clean up files etc.
;  R0373  1             SOR$INIT_SORT,                  ! Initialize sort routines
;  R0374  1             SOR$RELEASE_REC,                ! SORT32 record interface routine
;  R0375  1             SOR$RETURN_REC,                 ! Fetch a sorted record
;  R0376  1             SOR$SORT_MERGE,                 ! Initiate merge processing
;  R0377  1             STR$APPEND,                     ! Appends strings
;  R0378  1             STR$COMPARE,                    ! Compares two strings
;  R0379  1             STR$DUPL_CHAR,                  ! Generates a string
;  R0380  1             STR$LEFT,                       ! Extract left justified substring
;  R0381  1             STR$PREFIX,                     ! Prefix strings
;  R0382  1             STR$REPLACE,                    ! Replaces substrings
;  R0383  1             STR$RIGHT,                      ! Strips leading strings
;  R0384  1             STRIP_NEGATOR,                  ! Check/strip leading negator
;  R0385  1             STRIP_TRAIL,                    ! Strip trailing garbage
;  R0386  1             SUMMARIZE,                      ! Summation main control loop
;  R0387  1             SYS$NUMTIM,                     ! Converts times
```

```
;  R0388  1          TRANSLATE_STATUS,          ! Looks up status messages
;  R0389  1          WRITE_BAR_GRAPH,           ! Output bar graph
;  R0390  1          WRITE_BINARY,              ! Output a binary record
;  R0391  1          WRITE_SUMMARY,             ! Output summary at end of file
;  R0392  1          WRITE_TOTALS;              ! Output totals at end of program
```

```
;  R0393  1      !----------------------------------------------------------------------+
;  R0394  1      !                                                                      |
;  R0395  1      !                       DEFINE  INTERNAL  MACROS                       |
;  R0396  1      !                                                                      |
;  R0397  1      !----------------------------------------------------------------------+
;  R0398  1      MACRO
;  R0399  1
;  R0400  1      !
;  R0401  1      !COMPARE QUADWORD VALUES--
;  R0402  1      !       Macro to compare two quadword values using the user supplied
;  R0403  1      !       operator.  This macro is somewhat inefficent if the supplied
;  R0404  1      !       operator is EQL or NEQL since the expansion becomes (in part):
;  R0405  1      !
;  R0406  1      !               IF Q1 EQL Q2 THEN IF Q1 EQL Q2 THEN TRUE ELSE FALSE
;  R0407  1      !
;  R0408  1      !       However the complier may remove this inefficeny through optimization.
;  R0409  1      !       This macro works better for values that are close to one another as
;  R0410  1      !       it does the equality check of the high order words first.  This
;  R0411  1      !       macro could be improved by inspecting the supplied operator and
;  R0412  1      !       generating a tailored macro.
;  R0413  1      !
;  R0414  1
; MR0415  1      COMPARE_QUAD (Q1, OPER, Q2) =
; MR0416  1
; MR0417  1              BEGIN
; MR0418  1              BIND A = Q1: VECTOR [2,LONG];
; MR0419  1              BIND B = Q2: VECTOR [2,LONG];
; MR0420  1
; MR0421  1              IF   .A[1] EQL .B[1] THEN .A[0] OPER .B[0] ELSE .A[1] OPER .B[1]
;  R0422  1              END%,
;  R0423  1
;  R0424  1
;  R0425  1
;  R0426  1      !
;  R0427  1      ! A) Macro to describe a string
;  R0428  1      ! B) Macro to generate a quadword string descriptor
;  R0429  1      ! C) Macro to generate the address of a string descriptor
;  R0430  1      ! D) Macro to abbreviate last macro
;  R0431  1
; MR0432  1          PRIMDESC [] = %CHARCOUNT (%STRING (%REMAINING)),
;  R0433  1                        UPLIT      (%STRING (%REMAINING))%,
; MR0434  1          INITDESC [] = BBLOCK [DSC$C_S_BLN]
;  R0435  1                        INITIAL (PRIMDESC (%REMAINING))%,
;  R0436  1          ADDRDESC [] = UPLIT (PRIMDESC (%REMAINING))%,
;  R0437  1          AD []       = ADDRDESC (%REMAINING)%,
;  R0438  1
;  R0439  1
;  R0440  1      !
;  R0441  1      ! FIELD REFERENCE MACROS --
;  R0442  1      !       Define macros to reference fields.
;  R0443  1      !
;  R0444  1
;  R0445  1      KEY_W_TYPE      = 0,0,16,0%,    ! Sort key type
;  R0446  1      KEY_W_ORDER     = 2,0,16,0%,    ! Sort order
;  R0447  1      KEY_W_POS       = 4,0,16,0%,    ! Item position
;  R0448  1      KEY_W_LENGTH    = 6,0,16,0%,    ! Key length
;  R0449  1      SORT_TYPE       = 0,0,16,0%,    ! Sort key type (binary/char etc)
```

```
;  R0450  1      SORT_DESC       = 1,0,32,0%,    ! Address of item descriptor
;  R0451  1      SORT_LENGTH     = 2,0,16,0%,    ! Max. length of item in bytes
;  R0452  1
;  R0453  1
;  R0454  1
;  R0455  1      !
;  R0456  1      !SIGNAL_RETURN --
;  R0457  1      !       Signal the given arguments and then return the first parameter.
;  R0458  1      !
;  R0459  1
; MR0460  1      Signal_return (status) [] =
; MR0461  1              BEGIN
; MR0462  1              signal (status, %remaining);
; MR0463  1              return (status);
;  R0464  1              END%,
;  R0465  1
;  R0466  1
;  R0467  1
;  R0468  1
;  R0469  1      !
;  R0470  1      !PERFORM MACRO --
;  R0471  1      !       This renames the RETURN_IF_ERROR macro to be the PERFORM macro.
;  R0472  1      !
;  R0473  1
;  R0474  1      PERFORM (COMMAND) = RETURN_IF_ERROR (COMMAND)%,         ! *** HACK
;  R0475  1
;  R0476  1
;  R0477  1
;  R0478  1
;  R0479  1
;  R0480  1      !
;  R0481  1      !EXPAND FAO STRING
;  R0482  1      !       Expand an FAO directive and yeild the address of a descriptor
;  R0483  1      !       of the resultant buffer.
;  R0484  1      !
;  R0485  1
; MR0486  1              XFAO (DESC) =
; MR0487  1
; MR0488  1                      BEGIN
; MR0489  1                      EXTERNAL ROUTINE LIB$SYS_FAO: ADDRESSING_MODE (GENERAL);
; MR0490  1                      LOCAL   $$buffer:       vector [512, byte];
; MR0491  1                      LOCAL   $$buffdesc:     vector [2, long];
; MR0492  1
; MR0493  1                      $$buffdesc [0] = 512;           ! Initialize descriptor length
; MR0494  1                      $$buffdesc [1] = $$buffer;      ! Initialize descriptor address
; MR0495  1
; MR0496  1                      signal_if_error (lib$sys_fao (
; MR0497  1                              desc,                   ! Control string address
; MR0498  1                              $$buffdesc [0],         ! Resultant length
; MR0499  1                              $$buffdesc              ! Buffer descriptor
; MR0500  1                      %IF %LENGTH GTR 1  %THEN , %REMAINING %FI
; MR0501  1                              ));
; MR0502  1                      $$buffdesc
;  R0503  1                      END%,
;  R0504  1
;  R0505  1
;  R0506  1
```

```
;  R0507  1
; MR0508  1              XFAOL (DESC, LIST) =
; MR0509  1
; MR0510  1                      BEGIN
; MR0511  1                      EXTERNAL ROUTINE LIB$SYS_FAOL: ADDRESSING_MODE (GENERAL);
; MR0512  1                      LOCAL   $$buffer:       vector [512, byte];
; MR0513  1                      LOCAL   $$buffdesc:     vector [2, long];
; MR0514  1
; MR0515  1                      $$buffdesc [0] = 512;           ! Initialize descriptor length
; MR0516  1                      $$buffdesc [1] = $$buffer;      ! Initialize descriptor address
; MR0517  1
; MR0518  1                      signal_if_error (lib$sys_faol (
; MR0519  1                              desc,                   ! Control string address
; MR0520  1                              $$buffdesc [0],         ! Resultant length
; MR0521  1                              $$buffdesc,             ! Buffer descriptor
; MR0522  1                              list                    ! Arguement list
; MR0523  1                              ));
; MR0524  1                      $$buffdesc
;  R0525  1                      END%,
;  R0526  1
;  R0527  1
;  R0528  1
;  R0529  1      !
;  R0530  1      ! CLI PARSING MACROS --
;  R0531  1      !       Determine if a command line entity is present or get its value
;  R0532  1      !
;  R0533  1
; MR0534  1      GET_PRESENT (DESC) =
; MR0535  1              BEGIN
; MR0536  1              EXTERNAL ROUTINE CLI$PRESENT: ADDRESSING_MODE (GENERAL);
; MR0537  1              CLI$PRESENT (DESC)
;  R0538  1              END%,
;  R0539  1
; MR0540  1      PRESENT (QUAL_NUMB) =
; MR0541  1              BEGIN
; MR0542  1              EXTERNAL QUALIFIERS: BITVECTOR [64];
; MR0543  1              .QUALIFIERS [QUAL_NUMB]
;  R0544  1              END%,
;  R0545  1
; MR0546  1      GET_VALUE (STRING, DESC) =
; MR0547  1              BEGIN
; MR0548  1              EXTERNAL ROUTINE CLI$GET_VALUE: ADDRESSING_MODE (GENERAL);
; MR0549  1              CLI$GET_VALUE (AD (STRING), DESC)
;  R0550  1              END%,
;  R0551  1
;  R0552  1      !
;  R0553  1      !ATTRIBUTE DEFINITIONS
;  R0554  1      !
;  R0555  1
;  R0556  1              BOLD            = SCR$M_BOLD%,
;  R0557  1              REVERSE         = SCR$M_REVERSE%,
;  R0558  1              BLINK           = SCR$M_BLINK%,
;  R0559  1              UNDERLINE       = SCR$M_UNDERLINE%,
;  R0560  1
;  R0561  1
;  R0562  1      !
;  R0563  1      !SET OUTPUT MACRO --
```

```
;  R0564  1      !       This macro establishes an output stream through the screen package.
;  R0565  1      !
;  R0566  1
; MR0567  1      SET_OUTPUT (STREAM, FILENAME, USERSUB, ARGUEMENT, PREVIOUS) =
; MR0568  1              BEGIN
; MR0569  1              EXTERNAL ROUTINE SCR$SET_OUTPUT: ADDRESSING_MODE (GENERAL);
; MR0570  1              SIGNAL_IF_ERROR (SCR$SET_OUTPUT (
; MR0571  1                      %IF %NULL (STREAM)      %THEN 1 %ELSE STREAM    %FI,
; MR0572  1                      %IF %NULL (FILENAME)    %THEN 0 %ELSE FILENAME  %FI,
; MR0573  1                      %IF %NULL (USERSUB)     %THEN 0 %ELSE USERSUB   %FI,
; MR0574  1                      %IF %NULL (ARGUEMENT)   %THEN 0 %ELSE ARGUEMENT %FI,
; MR0575  1                      %IF %NULL (PREVIOUS)    %THEN 0 %ELSE PREVIOUS  %FI
; MR0576  1                              ));
;  R0577  1              END%,
;  R0578  1
;  R0579  1
;  R0580  1      !
;  R0581  1      !SCREEN MACRO --
;  R0582  1      !       This macro invokes the screen package to determine output
;  R0583  1      !       characteristics.
;  R0584  1      !
;  R0585  1
; MR0586  1      SCREEN_INFO (BUFFER) =
; MR0587  1              BEGIN
; MR0588  1              EXTERNAL ROUTINE SCR$SCREEN_INFO: ADDRESSING_MODE (GENERAL);
; MR0589  1              SIGNAL_IF_ERROR (SCR$SCREEN_INFO (BUFFER));
;  R0590  1              END%,
;  R0591  1
;  R0592  1
; MR0593  1      SCREEN (ARG) =
; MR0594  1              BEGIN
; MR0595  1              EXTERNAL SCREEN_CHAR: BBLOCK [SCR$K_LENGTH];
; MR0596  1              .SCREEN_CHAR [
; MR0597  1                      %IF %IDENTICAL (ARG, FLAGS)     %THEN SCR$L_FLAGS %FI
; MR0598  1                      %IF %IDENTICAL (ARG, WIDTH)     %THEN SCR$W_WIDTH %FI
; MR0599  1                      %IF %IDENTICAL (ARG, LENGTH)    %THEN SCR$W_PAGESIZE %FI
; MR0600  1                      %IF %IDENTICAL (ARG, TYPE)      %THEN SCR$B_DEVTYPE %FI
;  R0601  1              ] END%,
;  R0602  1
;  R0603  1
;  R0604  1      !
;  R0605  1      !SET CURSOR --
;  R0606  1      !
;  R0607  1
; MR0608  1      SET_CURSOR (LINE, COLUMN) =
; MR0609  1              BEGIN
; MR0610  1              EXTERNAL ROUTINE SCR$SET_CURSOR: ADDRESSING_MODE (GENERAL);
; MR0611  1              SIGNAL_IF_ERROR (SCR$SET_CURSOR (LINE, COLUMN))
;  R0612  1              END%,
;  R0613  1
;  R0614  1
;  R0615  1      !
;  R0616  1      !SET SCROLL --
;  R0617  1      !       Establish a scrolling region
;  R0618  1      !
;  R0619  1
; MR0620  1      SET_SCROLL (TOP, BOTTOM) =
```

```
;  MR0621  1              BEGIN
;  MR0622  1              EXTERNAL ROUTINE SCR$SET_SCROLL: ADDRESSING_MODE (GENERAL);
;  MR0623  1              SIGNAL_IF_ERROR (SCR$SET_SCROLL (
;  MR0624  1                                      TOP
;  MR0625  1                      %IF %LENGTH GTR 1 %THEN , %FI
;  MR0626  1                                      BOTTOM))
;   R0627  1              END%,
;   R0628  1
;   R0629  1
;   R0630  1
;   R0631  1      !
;   R0632  1      !ERASE SCREEN --
;   R0633  1      !
;   R0634  1
;  MR0635  1      ERASE_PAGE (LINE, COLUMN) =
;  MR0636  1              BEGIN
;  MR0637  1              EXTERNAL ROUTINE SCR$ERASE_PAGE: ADDRESSING_MODE (GENERAL);
;  MR0638  1              SIGNAL_IF_ERROR (SCR$ERASE_PAGE (
;  MR0639  1                      %IF %NULL (LINE) %THEN 1 %ELSE LINE %FI,
;  MR0640  1                      %IF %NULL (COLUMN) %THEN 1 %ELSE COLUMN %FI
;  MR0641  1                      ))
;   R0642  1              END%,
;   R0643  1
;   R0644  1
;   R0645  1
;   R0646  1      !
;   R0647  1      !ERASE LINE--
;   R0648  1      !
;   R0649  1
;  MR0650  1      ERASE_LINE (LINE, COLUMN) =
;  MR0651  1              BEGIN
;  MR0652  1              EXTERNAL ROUTINE SCR$ERASE_LINE: ADDRESSING_MODE (GENERAL);
;  MR0653  1              SIGNAL_IF_ERROR (SCR$ERASE_LINE (LINE, COLUMN))
;   R0654  1              END%,
;   R0655  1
;   R0656  1
;   R0657  1
;   R0658  1      !
;   R0659  1      !WRITE TO SCREEN --
;   R0660  1      !       Output a string to the screen with associated attributes at
;   R0661  1      !       indicated cursor position.  Do not append carriage control.
;   R0662  1      !
;   R0663  1
;  MR0664  1              WRITE_AT (DESC, LINE, COLUMN, ATTR) =
;  MR0665  1                      BEGIN
;  MR0666  1                      EXTERNAL ROUTINE SCR$PUT_SCREEN: ADDRESSING_MODE (GENERAL);
;  MR0667  1                      SIGNAL_IF_ERROR (SCR$PUT_SCREEN (
;  MR0668  1                              DESC,
;  MR0669  1                              LINE,
;  MR0670  1                              COLUMN,
;  MR0671  1                              %IF %NULL (ATTR) %THEN 0 %ELSE ATTR %FI
;  MR0672  1                              ))
;   R0673  1                      END%,
;   R0674  1
;   R0675  1      !
;   R0676  1      !WRITE LINE TO SCREEN --
;   R0677  1      !       Output a string to the screen with associated attributes,
```

```
;  R0678  1      !       scroll the indicated number of LINES, and append a carriage return.
;  R0679  1      !
;  R0680  1
; MR0681  1              WRITE_LINE (DESC, LINES, ATTR) =
; MR0682  1                      BEGIN
; MR0683  1                      EXTERNAL ROUTINE SCR$PUT_LINE: ADDRESSING_MODE (GENERAL);
; MR0684  1                      SIGNAL_IF_ERROR (SCR$PUT_LINE (
; MR0685  1                              DESC,
; MR0686  1                              %IF %NULL (LINES) %THEN 1 %ELSE LINES %FI,
; MR0687  1                              %IF %NULL (ATTR) %THEN 0 %ELSE ATTR %FI
; MR0688  1                              ))
;  R0689  1                      END%,
;  R0690  1
;  R0691  1
;  R0692  1
;  R0693  1
;  R0694  1      !
;  R0695  1      ! WRITE FAO STRING TO SCREEN --
;  R0696  1      !       Write a STRING to the screen at the given line and
;  R0697  1      !       column with no attributers after first processing it
;  R0698  1      !       through FAO.
;  R0699  1      !
;  R0700  1
; MR0701  1              WRITE_FAO_AT (DESC,LINE,COLUMN) =
; MR0702  1
; MR0703  1                      WRITE_AT (
; MR0704  1                              XFAO (DESC
; MR0705  1                                      %IF %LENGTH GTR 3 %THEN , %REMAINING %FI
; MR0706  1                                      ),
; MR0707  1                              LINE,
; MR0708  1                              COLUMN,
;  R0709  1                              0)%,
;  R0710  1
;  R0711  1      !
;  R0712  1      ! GET INPUT
;  R0713  1      !
;  R0714  1
; MR0715  1      GET_REPLY (REPLY, PROMPT, LENGTH) =
; MR0716  1              BEGIN
; MR0717  1              EXTERNAL ROUTINE SCR$GET_SCREEN: ADDRESSING_MODE (GENERAL);
; MR0718  1              EXTERNAL LITERAL
; MR0719  1                  LIB$_INPSTRTRU;
; MR0720  1
; MR0721  1              LOCAL
; MR0722  1                  STATUS;
; MR0723  1
; MR0724  1              %IF %NULL (REPLY) %THEN
; MR0725  1                      LOCAL   $$TEMP,
; MR0726  1                              $$TEMPDESC: VECTOR [2];
; MR0727  1                      $$TEMPDESC [0] = 4;
; MR0728  1                      $$TEMPDESC [1] = $$TEMP;
; MR0729  1              %FI
; MR0730  1              STATUS = SCR$GET_SCREEN (
; MR0731  1                      %IF %NULL (REPLY) %THEN $$TEMPDESC %ELSE REPLY %FI,
; MR0732  1                      PROMPT,
; MR0733  1                      LENGTH);
; MR0734  1              IF NOT .STATUS AND ( .STATUS NEQU LIB$_INPSTRTRU )
```

```
;  MR0735  1                THEN
;  MR0736  1                    BEGIN
;  MR0737  1                    SIGNAL( .STATUS );
;  MR0738  1                    RETURN .STATUS;
;  MR0739  1                    END;
;   R0740  1                END%,
;   R0741  1
;   R0742  1
;   R0743  1        !
;   R0744  1        !SET AND FLUSH BUFFER
;   R0745  1        !
;   R0746  1
;  MR0747  1        SET_BUFFER (BUFFER) =
;  MR0748  1                BEGIN
;  MR0749  1                EXTERNAL ROUTINE SCR$SET_BUFFER: ADDRESSING_MODE (GENERAL);
;  MR0750  1                SIGNAL_IF_ERROR (SCR$SET_BUFFER (BUFFER))
;   R0751  1                END%,
;   R0752  1
;  MR0753  1        PUT_BUFFER (BUFFER) =
;  MR0754  1                BEGIN
;  MR0755  1                EXTERNAL ROUTINE SCR$PUT_BUFFER: ADDRESSING_MODE (GENERAL);
;  MR0756  1                SIGNAL_IF_ERROR (SCR$PUT_BUFFER (BUFFER))
;   R0757  1                END%,
;   R0758  1
;   R0759  1
;   R0760  1
;   R0761  1
;   R0762  1        !
;   R0763  1        !LOOKUP MACRO --
;   R0764  1        !       This macro invokes lib$lookup_key and, if the lookup fails,
;   R0765  1        !       signals the user with the status and the failed value and
;   R0766  1        !       returns to the caller.
;   R0767  1        !
;   R0768  1
;  MR0769  1        LOOKUP (KEY, TABLE, RESULT) =
;  MR0770  1                BEGIN
;  MR0771  1                EXTERNAL ROUTINE LIB$LOOKUP_KEY: ADDRESSING_MODE (GENERAL);
;  MR0772  1                LOCAL STATUS;
;  MR0773  1                IF NOT (STATUS = LIB$LOOKUP_KEY (KEY, TABLE, RESULT))
;  MR0774  1                THEN SIGNAL_RETURN (MSG$_SYNTAX, 1, KEY);
;   R0775  1                END;%,
;   R0776  1
;   R0777  1        !
;   R0778  1        ! THIS RENAMES THE $BYTEOFFSET MACRO TO BE $BOFF FOR BREVITY
;   R0779  1        !
;   R0780  1
;   R0781  1        $BOFF (arg) = $BYTEOFFSET (arg)%;       ! Make a shorter name
;   R0782  1
;   R0783  1
;   R0784  1
;   R0785  1        !----------------------------------------------------------------------+
;   R0786  1        !                                                                      !
;   R0787  1        !                       DEFINE  INTERNAL  STRUCTURES                   !
;   R0788  1        !                                                                      !
;   R0789  1        !----------------------------------------------------------------------+
```

```
;  R0790  1      !----------------------------------------------------------------------+
;  R0791  1      !                                                                      |
;  R0792  1      !                        DEFINE  MESSAGE  CODES                        |
;  R0793  1      !                                                                      |
;  R0794  1      !----------------------------------------------------------------------+
;  R0795  1
; PR0796  1      $SHR_MESSAGES (MSG.159,                    ! COMMON I/O AND MISC. MESSAGES
; PR0797  1
; PR0798  1      !       ********     ************             ***************
; PR0799  1      !       * NAME *     * SEVERITY *             * DESCRIPTION *
; PR0800  1      !       ********     ************             ***************
; PR0801  1
; PR0802  1              (SEARCHFAIL,    ERROR),          !  -Error while searching for file
; PR0803  1              (OPENIN,        ERROR),          !  -Unable to open or connect to input
; PR0804  1              (READERR,       ERROR),          !  -Error while reading input
; PR0805  1              (CLOSEIN,       ERROR),          !  -Unable to close output
; PR0806  1              (OPENOUT,       ERROR),          !  -Unable to create, open, or connect
; PR0807  1                                               !     to output
; PR0808  1              (WRITEERR,      ERROR),          !  -Error while writing output
; PR0809  1              (CLOSEOUT,      ERROR),          !  -Unable to close output
; PR0810  1              (SYNTAX,        SEVERE),         !  -Parse failure
;  R0811  1              );
;  R0812  1
;  R0813  1
```

```
;  R0814  1     !----------------------------------------------------------------------+
;  R0815  1     !                                                                      !
;  R0816  1     !                          EQUATED  SYMBOLS                            !
;  R0817  1     !                                                                      !
;  R0818  1     !----------------------------------------------------------------------+
;  R0819  1
;  R0820  1     LITERAL
;  R0821  1
;  R0822  1     !
;  R0823  1     !FLAGS AND MISCELLANEOUS VALUES --
;  R0824  1     !
;  R0825  1
;  R0826  1             COLUMNS_PER_GROUP = 15,         ! Bar graph column grouping factor
;  R0827  1             REC_PREFIX        = 8,          ! Size of data prefixed to record
;  R0828  1             ACCSK_UNKNOWN     = 0,          ! Reserve the value 0 for unknown types
;  R0829  1             NEGATOR           = '-',        ! Define list negator charater
;  R0830  1
;  R0831  1     !
;  R0832  1     !QUALIFIER NUMBERS
;  R0833  1     ! Define local bitnumbers for qualifers (also used as symbol table index
;  R0834  1     ! numbers for summation).
;  R0835  1     !
;  R0836  1             ACCOUNT           = 00,
;  R0837  1             BAR_GRAPH         = 01,
;  R0838  1             BEFORE            = 02,
;  R0839  1             BINARY            = 03,
;  R0840  1             USER              = 04,
;  R0841  1             ENTRY             = 05,
;  R0842  1             FULL              = 06,
;  R0843  1             IDENT             = 07,
;  R0844  1             IMAGE             = 08,
;  R0845  1             JOB               = 09,
;  R0846  1             LOG               = 10,
;  R0847  1             ADDRESS           = 11,
;  R0848  1             NODE              = 12,
;  R0849  1             OWNER             = 13,
;  R0850  1             OUTPUT            = 14,
;  R0851  1             PRIORITY          = 15,
;  R0852  1             PROCESS           = 16,
;  R0853  1             QUEUE             = 17,
;  R0854  1             REJECTED          = 18,
;  R0855  1             REMOTE_ID         = 19,
;  R0856  1             REPORT            = 20,
;  R0857  1             SINCE             = 21,
;  R0858  1             STATUS            = 22,
;  R0859  1             SORT              = 23,
;  R0860  1             SUMMARY           = 24,
;  R0861  1             TERMINAL          = 25,
;  R0862  1             TITLE             = 26,
;  R0863  1             TYPE              = 27,
;  R0864  1             UIC               = 28,
;  R0865  1             QUAL_COUNT        = 29,
;  R0866  1             EXPAND_DATE       = 31,         ! See /REPORT parsing
;  R0867  1             UIC_GROUP         = 32,
;  R0868  1             UIC_MEMBER        = 33,
;  R0869  1
;  R0870  1             SUMMATION_TABLE = 63,         ! Must be different than above numbers.
```

```
;  R0871  1
;  R0872  1        !
;  R0873  1        !SUMMATION TYPES --
;  R0874  1        ! These numbers describes the various summation rules for fields
;  R0875  1        ! in accounting records when /SUMMARY is invoked
;  R0876  1        !
;  R0877  1                SUM_TYPE_ADD    = 0,            ! Simple longword addition
;  R0878  1                SUM_TYPE_ADDX   = 1,            ! Extended longword addition
;  R0879  1                SUM_TYPE_PEAK   = 2,            ! Longword peak value recording
;  R0880  1                SUM_TYPE_INCR   = 3,            ! Increment per occurence
;  R0881  1                SUM_TYPE_TYPE   = 4,            ! Increment if type matches
;  R0882  1                SUM_TYPE_ELAP   = 5,            ! Summarize connect time
;  R0883  1
;  R0884  1                SUM_ENT_TYPE    = 0,            ! Type of summation entry
;  R0885  1                SUM_ENT_ADR     = 1,            ! Address or other value
;  R0886  1                SUM_ENT_BSIZE   = 2,            ! Accumulation bucket size in longwords
;  R0887  1                SUM_ENT_FAO     = 3,            ! Address of FAO directive
;  R0888  1                SUM_ENT_HDR1    = 4,            ! Address of 1st column header
;  R0889  1                SUM_ENT_HDR2    = 5,            ! Address of 2nd column header
;  R0890  1
;  R0891  1
;  R0892  1        !
;  R0893  1        !MAXIMUM TABLE SIZES --
;  R0894  1        ! Specify the maximum number of entries the user can make in qualifier
;  R0895  1        ! value lists for /SORT and /SUMMARIZE.
;  R0896  1        !
;  R0897  1
;  R0898  1            MAX_DEFAULT         = 30,           ! Default maximum
;  R0899  1            MAX_SUM             = MAX_DEFAULT,  ! Maximum number of /SUMMARY values
;  R0900  1            MAX_REPORT          = MAX_DEFAULT,  ! Maximum number of /REPORT values
;  R0901  1            MAX_SORT            = 10;           ! Sort keys -- sort package limit
```

```
;    64          0902  1
;    65          0903  1 !----------------------------------------------------------------------+
;    66          0904  1 !                                                                      !
;    67          0905  1 !                          TABLE  OF  CONTENTS                         !
;    68          0906  1 !                                                                      !
;    69          0907  1 !----------------------------------------------------------------------+
```

```
;    71          0908  1 UNDECLARE WRITE_BINARY;
;    72          0909  1
;    73          0910  1 GLOBAL ROUTINE WRITE_BINARY (BUFFER, RAB) =
;    74          0911  1
;    75          0912  1 !----
;    76          0913  1 !
;    77          0914  1 ! Functional description
;    78          0915  1 !
;    79          0916  1 !       This routine accepts a pointer to a buffer and writes
;    80          0917  1 !       the buffer to an output stream in binary format.
;    81          0918  1 !
;    82          0919  1 ! Input parameters
;    83          0920  1 !
;    84          0921  1 !       BUFFER = address of an input record buffer
;    85          0922  1 !                the size of the record is stored in the second
;    86          0923  1 !                word of the buffer
;    87          0924  1 !
;    88          0925  1 !       RAB =    address of output rab
;    89          0926  1 !
;    90          0927  1 !----
;    91          0928  1
;    92          0929  2 BEGIN
;    93          0930  2
;    94          0931  2 MAP
;    95          0932  2     rab:        ref bblock,             ! Pointer to rab
;    96          0933  2     buffer:     ref bblock;             ! Describe the input buffer
;    97          0934  2
;    98          0935  2 LOCAL
;    99          0936  2         desc:   vector [2, long];       ! Temporary string descriptor
;   100          0937  2
;   101          0938  2
;   102          0939  2 If .rab eql 0 then return true;         ! Exit immediately if no output
;   103          0940  2
;   104          0941  2 !
;   105          0942  2 !INITIALIZE THE RAB
;   106          0943  2 !       Store the buffer address and length in the RAB.
;   107          0944  2 !
;   108          0945  2
;   109          0946  2 rab [rab$l_rbf] = .buffer;                      ! Store buffer address in RAB
;   110          0947  2 rab [rab$w_rsz] = .buffer [acc$w_msgsiz];       ! Store buffer size in RAB
;   111          0948  2
;   112          0949  2
;   113          0950  2
;   114          0951  2 !
;   115          0952  2 !WRITE TO FILE ---
;   116          0953  2 !       Output the buffer via RMS.
;   117          0954  2 !
;   118          0955  2
;   119        P 0956  2 Perform  ($put (                          ! Call RMS with
;   120        P 0957  2                rab = .rab,                !  -record stream identifier
;   121          0958  2                err = log_filename));      !  -error action routine
;   122          0959  2
;   123          0960  2 return true;
;   124          0961  1 END;

                                                        .TITLE  UTILITY
```

```
                                                        .IDENT  \V04-000\

                                                        .EXTRN  ACC$_INVACCREC, ACC$_TOTAL
                                                        .EXTRN  ACC$_MERGE, ACC$_INPUT
                                                        .EXTRN  ACC$_TITLETRUNC
                                                        .EXTRN  ADD_SYMBOL, ALLOCATE
                                                        .EXTRN  FIND_WATERMARK, HANDLER
                                                        .EXTRN  LIB$ADDX, LIB$CVT_DTB
                                                        .EXTRN  LIB$CVT_HTB, LIB$CVT_TIME
                                                        .EXTRN  LIB$DAY, LIB$FILE_SCAN
                                                        .EXTRN  LIB$ICHAR, LIB$LOOKUP_KEY
                                                        .EXTRN  LIB$SUBX, LIB$SYS_ASCTIM
                                                        .EXTRN  LIB$SYS_FAO, LIB$SYS_FAOL
                                                        .EXTRN  LIB$TPARSE, LOG_FILENAME
                                                        .EXTRN  LOOKUP_SYMBOL, MAP_QUALIFIERS
                                                        .EXTRN  PARSE_OUTPUT_FILES
                                                        .EXTRN  RELEASE_TO_SORT
                                                        .EXTRN  SCAN_SYMBOLS, SHOW_RECORD
                                                        .EXTRN  SOR$END_SORT, SOR$INIT_SORT
                                                        .EXTRN  SOR$RELEASE_REC
                                                        .EXTRN  SOR$RETURN_REC, SOR$SORT_MERGE
                                                        .EXTRN  STR$APPEND, STR$COMPARE
                                                        .EXTRN  STR$DUPL_CHAR, STR$LEFT
                                                        .EXTRN  STR$PREFIX, STR$REPLACE
                                                        .EXTRN  STR$RIGHT, STRIP_NEGATOR
                                                        .EXTRN  STRIP_TRAIL, SUMMARIZE
                                                        .EXTRN  SYS$NUMTIM, TRANSLATE_STATUS
                                                        .EXTRN  WRITE_BAR_GRAPH
                                                        .EXTRN  WRITE_BINARY, WRITE_SUMMARY
                                                        .EXTRN  WRITE_TOTALS, SYS$PUT

                                                        .PSECT  CODE,NOWRT,2

                                        0000 00000      .ENTRY  WRITE_BINARY, Save nothing                      ; 0910
                                    5E      08  C2 00002       SUBL2   #8, SP                                  ;
                                    51  08  AC  D0 00005       MOVL    RAB, R1                                 ; 0939
                                        1F  13  13 00009       BEQL    1$                                      ;
                                    50  04  AC  D0 0000B       MOVL    BUFFER, R0                              ; 0946
                                28  A1      50  D0 0000F       MOVL    R0, 40(R1)                              ;
                                22  A1  02  A0  B0 00013       MOVW    2(R0), 34(R1)                           ; 0947
                                    00000000G   00  9F 00018   PUSHAB  LOG_FILENAME                            ; 0958
                                        51  DD 0001E           PUSHL   R1                                      ;
                            00000000G   00  02  FB 00020       CALLS   #2, SYS$PUT                             ;
                                    03      50  E9 00027       BLBC    STATUS, 2$                              ;
                                    50      01  D0 0002A 1$:   MOVL    #1, R0                                  ; 0960
                                            04  0002D 2$:     RET                                             ; 0961

; Routine Size:  46 bytes,    Routine Base:  CODE + 0000
```

```
;  126          0962  1 UNDECLARE STRIP_NEGATOR;
;  127          0963  1
;  128          0964  1 GLOBAL ROUTINE STRIP_NEGATOR (TARGET) =
;  129          0965  1
;  130          0966  1 !----
;  131          0967  1 !
;  132          0968  1 ! Functional description
;  133          0969  1 !
;  134          0970  1 !       This routine is called to strip list negators.  If the first
;  135          0971  1 !       character of the string pointed to by the descriptor is the
;  136          0972  1 !       negator character then strip that character from the string and
;  137          0973  1 !       return TRUE else return FALSE.
;  138          0974  1 !
;  139          0975  1 ! Input parameters
;  140          0976  1 !
;  141          0977  1 !       TARGET = Address of a descriptor
;  142          0978  1 !
;  143          0979  1 ! Output parameters
;  144          0980  1 !
;  145          0981  1 !       If the item is negated return FALSE
;  146          0982  1 !       Else return TRUE
;  147          0983  1 !       Any errors encountered are RETURNed immediately.
;  148          0984  1 !
;  149          0985  1 !----
;  150          0986  1
;  151          0987  2 BEGIN
;  152          0988  2
;  153          0989  2
;  154          0990  2 MAP
;  155          0991  2         target: ref bblock [dsc$k_d_bln];! Describe te input parameter
;  156          0992  2
;  157          0993  2
;  158          0994  2 !
;  159          0995  2 !TEST FIRST CHARACTER --
;  160          0996  2 !       If its te negator character then strip the character and
;  161          0997  2 !       return false.
;  162          0998  2 !
;  163          0999  2
;  164          1000  2 If  .target [dsc$w_length] neq 0                ! If string is non-null
;  165          1001  2 and lib$ichar (.target) eql negator             ! - and char[1] = negator
;  166          1002  3 then BEGIN
;  167          1003  3      str$right (.target, .target, %ref(2));     ! - then strip it off
;  168          1004  3      return false;                              ! - and tell the user
;  169          1005  2      END;
;  170          1006  2
;  171          1007  2 return true;
;  172          1008  1 END;


                                0004 00000          .ENTRY  STRIP_NEGATOR, Save R2                  ; 0964
                            5E  04  C2 00002        SUBL2   #4, SP                                  ;
                            52  04  AC  D0 00005    MOVL    TARGET, R2                              ; 1000
                                62  B5 00009        TSTW    (R2)                                    ;
                                20  13 0000B        BEQL    1$                                      ;
```

```
                                        52  DD 0000D          PUSHL   R2                                          ; 1001
                00000000G   00          01  FB 0000F          CALLS   #1, LIB$ICHAR                               ;
                            2D          50  D1 00016          CMPL    R0, #45                                     ;
                                        12  12 00019          BNEQ    1$                                          ;
                            6E          02  D0 0001B          MOVL    #2, (SP)                                    ; 1003
                                  4004  8F  BB 0001E          PUSHR   #^M<R2,SP>                                  ;
                                        52  DD 00022          PUSHL   R2                                          ;
                00000000G   00          03  FB 00024          CALLS   #3, STR$RIGHT                               ;
                                        04  11 0002B          BRB     2$                                          ; 1004
                            50          01  D0 0002D 1$:      MOVL    #1, R0                                      ; 1007
                                        04  00030             RET                                                 ;
                                        50  D4 00031 2$:      CLRL    R0                                          ; 1008
                                            04 00033          RET                                                 ;

; Routine Size:  52 bytes,    Routine Base:  CODE + 002E
```

```
;  174          1009  1 UNDECLARE STRIP_TRAIL;
;  175          1010  1
;  176          1011  1 GLOBAL ROUTINE STRIP_TRAIL (DESC) =
;  177          1012  1
;  178          1013  1 !----
;  179          1014  1 !
;  180          1015  1 ! Functional description
;  181          1016  1 !
;  182          1017  1 !       This routine is called to strip garbage from the end of a string.
;  183          1018  1 !
;  184          1019  1 ! Input parameters
;  185          1020  1 !
;  186          1021  1 !       DESC    = address of a descriptor of the string
;  187          1022  1 !
;  188          1023  1 ! Output parameters
;  189          1024  1 !
;  190          1025  1 !       The byte count in the descriptor is decremented by the
;  191          1026  1 !       number of bytes of trailing garbage.  Garbage is defined
;  192          1027  1 !       as NULLS, TABS, and SPACES.
;  193          1028  1 !
;  194          1029  1 !----
;  195          1030  1
;  196          1031  2 BEGIN
;  197          1032  2
;  198          1033  2 MAP
;  199          1034  2         desc:   ref bblock [dsc$k_d_bln];       ! Address of a descriptor
;  200          1035  2
;  201          1036  3 decr ptr from (.desc [dsc$a_pointer] + .desc [dsc$w_length] - 1)
;  202          1037  3           to   (.desc [dsc$a_pointer])
;  203          1038  2 do
;  204          1039  2         if .(.ptr)<0,8> eql 0                   ! NULLS
;  205          1040  2         or .(.ptr)<0,8> eql '   '               ! TABS
;  206          1041  2         or .(.ptr)<0,8> eql ' '                 ! SPACES
;  207          1042  2         then desc [dsc$w_length] = .desc [dsc$w_length] - 1
;  208          1043  2         else exitloop;
;  209          1044  2
;  210          1045  2
;  211          1046  2 return true;
;  212          1047  2
;  213          1048  1 END;


                                  0004 00000          .ENTRY  STRIP_TRAIL, Save R2                    ; 1011
                        51    04  AC  D0 00002          MOVL    DESC, R1                                ; 1036
                        52    04  A1  D0 00006          MOVL    4(R1), R2                               ;
                        50        61  3C 0000A          MOVZWL  (R1), R0                                ;
                        50        52  C0 0000D          ADDL2   R2, R0                                  ;
                                  10  11 00010          BRB     3$                                      ;
                                  60  95 00012 1$:      TSTB    (PTR)                                   ; 1039
                                  0A  13 00014          BEQL    2$                                      ;
                        09        60  91 00016          CMPB    (PTR), #9                               ; 1040
                                  05  13 00019          BEQL    2$                                      ;
                        20        60  91 0001B          CMPB    (PTR), #32                              ; 1041
                                  09  12 0001E          BNEQ    4$                                      ;
```

```
                        61  B7 00020 2$:    DECW    (R1)                                    ; 1042
                        50  D7 00022 3$:    DECL    PTR                                     ; 1039
              52        50  D1 00024        CMPL    PTR, R2                                 ;
                        E9  18 00027        BGEQ    1$                                      ;
              50        01  D0 00029 4$:    MOVL    #1, R0                                  ; 1046
                            04 0002C        RET                                             ; 1048

; Routine Size:  45 bytes,    Routine Base:  CODE + 0062
```

```
;  215     1049  1 UNDECLARE TRANSLATE_STATUS;
;  216     1050  1
;  217     1051  1 GLOBAL ROUTINE TRANSLATE_STATUS (EXIT_STATUS,DESCRIP) =
;  218     1052  1
;  219     1053  1 !++
;  220     1054  1 !
;  221     1055  1 ! Functional description:
;  222     1056  1 !
;  223     1057  1 !       This routine will return the text associated with a given exit status.
;  224     1058  1 !       If the status code has no text associated with it, return a routine
;  225     1059  1 !       status value of SS$_MSGNOTFND, and the text "<no text>" in DESCRIP.
;  226     1060  1 !
;  227     1061  1 ! Input:
;  228     1062  1 !
;  229     1063  1 !       EXIT_STATUS     - a longword containing the exit status
;  230     1064  1 !
;  231     1065  1 ! Output:
;  232     1066  1 !
;  233     1067  1 !       DESCRIP         - address of a quadword descriptor that points to the
;  234     1068  1 !                         buffer to receive the message text.  Note that the
;  235     1069  1 !                         size field of the descriptor will reflect the actual
;  236     1070  1 !                         message length.
;  237     1071  1 !
;  238     1072  1 !--
;  239     1073  1
;  240     1074  2 BEGIN
;  241     1075  2
;  242     1076  2 MAP
;  243     1077  2         exit_status     : REF bblock [LONG],
;  244     1078  2         descrip         : REF BBLOCK [dsc$c_s_bln];
;  245     1079  2
;  246     1080  2 OWN
;  247     1081  2         no_text_msg     : initdesc ('<no text>');
;  248     1082  2
;  249     1083  2 LOCAL
;  250     1084  2         status          : bblock [LONG];
;  251     1085  2
;  252     1086  2 !
;  253     1087  2 !TRANSLATE THE EXIT STATUS ---
;  254     1088  2 !
;  255     1089  2
;  256   P 1090  3 status= ($getmsg (                        ! Call message translator with
;  257   P 1091  3         msgid  = .exit_status,           !  -message code
;  258   P 1092  3         msglen = descrip [dsc$w_length],!  -$GETMSG returns length here
;  259     1093  2         bufadr = .descrip));             !  -address of work buffer desc
;  260     1094  2
;  261     1095  2 !
;  262     1096  2 ! If the status code returned by $GETMSG
;  263     1097  2 ! was SS$_MSGNOTFND, then return the 'no text' message.
;  264     1098  2 !
;  265     1099  2
;  266     1100  2 IF .status EQL ss$_msgnotfnd
;  267     1101  2 THEN
;  268     1102  3     BEGIN
;  269     1103  3     CH$MOVE ( .no_text_msg [dsc$w_length],      ! Length
;  270     1104  3               .no_text_msg [dsc$a_pointer],     ! Source address
;  271     1105  3               .descrip [dsc$a_pointer]          ! Destination address
```

```
;  272          1106  3              );
;  273          1107  3      descrip [dsc$w_length] = .no_text_msg [dsc$w_length];
;  274          1108  2      END;
;  275          1109  2
;  276          1110  2  RETURN .status
;  277          1111  1  END;


                                                        .PSECT  DATA,NOEXE,2

                                        00000009  00000 NO_TEXT_MSG:
                                                        .LONG   9
                                        00000000' 00004         .ADDRESS P.AAA                                  ;

                                                        .PSECT  CODE,NOWRT,2

                                                  0008F         .BLKB   1
00  00  00  3E  74  78  65  74  20  6F  6E  3C  00090 P.AAA:    .ASCII  \<no text>\<0><0><0>                    ;

                                                        .EXTRN  SYS$GETMSG

                                                    01FC 00000  .ENTRY  TRANSLATE_STATUS, Save R2,R3,R4,R5,R6,R7,R8 ; 1051
                                  58 00000000'  EF   9E 00002   MOVAB   NO_TEXT_MSG, R8                           ;
                                  7E            0F   7D 00009   MOVQ    #15, -(SP)                              ; 1093
                                  56       08   AC   D0 0000C   MOVL    DESCRIP, R6                             ;
                                               56   DD 00010   PUSHL   R6                                      ;
                                               56   DD 00012   PUSHL   R6                                      ;
                                           04  AC   DD 00014   PUSHL   EXIT_STATUS                             ;
                              00000000G    00       05   FB 00017   CALLS   #5, SYS$GETMSG                      ;
                                  57               50   D0 0001E   MOVL    R0, STATUS                          ;
                              00000621     8F       57   D1 00021   CMPL    STATUS, #1569                       ; 1100
                                               09   12 00028   BNEQ    1$                                      ;
                 04   B6       04   B8         68   28 0002A   MOVC3   NO_TEXT_MSG, @NO_TEXT_MSG+4, @4(R6)    ; 1105
                                  66           68   B0 00030   MOVW    NO_TEXT_MSG, (R6)                       ; 1107
                                  50           57   D0 00033 1$:  MOVL    STATUS, R0                          ; 1110
                                               04   00036       RET                                             ; 1111

; Routine Size:  55 bytes,    Routine Base:  CODE + 009C
```

```
;   279          1112  1 UNDECLARE LOG_FILENAME;
;   280          1113  1
;   281          1114  1 GLOBAL ROUTINE LOG_FILENAME (RMS) =
;   282          1115  1
;   283          1116  1 !----
;   284          1117  1 !
;   285          1118  1 ! Functional description
;   286          1119  1 !
;   287          1120  1 !       This routine is called to signal a message to
;   288          1121  1 !       the user based on an error code and file name
;   289          1122  1 !       that are imbedded in the passed parameter.
;   290          1123  1 !
;   291          1124  1 ! Input parameters
;   292          1125  1 !
;   293          1126  1 !       RMS = Either a FAB or a RAB
;   294          1127  1 !       RAB$L_FAB = pointer to fab block        (If input was a RAB)
;   295          1128  1 !       FAB$L_NAM = pointer to name block
;   296          1129  1 !       RAB$L_CTX = error message to be used    (If input was a RAB)
;   297          1130  1 !       FAB$L_CTX = error message to be used    (If input was a FAB)
;   298          1131  1 !
;   299          1132  1 !
;   300          1133  1 ! Output parameters
;   301          1134  1 !
;   302          1135  1 !       Expanded error messages to user
;   303          1136  1 !       Status is RETURNed
;   304          1137  1 !
;   305          1138  1 !----
;   306          1139  1
;   307          1140  2 BEGIN
;   308          1141  2
;   309          1142  2 MAP
;   310          1143  2     rms:  ref bblock;                          ! Define block format
;   311          1144  2
;   312          1145  2
;   313          1146  2 LOCAL
;   314          1147  2         fab: ref bblock,                       ! Pointer to FAB block
;   315          1148  2         nam: ref bblock,                       ! Pointer to NAM block
;   316          1149  2         rms_sts,                               ! Temporary primary status holder
;   317          1150  2         rms_stv,                               ! Temporary secondary status holder
;   318          1151  2         rms_ctx,                               ! Temporary user context holder
;   319          1152  2         status: bblock [long],                 ! Local "catch all" status return
;   320          1153  2         desc:   vector [2, long];              ! Temporary string descriptor
;   321          1154  2
;   322          1155  2
;   323          1156  2 !
;   324          1157  2 !SET UP VALUES --
;   325          1158  2 !       Fetch the primary and secondary status values and the user
;   326          1159  2 !       context field from the RMS structure.  If a RAB was passed
;   327          1160  2 !       then fetch the address of the associated FAB.
;   328          1161  2 !
;   329          1162  2
;   330          1163  2 If .rms [rab$b_bid] eql rab$c_bid then  ! If this is a rab
;   331          1164  2         BEGIN
;   332          1165  2         fab = .rms [rab$l_fab];
;   333          1166  2         rms_sts = .rms [rab$l_sts];
;   334          1167  2         rms_stv = .rms [rab$l_stv];
;   335          1168  2         rms_ctx = .rms [rab$l_ctx];
```

```
;  336     1169  3          END
;  337     1170  3      else BEGIN
;  338     1171  3           fab = .rms;
;  339     1172  3           rms_sts = .rms [fab$l_sts];
;  340     1173  3           rms_stv = .rms [fab$l_stv];
;  341     1174  3           rms_ctx = .rms [fab$l_ctx];
;  342     1175  2           END;
;  343     1176  2
;  344     1177  2 nam = .fab [fab$l_nam];                 ! Fetch address of NAM block
;  345     1178  2
;  346     1179  2
;  347     1180  2
;  348     1181  2 !
;  349     1182  2 !CHECK FOR EOF --
;  350     1183  2 !       End of file errors are not reported by this routine.
;  351     1184  2 !
;  352     1185  2
;  353     1186  2 If  .rms [rab$b_bid] eql rab$c_bid       ! If this is a rab
;  354     1187  2 and .rms_sts eql rms$_eof               !  - and error is end of file
;  355     1188  2 and .rms_ctx eql msg$_readerr           !  - and this was a read call
;  356     1189  2     then return rms$_eof;               ! don't bother to report it
;  357     1190  2
;  358     1191  2
;  359     1192  2
;  360     1193  2 !
;  361     1194  2 !FETCH FILE NAME --
;  362     1195  2 !       Find the best filename available.  Start with the
;  363     1196  2 !       resultant name; if not present try for the expanded
;  364     1197  2 !       name; if also missing then settle for the original
;  365     1198  2 !       file name.
;  366     1199  2 !
;  367     1200  2
;  368     1201  2 If .nam[nam$b_rsl] neq 0 then            ! IF result string nonblank,
;  369     1202  3     BEGIN
;  370     1203  3     desc[0] = .nam[nam$b_rsl];          ! then display it
;  371     1204  3     desc[1] = .nam[nam$l_rsa];
;  372     1205  3     END
;  373     1206  3
;  374     1207  2 else if .nam[nam$b_esl] neq 0 then       ! Or if expanded name nonblank
;  375     1208  3     BEGIN
;  376     1209  3     desc[0] = .nam[nam$b_esl];          ! then display it
;  377     1210  3     desc[1] = .nam[nam$l_esa];
;  378     1211  3     END
;  379     1212  3
;  380     1213  3 else BEGIN
;  381     1214  3     desc[0] = .fab[fab$b_fns];          ! Otherwise, use original
;  382     1215  3     desc[1] = .fab[fab$l_fna];          ! name string in FAB
;  383     1216  2     END;
;  384     1217  2
;  385     1218  2
;  386     1219  2
;  387     1220  2 !
;  388     1221  2 !NOTIFY THE USER --
;  389     1222  2 !       Construct an error message using the user supplied context (CTX)
;  390     1223  2 !       field and the RMS supplied primary (STS) and secondary (STV)
;  391     1224  2 !       status fields.  Signal it to the user.
;  392     1225  2 !
```

```
;  393          1226  2
;  394          1227  2 signal (.rms_ctx, 1 ,desc,                ! Output an error message
;  395          1228  2                   .rms_sts,               ! with RMS error code
;  396          1229  2                   .rms_stv);              ! and secondary code
;  397          1230  2
;  398          1231  2
;  399          1232  2 return .rms_sts;                          ! Pass on the status
;  400          1233  2
;  401          1234  1 END;


                                                 003C 00000          .ENTRY  LOG_FILENAME, Save R2,R3,R4,R5          ; 1114
                                          5E         08   C2 00002          SUBL2   #8, SP                          ;
                                          50    04   AC   D0 00005          MOVL    RMS, R0                         ; 1163
                                                     52   D4 00009          CLRL    R2                              ;
                                          01         60   91 0000B          CMPB    (R0), #1                        ;
                                                     08   12 0000E          BNEQ    1$                              ;
                                                     52   D6 00010          INCL    R2                              ;
                                          51    3C   A0   D0 00012          MOVL    60(R0), FAB                     ; 1165
                                                     03   11 00016          BRB     2$                              ; 1166
                                          51         50   D0 00018 1$:      MOVL    R0, FAB                         ; 1171
                                          53    08   A0   D0 0001B 2$:      MOVL    8(R0), RMS_STS                  ; 1172
                                          55    0C   A0   D0 0001F          MOVL    12(R0), RMS_STV                 ; 1173
                                          54    18   A0   D0 00023          MOVL    24(R0), RMS_CTX                 ; 1174
                                          50    28   A1   D0 00027          MOVL    40(FAB), NAM                    ; 1177
                                          1A         52   E9 0002B          BLBC    R2, 3$                          ; 1186
                                0001827A  8F         53   D1 0002E          CMPL    RMS_STS, #98938                 ; 1187
                                                     11   12 00035          BNEQ    3$                              ;
                                009F10B2  8F         54   D1 00037          CMPL    RMS_CTX, #10424498              ; 1188
                                                     08   12 0003E          BNEQ    3$                              ;
                                          50 0001827A  8F   D0 00040          MOVL    #98938, R0                      ; 1189
                                                          04 00047          RET                                     ;
                                                03   A0   95 00048 3$:      TSTB    3(NAM)                          ; 1201
                                                     0B   13 0004B          BEQL    4$                              ;
                                          6E         03   A0   9A 0004D          MOVZBL  3(NAM), DESC                    ; 1203
                                             04      AE   04   A0   D0 00051          MOVL    4(NAM), DESC+4          ; 1204
                                                     19   11 00056          BRB     6$                              ; 1201
                                                0B   A0   95 00058 4$:      TSTB    11(NAM)                         ; 1207
                                                     0B   13 0005B          BEQL    5$                              ;
                                          6E         0B   A0   9A 0005D          MOVZBL  11(NAM), DESC                   ; 1209
                                             04      AE   0C   A0   D0 00061          MOVL    12(NAM), DESC+4         ; 1210
                                                     09   11 00066          BRB     6$                              ; 1207
                                          6E         34   A1   9A 00068 5$:      MOVZBL  52(FAB), DESC                   ; 1214
                                             04      AE   2C   A1   D0 0006C          MOVL    44(FAB), DESC+4         ; 1215
                                                     28   BB 00071 6$:      PUSHR   #^M<R3,R5>                      ; 1228
                                                0B   AE   9F 00073          PUSHAB  DESC                            ; 1227
                                                     01   DD 00076          PUSHL   #1                              ;
                                                     54   DD 00078          PUSHL   RMS_CTX                         ;
                                00000000G   00         05   FB 0007A          CALLS   #5, LIB$SIGNAL                  ;
                                          50         53   D0 00081          MOVL    RMS_STS, R0                     ; 1232
                                                          04 00084          RET                                     ; 1234

; Routine Size:  133 bytes,    Routine Base:  CODE + 00D3
```

```
;  403     1235  1 UNDECLARE MAP_QUALIFIERS;
;  404     1236  1
;  405     1237  1 GLOBAL ROUTINE MAP_QUALIFIERS  =
;  406     1238  1
;  407     1239  1 !----
;  408     1240  1 !
;  409     1241  1 ! Functional description
;  410     1242  1 !
;  411     1243  1 !       This routine searches all possible qualifiers and makes
;  412     1244  1 !       a bit mask of those that are present
;  413     1245  1 !
;  414     1246  1 ! Input parameters
;  415     1247  1 !
;  416     1248  1 !       None
;  417     1249  1 !
;  418     1250  1 ! Output parameters
;  419     1251  1 !
;  420     1252  1 !       The mask is established
;  421     1253  1 !       Always returns success
;  422     1254  1 !
;  423     1255  1 !----
;  424     1256  1
;  425     1257  2 BEGIN
;  426     1258  2
;  427     1259  2 GLOBAL
;  428     1260  2         qualifiers: bitvector [64];             ! global storage for mask
;  429     1261  2
;  430     1262  2 OWN
;  431     1263  2         qual_array: vector [qual_count * 2] initial (
;  432     1264  2                         primdesc (ACCOUNT),
;  433     1265  2                         primdesc (ACCOUNT),
;  434     1266  2 !                       primdesc (BAR_GRAPH),
;  435     1267  2                         primdesc (BEFORE),
;  436     1268  2                         primdesc (BINARY),
;  437     1269  2                         primdesc (USER),
;  438     1270  2                         primdesc (ENTRY),
;  439     1271  2                         primdesc (FULL),
;  440     1272  2                         primdesc (IDENT),
;  441     1273  2                         primdesc (IMAGE),
;  442     1274  2                         primdesc (JOB),
;  443     1275  2                         primdesc (LOG),
;  444     1276  2                         primdesc (ADDRESS),
;  445     1277  2                         primdesc (NODE),
;  446     1278  2                         primdesc (OWNER),
;  447     1279  2                         primdesc (OUTPUT),
;  448     1280  2                         primdesc (PRIORITY),
;  449     1281  2                         primdesc (PROCESS),
;  450     1282  2                         primdesc (QUEUE),
;  451     1283  2                         primdesc (REJECTED),
;  452     1284  2                         primdesc (REMOTE_ID),
;  453     1285  2                         primdesc (REPORT),
;  454     1286  2                         primdesc (SINCE),
;  455     1287  2                         primdesc (STATUS),
;  456     1288  2                         primdesc (SORT),
;  457     1289  2                         primdesc (SUMMARY),
;  458     1290  2                         primdesc (TERMINAL),
;  459     1291  2                         primdesc (TITLE),
```

```
;  460          1292  2                          primdesc (TYPE),
;  461          1293  2                          primdesc (UIC)
;  462          1294  2                          );
;  463          1295  2
;  464          1296  2 !
;  465          1297  2 ! Establish a bit mask of qualifiers that are present
;  466          1298  2 !
;  467          1299  2
;  468          1300  2 Incra i to qual_count - 1 do qualifiers [.i] = get_present (qual_array [.i*2]);
;  469          1301  2 qualifiers [bar_graph] = 0;
;  470          1302  2 return true;
;  471          1303  1 END;


                                                .PSECT  DATA,NOEXE,2

                                00008 QUALIFIERS::
                                                .BLKB   8
                     00000007   00010 QUAL_ARRAY:
                                                .LONG   7
                     00000000'  00014           .ADDRESS P.AAB
                     00000007   00018           .LONG   7
                     00000000'  0001C           .ADDRESS P.AAC
                     00000006   00020           .LONG   6
                     00000000'  00024           .ADDRESS P.AAD
                     00000006   00028           .LONG   6
                     00000000'  0002C           .ADDRESS P.AAE
                     00000004   00030           .LONG   4
                     00000000'  00034           .ADDRESS P.AAF
                     00000005   00038           .LONG   5
                     00000000'  0003C           .ADDRESS P.AAG
                     00000004   00040           .LONG   4
                     00000000'  00044           .ADDRESS P.AAH
                     00000005   00048           .LONG   5
                     00000000'  0004C           .ADDRESS P.AAI
                     00000005   00050           .LONG   5
                     00000000'  00054           .ADDRESS P.AAJ
                     00000003   00058           .LONG   3
                     00000000'  0005C           .ADDRESS P.AAK
                     00000003   00060           .LONG   3
                     00000000'  00064           .ADDRESS P.AAL
                     00000007   00068           .LONG   7
                     00000000'  0006C           .ADDRESS P.AAM
                     00000004   00070           .LONG   4
                     00000000'  00074           .ADDRESS P.AAN
                     00000005   00078           .LONG   5
                     00000000'  0007C           .ADDRESS P.AAO
                     00000006   00080           .LONG   6
                     00000000'  00084           .ADDRESS P.AAP
                     00000008   00088           .LONG   8
                     00000000'  0008C           .ADDRESS P.AAQ
                     00000007   00090           .LONG   7
                     00000000'  00094           .ADDRESS P.AAR
                     00000005   00098           .LONG   5
                     00000000'  0009C           .ADDRESS P.AAS
                     00000008   000A0           .LONG   8
```

```
                00000000' 000A4          .ADDRESS P.AAT                                  ;
                00000009  000A8          .LONG   9                                       ;
                00000000' 000AC          .ADDRESS P.AAU                                  ;
                00000006  000B0          .LONG   6                                       ;
                00000000' 000B4          .ADDRESS P.AAV                                  ;
                00000005  000B8          .LONG   5                                       ;
                00000000' 000BC          .ADDRESS P.AAW                                  ;
                00000006  000C0          .LONG   6                                       ;
                00000000' 000C4          .ADDRESS P.AAX                                  ;
                00000004  000C8          .LONG   4                                       ;
                00000000' 000CC          .ADDRESS P.AAY                                  ;
                00000007  000D0          .LONG   7                                       ;
                00000000' 000D4          .ADDRESS P.AAZ                                  ;
                00000008  000D8          .LONG   8                                       ;
                00000000' 000DC          .ADDRESS P.ABA                                  ;
                00000005  000E0          .LONG   5                                       ;
                00000000' 000E4          .ADDRESS P.ABB                                  ;
                00000004  000E8          .LONG   4                                       ;
                00000000' 000EC          .ADDRESS P.ABC                                  ;
                00000003  000F0          .LONG   3                                       ;
                00000000' 000F4          .ADDRESS P.ABD                                  ;

                                         .PSECT  CODE,NOWRT,2

00  54  4E  55  4F  43  43  41  00158 P.AAB:  .ASCII  \ACCOUNT\<0>                        ;
00  54  4E  55  4F  43  43  41  00160 P.AAC:  .ASCII  \ACCOUNT\<0>                        ;
00  00  45  52  4F  46  45  42  00168 P.AAD:  .ASCII  \BEFORE\<0><0>                     ;
00  00  59  52  41  4E  49  42  00170 P.AAE:  .ASCII  \BINARY\<0><0>                     ;
                52  45  53  55  00178 P.AAF:  .ASCII  \USER\                           ;
00  00  00  59  52  54  4E  45  0017C P.AAG:  .ASCII  \ENTRY\<0><0><0>                  ;
                4C  4C  55  46  00184 P.AAH:  .ASCII  \FULL\                           ;
00  00  00  54  4E  45  44  49  00188 P.AAI:  .ASCII  \IDENT\<0><0><0>                  ;
00  00  00  45  47  41  4D  49  00190 P.AAJ:  .ASCII  \IMAGE\<0><0><0>                  ;
                00  42  4F  4A  00198 P.AAK:  .ASCII  \JOB\<0>                        ;
                00  47  4F  4C  0019C P.AAL:  .ASCII  \LOG\<0>                        ;
00  53  53  45  52  44  44  41  001A0 P.AAM:  .ASCII  \ADDRESS\<0>                    ;
                45  44  4F  4E  001A8 P.AAN:  .ASCII  \NODE\                          ;
00  00  00  52  45  4E  57  4F  001AC P.AAO:  .ASCII  \OWNER\<0><0><0>                ;
00  00  54  55  50  54  55  4F  001B4 P.AAP:  .ASCII  \OUTPUT\<0><0>                  ;
59  54  49  52  4F  49  52  50  001BC P.AAQ:  .ASCII  \PRIORITY\                      ;
00  53  53  45  43  4F  52  50  001C4 P.AAR:  .ASCII  \PROCESS\<0>                   ;
00  00  00  45  55  45  55  51  001CC P.AAS:  .ASCII  \QUEUE\<0><0><0>               ;
44  45  54  43  45  4A  45  52  001D4 P.AAT:  .ASCII  \REJECTED\                     ;
00  00  00  44  49  5F  45  54  4F  4D  45  52  001DC P.AAU:  .ASCII  \REMOTE_ID\<0><0><0>  ;
00  00  54  52  4F  50  45  52  001E8 P.AAV:  .ASCII  \REPORT\<0><0>                 ;
00  00  00  45  43  4E  49  53  001F0 P.AAW:  .ASCII  \SINCE\<0><0><0>               ;
00  00  53  55  54  41  54  53  001F8 P.AAX:  .ASCII  \STATUS\<0><0>                ;
                54  52  4F  53  00200 P.AAY:  .ASCII  \SORT\                          ;
00  59  52  41  4D  4D  55  53  00204 P.AAZ:  .ASCII  \SUMMARY\<0>                   ;
4C  41  4E  49  4D  52  45  54  0020C P.ABA:  .ASCII  \TERMINAL\                     ;
00  00  00  45  4C  54  49  54  00214 P.ABB:  .ASCII  \TITLE\<0><0><0>               ;
                45  50  59  54  0021C P.ABC:  .ASCII  \TYPE\                         ;
                00  43  49  55  00220 P.ABD:  .ASCII  \UIC\<0>                       ;

                                         .EXTRN  CLI$PRESENT

                                000C 00000       .ENTRY  MAP_QUALIFIERS, Save R2,R3     ; 1237
```

```
                                53 00000000'   EF  9E 00002          MOVAB   QUALIFIERS, R3                                  ;
                                               52  D4 00009          CLRL    I                                               ; 1300
                          50                   52  01 78 0000B 1$:   ASHL    #1, I, R0                                       ;
                                            08 A340  DF 0000F          PUSHAL  QUAL_ARRAY[R0]                                  ;
                             00000000G          00  FB 00013          CALLS   #1, CLI$PRESENT                                 ;
                    63       01                 52  50 F0 0001A      INSV    R0, I, #1, QUALIFIERS                           ;
                                                52  D6 0001F          INCL    I                                               ;
                                            1C  52  D1 00021          CMPL    I, #28                                          ;
                                                E5  1B 00024          BLEQU   1$                                              ;
                                            63  02  8A 00026          BICB2   #2, QUALIFIERS                                  ; 1301
                                            50  01  D0 00029          MOVL    #1, R0                                          ; 1302
                                                    04 0002C          RET                                                     ; 1303

; Routine Size:  45 bytes,    Routine Base:  CODE + 0224
```

```
;  473     1304  1 UNDECLARE HANDLER;
;  474     1305  1
;  475     1306  1 GLOBAL ROUTINE HANDLER (SIGNAL_ARGS, MECHANISM_ARGS) =
;  476     1307  1
;  477     1308  1 !---
;  478     1309  1 !
;  479     1310  1 !       This condition handler gets control on any signalled
;  480     1311  1 !       condition in order to save the highest severity error
;  481     1312  1 !       to be returned by exit from the image.
;  482     1313  1 !
;  483     1314  1 ! Inputs:
;  484     1315  1 !
;  485     1316  1 !       signal_args = Address of signal argument list
;  486     1317  1 !       mechanism_args = Address of mechanism argument list
;  487     1318  1 !
;  488     1319  1 ! Outputs:
;  489     1320  1 !
;  490     1321  1 !       WORST_ERROR is updated with highest severity error.
;  491     1322  1 !
;  492     1323  1 !---
;  493     1324  1
;  494     1325  2 BEGIN
;  495     1326  2
;  496     1327  2 GLOBAL worst_error: initial (1);        ! Holds worst error encountered
;  497     1328  2
;  498     1329  2 MAP
;  499     1330  2     signal_args:        ref bblock,     ! Address of signal argument list
;  500     1331  2     mechanism_args:     ref bblock;     ! Address of mechanism argument list
;  501     1332  2
;  502     1333  2 LOCAL
;  503     1334  2     cond_code:          long;           ! Condition code (longword)
;  504     1335  2
;  505     1336  2 cond_code = .signal_args [chf$l_sig_name];      ! Get condition code
;  506     1337  2 If .cond_code eql rms$_eof then return true;
;  507     1338  2 If severity_level (.cond_code) gtr
;  508     1339  3    severity_level (.worst_error)              ! If higher than watermark
;  509     1340  2 then worst_error = .cond_code or sts$m_inhib_msg; !  -then set new worst error
;  510     1341  2
;  511     1342  2 ss$_resignal                                ! Continue signalling
;  512     1343  2
;  513     1344  1 END;


                                                .PSECT  DATA,NOEXE,2

                                00000001  000F8 WORST_ERROR::
                                                .LONG   1                                       ;


                                                .PSECT  CODE,NOWRT,2

                                    001C 00000          .ENTRY  HANDLER, Save R2,R3,R4          ; 1306
                    54 00000000'    EF   9E 00002       MOVAB   WORST_ERROR, R4                 ;
                    50          04  AC   D0 00009       MOVL    SIGNAL_ARGS, R0                 ; 1336
                    53          04  A0   D0 0000D       MOVL    4(R0), COND_CODE                ;
```

```
                          0001827A  8F              53  D1 00011          CMPL    COND_CODE, #98938                                   ; 1337
                                                    04  12 00018          BNEQ    1$                                                  ;
                                    50              01  D0 0001A          MOVL    #1, R0                                              ;
                                                    04  0001D            RET                                                         ;
                                    50              53  D0 0001E 1$:      MOVL    COND_CODE, CODE                                     ; 1338
              51            50      03              00  EF 00021          EXTZV   #0, #3, CODE, R1                                    ;
              50            50      01              00  EF 00026          EXTZV   #0, #1, CODE, R0                                    ;
                                    50              04  C4 0002B          MULL2   #4, R0                                              ;
                                    51              50  C2 0002E          SUBL2   R0, R1                                              ;
                                    51              03  C0 00031          ADDL2   #3, R1                                              ;
                                    50              64  D0 00034          MOVL    WORST_ERROR, CODE                                   ; 1339
              52            50      02              00  EF 00037          EXTZV   #0, #3, CODE, R2                                    ;
              50            50      01              00  EF 0003C          EXTZV   #0, #1, CODE, R0                                    ;
                                    50              04  C4 00041          MULL2   #4, R0                                              ;
                                    52              50  C2 00044          SUBL2   R0, R2                                              ;
                                    50        03    A2  9E 00047          MOVAB   3(R2), R0                                           ;
                                    50              51  D1 0004B          CMPL    R1, R0                                              ;
                                                    08  15 0004E          BLEQ    2$                                                  ;
                          64                53 10000000 8F  C9 00050          BISL3   #268435456, COND_CODE, WORST_ERROR                  ; 1340
                                    50      0918    8F  3C 00058 2$:      MOVZWL  #2328, R0                                           ; 1344
                                                    04  0005D            RET                                                         ;

; Routine Size:  94 bytes,    Routine Base:  CODE + 0251
```

```
;  515          1345  1 END
;  516          1346  0 ELUDOM


                                                        .EXTRN  LIB$SIGNAL

;                               PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  CODE                          687  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  DATA                          252  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)


;                           Library Statistics
;
;                                       -------- Symbols --------      Pages       Processing
;       File                            Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]STARLET.L32;1     9776        57         0         581        00:01.0
;  _$255$DUA28:[SYSLIB]TPAMAC.L32;1        42         0         0          14        00:00.1


;                                   COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:UTILITY/OBJ=OBJ$:UTILITY MSRC$:UTILITY/UPDATE=(ENH$:UTILITY)

; Size:          470 code + 469 data bytes
; Run Time:          00:19.5
; Elapsed Time:      00:46.6
; Lines/CPU Min:      4135
; Lexemes/CPU-Min: 22525
; Memory Used:  134 pages
; Compilation Complete
```

AEDCLENUP
LIS

SYMBOLS
LIS

UTILITY
LIS

ACLEDTDEF
REQ

ACLEDT

SETSHOACL
MAP

ACLEDT
MAP

AEDLCLDEE
SDL

SUMMARY
LIS

SYMBOLMSG
LIS

AEDDECODE
LIS