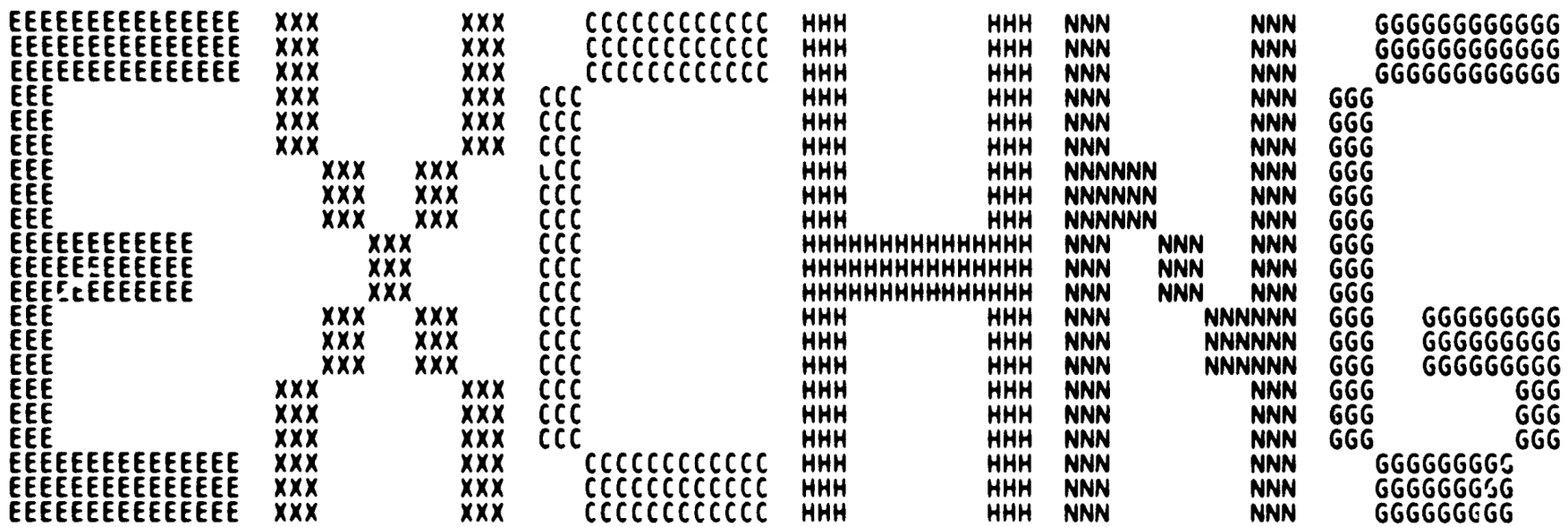
EEEEEEEEEEEEEEE XXX XXX CCCCCCCCCCCC HHH HHH NNN NNN GGGGGGGGGGG
EEE XXX XXX CCC HHH HHH NNN NNN GGG
EEE XXX XXX CCC HHH HHH NNN NNN GGG
EEE XXX XXX CCC HHH HHH NNNNNN NNN GGG
EEE XXX XXX CCC HHH HHH NNNNNN NNN GGG
EEE XXX XXX CCC HHH HHH NNNNNN NNN GGG
EEEEEEEEEEEE XXX CCC HHHHHHHHHHHHHHH NNN NNN NNN GGG
EEEEEEEEEEEE XXX CCC HHHHHHHHHHHHHHH NNN NNN NNN GGG
EEEEEEEEEEEE XXX CCC HHHHHHHHHHHHHHH NNN NNN NNN GGG
EEE XXX XXX CCC HHH HHH NNN NNNNNN GGG GGGGGGGG
EEE XXX XXX CCC HHH HHH NNN NNNNNN GGG GGGGGGGG
EEE XXX XXX CCC HHH HHH NNN NNNNNN GGG GGGGGGGG
EEE XXX XXX CCC HHH HHH NNN NNN GGG GGG
EEE XXX XXX CCC HHH HHH NNN NNN GGG GGG
EEEEEEEEEEEEEEE XXX XXX CCCCCCCCCCCC HHH HHH NNN NNN GGGGGGGGG
EEEEEEEEEEEEEEE XXX XXX CCCCCCCCCCCC HHH HHH NNN NNN GGGGGGGGG
EEEEEEEEEEEEEEE XXX XXX CCCCCCCCCCCC HHH HHH NNN NNN GGGGGGGGG

FILEID**EXCLIB

```
EEEEEEEEEE  XX      XX    CCCCCCCC  LL          IIIIII    BBBBBBBB
EEEEEEEEEE  XX      XX    CCCCCCCC  LL          IIIIII    BBBBBBBB
EE          XX      XX  CC          LL            II      BB      BB
EE          XX      XX  CC          LL            II      BB      BB
EE            XX  XX    CC          LL            II      BB      BB
EE            XX  XX    CC          LL            II      BB      BB
EEEEEEEE        XX      CC          LL            II      BBBBBBBB
EEEEEEEE        XX      CC          LL            II      BBBBBBBB
EE            XX  XX    CC          LL            II      BB      BB
EE            XX  XX    CC          LL            II      BB      BB
EE          XX      XX  CC          LL            II      BB      BB      ....
EE          XX      XX  CC          LL            II      BB      BB      ....
EEEEEEEEEE  XX      XX    CCCCCCCC  LLLLLLLLLL    IIIIII    BBBBBBBB      ....
EEEEEEEEEE  XX      XX    CCCCCCCC  LLLLLLLLLL    IIIIII    BBBBBBBB      ...

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
0001  0
0002  0     !MODULE exch$library    (IDENT = 'V04-000') =   %TITLE 'Facility-wide library module'
0003  0     ! BEGIN
0004  0     !
0005  0     !*****************************************************************************
0006  0     !*                                                                           *
0007  0     !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0008  0     !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0009  0     !*  ALL RIGHTS RESERVED.                                                     *
0010  0     !*                                                                           *
0011  0     !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0012  0     !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0013  0     !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0014  0     !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0015  0     !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0016  0     !*  TRANSFERRED.                                                             *
0017  0     !*                                                                           *
0018  0     !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0019  0     !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL FQUIPMENT    *
0020  0     !*  CORPORATION.                                                             *
0021  0     !*                                                                           *
0022  0     !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIAEILITY OF ITS    *
0023  0     !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0024  0     !*                                                                           *
0025  0     !*                                                                           *
0026  0     !*****************************************************************************
0027  0     !
0028  0     !++
0029  0     ! FACILITY:     EXCHANGE - Foreign volume interchange facility
0030  0     !
0031  0     ! ABSTRACT:     BLISS Library for EXCHANGE facility
0032  0     !
0033  0     ! ENVIRONMENT:  VAX/VMS User mode
0034  0     !
0035  0     ! AUTHOR:       CW Hobbs        , CREATION DATE: 1-July-1982
0036  0     !
0037  0     ! MODIFIED BY:
0038  0     !
0039  0     !       V03-002 CWH3002                 CW Hobbs                12-Apr-1984
0040  0     !               Add NOREMOTE, NOTSAMEDEV and RT11_DIRSIZE message codes.
0041  0     !
0042  0     !
0043  0     !--
```

```
;  0044  0     ! Include files:
;  0045  0     !
;  0046  0
;  0047  0     ! LIBRARY files:
;  0048  0     !
;  0049  0     LIBRARY
;  0050  0         'SYS$LIBRARY:LIB'           ! VMS operating system library
;  0051  0         ;
;  0052  0
;  0053  0     ! REQUIRE files:
;  0054  0     !
;  0055  0     REQUIRE
;  0056  0         'LIB$:EXCDEFS'              ! include the SDL definitions
;  0057  0         ;
;  1687  0
;  1688  0     ! Macros:
;  1689  0     !
;  1690  0
;  1691  0     ! Declare some macros as shorthand for the psect names
;  1692  0     !
;  1693  0     MACRO
;  1694  0             $global_rw = PSECT GLOBAL  = exch$rw_global (ADDRESSING_MODE (LONG_RELATIVE)); GLOBAL %
;  1695  0             ;
```

```
;    1696  0      ! Declare some common data structure initialization macros
;    1697  0      !
;    1698  0      MACRO
;    1699  0              ! Define shorthand for a single initialized dynamic string desc
;    1700  0              !
;    1701  0              $dyn_str_desc                           ! Static declaration
;  M 1702  0                      =
;  M 1703  0                      BLOCK [dsc$k_d_bln,BYTE]
;  M 1704  0                      PRESET ([dsc$b_class]   = dsc$k_class_d,
;  M 1705  0                              [dsc$b_dtype]   = dsc$k_dtype_t,
;  M 1706  0                              [dsc$w_length]  = 0,
;  M 1707  0                              [dsc$a_pointer] = 0 )
;    1708  0                      %,
;    1709  0
;    1710  0              $dyn_str_desc_init (desci)              ! Run-time initialization
;  M 1711  0                      =
;  M 1712  0                      BEGIN
;  M 1713  0                      BIND
;  M 1714  0                          desc = (desci) : VECTOR [2, LONG],
;  M 1715  0                          tmpl = exch$gg_dyn_str_template : VECTOR [2, LONG];
;  M 1716  0                      desc [0] = .tmpl [0];
;  M 1717  0                      desc [1] = .tmpl [1];
;  M 1718  0                      END
;    1719  0                      %,
;    1720  0
;    1721  0              ! Define macro for a single initialized static string desc.
;    1722  0              !
;    1723  0              $stat_str_desc (L, A)                   ! Static declaration
;  M 1724  0                      =
;  M 1725  0                      BLOCK [dsc$k_s_bln,BYTE]
;  M 1726  0                      PRESET( [dsc$b_class]   = dsc$k_class_s,
;  M 1727  0                              [dsc$b_dtype]   = dsc$k_dtype_t,
;  M 1728  0                              [dsc$w_length]  = (L),
;  M 1729  0                              [dsc$a_pointer] = (A) )
;    1730  0                      %,
;    1731  0
;    1732  0              $stat_str_desc_init (desci, L, A)       ! Run-time initialization
;  M 1733  0                      =
;  M 1734  0                      BEGIN
;  M 1735  0                      BIND
;  M 1736  0                          desc = (desci) : BLOCK [, BYTE];
;  M 1737  0                      desc [dsc$b_class]   = dsc$k_class_s;
;  M 1738  0                      desc [dsc$b_dtype]   = dsc$k_dtype_t;
;  M 1739  0                      desc [dsc$w_length]  = (L);
;  M 1740  0                      desc [dsc$a_pointer] = (A);
;  M 1741  0                      END
;    1742  0                      %,
;    1743  0
;    1744  0              $str_desc_set (desci, L, A)     ! Copy new length and pointer fields (both static and dynamic)
;  M 1745  0                      =
;  M 1746  0                      BEGIN
;  M 1747  0                      BIND
;  M 1748  0                          desc = (desci) : BLOCK [, BYTE];
;  M 1749  0                      desc [dsc$w_length]  = (L);
;  M 1750  0                      desc [dsc$a_pointer] = (A);
;  M 1751  0                      END
;    1752  0                      %,
```

```
;  1753  0
;  1754  0              ! And shorthand for just a descriptor declaration
;  1755  0              !
;  1756  0              $desc_block
;M 1757  0                              =
;M 1758  0                              BLOCK [dsc$k_s_bln, BYTE]
;  1759  0                              %,
;  1760  0
;  1761  0              ! Short form for byte vector reference
;  1762  0              !
;  1763  0              $ref_bvector
;M 1764  0                              =
;M 1765  0                              REF $bvector
;  1766  0                              %,
;  1767  0
;  1768  0              ! Short form for byte block reference
;  1769  0              !
;  1770  0              $ref_bblock
;M 1771  0                              =
;M 1772  0                              REF $bblock
;  1773  0                              %;
;  1774  0
;  1775  0              STRUCTURE
;  1776  0                              $bvector [I; N] =
;  1777  0                                              [N]
;  1778  0                                              ($bvector+I)<0,8,0>;
;  1779  0
;  1780  0      !+
;  1781  0      ! SIGNAL_STOP a condition assuming no return.  LIB$exch_signal_STOP is not
;  1782  0      ! supposed to return, but BLISS doesn't know this, so we block further
;  1783  0      ! flow here.  This will generate better code for us.
;  1784  0      !-
;  1785  0      MACRO
;  1786  0          $exch_signal_stop []
;M 1787  0                                  =
;M 1788  0                                  BEGIN
;M 1789  0                                  LINKAGE
;M 1790  0                                      LNK = CALL : PRESERVE (0,1,2,3,4,5,6,7,8,9,10,11);
;M 1791  0                                  EXTERNAL ROUTINE
;M 1792  0                                      LIB$STOP : ADDRESSING_MODE (GENERAL) LNK NOVALUE;
;M 1793  0                                  BUILTIN
;M 1794  0                                      R0;
;M 1795  0
;M 1796  0                                  LIB$STOP (%REMAINING);
;M 1797  0                                  RETURN (.R0);
;M 1798  0
;M 1799  0                                  END
;  1800  0                                  %;
;  1801  0
;  1802  0      !+
;  1803  0      ! SIGNAL a condition and return.
;  1804  0      !-
;  1805  0      MACRO
;  1806  0          $exch_signal_return (code)
;M 1807  0                                  =
;M 1808  0                                  BEGIN
;M 1809  0                                  LOCAL
```

```
;  M 1810  0                                temp;
;  M 1811  0
;  M 1812  0                            temp = (code);          ! Need to avoid multiple calls, etc
;  M 1813  0                            SIGNAL (.temp           ! Perform the actual signal of the error
;  M 1814  0                                    %IF %LENGTH GTR 1 %THEN ,%REMAINING %FI);
;  M 1815  0                            RETURN .temp
;  M 1816  0
;  M 1817  0                            END
;     1818  0                            %;
;     1819  0
;     1820  0      !+
;     1821  0      ! SIGNAL a condition and continue.
;     1822  0      !-
;     1823  0      MACRO
;     1824  0          $exch_signal (code)
;  M 1825  0                            =
;  M 1826  0                            SIGNAL ( (code)         ! Perform the actual signal of the error
;  M 1827  0                                    %IF %LENGTH GTR 1 %THEN ,%REMAINING %FI)
;     1828  0                            %;
;     1829  0
```

```
;    1830  0      !+
;    1831  0      ! Initialize a control block type and size fields.  We do not depend on them being in the standard positions
;    1832  0      !-
;    1833  0      MACRO
;    1834  0          $block_init (addr, prefix)
; M  1835  0                              =
; M  1836  0                              BEGIN
; M  1837  0                              BIND
; M  1838  0                                  addr2 = (addr) : BLOCK [, BYTE];
; M  1839  0                              addr2 [%NAME (prefix,'$w_size')] = %NAME ('exchblk$s_',prefix);
; M  1840  0                              addr2 [%NAME (prefix,'$b_type')] = %NAME ('exchblk$k_',prefix);
; M  1841  0                              END
;    1842  0                              %;
;    1843  0
;    1844  0      !+
;    1845  0      ! Check a control block type and size fields.  Note that we depend on them being in the standard positions
;    1846  0      !-
;    1847  0      MACRO
;    1848  0          $block_check (level, addr, prefix, error_code)
; M  1849  0                              =
; M  1850  0                              %IF switch_variant GEQ (level)
; M  1851  0                              %THEN
; M  1852  0                                      BEGIN
; M  1853  0                                      EXTERNAL ROUTINE
; M  1854  0                                          exch$util_block_check : jsb_r0r1r2 NOVALUE;
; M  1855  0
; M  1856  0                                      exch$util_block_check ( (addr), (error_code),
; M  1857  0                                                  (%NAME ('exchblk$s_',prefix) ^ 16  OR %NAME ('exchblk$k_',prefix)));
; M  1858  0
; M  1859  0                                      END
; M  1860  0                              %FI
;    1861  0                              %;
;    1862  0
;    1863  0      MACRO
;    1864  0          $block_check_if_nonzero (level, addr, prefix, error_code)
; M  1865  0                              =
; M  1866  0                              %IF switch_variant GEQ (level)
; M  1867  0                              %THEN
; M  1868  0                                      BEGIN
; M  1869  0                                      BIND
; M  1870  0                                          addr2 = (addr) : BLOCK [, BYTE];
; M  1871  0                                      IF addr2 NEQ 0
; M  1872  0                                      THEN
; M  1873  0                                          $block_check ((level), (addr), prefix, (error_code));
; M  1874  0                                      END
; M  1875  0                              %FI
;    1876  0                              %;
;    1877  0
;    1878  0      !+
;    1879  0      ! Check for a logic error.  If the expression is not true, then we have a problem.
;    1880  0      !-
;    1881  0      MACRO
;    1882  0          $logic_check (level, condition, error_code)
; M  1883  0                              =
; M  1884  0
; M  1885  0                              ! See if a compile time check is possible
; M  1886  0                              !
```

```
;  M 1887  0                %IF %CTCE ((condition))
;  M 1888  0                %THEN
;  M 1889  0
;  M 1890  0                    ! The condition is a compile-time expression.  There is one special case, when the
;  M 1891  0                    ! condition is the string "(false)".  This is used as an unconditional logic abort.
;  M 1892  0                    ! If we have "(false)", then do a naked SIGNAL_STOP
;  M 1893  0                    !
;  M 1894  0                    %IF %IDENTICAL (condition, (false))
;  M 1895  0                    %THEN
;  M 1896  0                        SIGNAL_STOP (exch$_badlogic, 1, (error_code))
;  M 1897  0
;  M 1898  0                    ! The condition is a normal test.  If it is true, print a message that the condition
;  M 1899  0                    ! was verified during compilation.  If false, generate a serious error.
;  M 1900  0                    !
;  M 1901  0                    %ELSE
;  M 1902  0                        %IF (condition)
;  M 1903  0                        %THEN
;  M 1904  0                            %PRINT ('assumption ',error_code,' verified during compilation')
;  M 1905  0                        %ELSE
;  M 1906  0                            %ERROR ('assumption ',error_code,' is not true')
;  M 1907  0                        %FI
;  M 1908  0                    %FI
;  M 1909  0
;  M 1910  0                ! The condition is not a compile-time constant.  If the current variant calls for it,
;  M 1911  0                ! generate run-time code to test the assumption.
;  M 1912  0                !
;  M 1913  0                %ELSE
;  M 1914  0                    %IF switch_variant GEQ (level)
;  M 1915  0                    %THEN
;  M 1916  0                            BEGIN
;  M 1917  0                            IF NOT (condition)
;  M 1918  0                            THEN
;  M 1919  0                                SIGNAL_STOP (exch$_badlogic, 1, (error_code));
;  M 1920  0                            END
;  M 1921  0                    %FI
;  M 1922  0                %FI
;     1923  0                %;
```

```
;  1924  0      !+
;  1925  0      ! Most messages are defined as warnings so that we can signal without changing flow of execution.  Several
;  1926  0      ! macros are defined in EXCLIB to change the severity code.  These are $warning_stat, $success_stat,
;  1927  0      ! $error_stat, $info_stat and $severe_stat.  A typical use would be:
;  1928  0      !
;  1929  0      !               status = lib$foo (bar);
;  1930  0      !               IF NOT .status
;  1931  0      !               THEN
;  1932  0      !                   BEGIN
;  1933  0      !                   $warning_stat (status);     ! Convert unknown severity to warning
;  1934  0      !                   SIGNAL (.status);
;  1935  0      !                   RETURN .status;
;  1936  0      !                   END;
;  1937  0      !
;  1938  0      ! These macros modify the status variable and return the value of the modified status.
;  1939  0      !
;  1940  0      ! For those situations where it is inappropriate to modify the code, forms of the macro are available which
;  1941  0      ! modify copies of the status code.  These macro names have "_copy" appended to the modify form of the macro
;  1942  0      ! name.  A couple of examples of their use are:
;  1943  0      !
;  1944  0      !       status = lib$foo (bar);                             status = lib$foo (bar);
;  1945  0      !       IF NOT .status                                      IF NOT .status
;  1946  0      !       THEN                                                THEN
;  1947  0      !           BEGIN                                               BEGIN
;  1948  0      !           new_stat = $warning_status_copy (.status);          SIGNAL ($error_status_copy (exch$_badfoo), 0, .status);
;  1949  0      !           SIGNAL (.new_stat);                                 RETURN (exch$_badfoo);
;  1950  0      !           RETURN (.status);                                   END;
;  1951  0      !           END;
;  1952  0      !
;  1953  0      ! Note that the "_copy" forms have value arguments, the regular forms have address arguments.
;  1954  0      !-
;  1955  0      !
;  1956  0      ! Convert status codes to specific status values.
;  1957  0      !-
;  1958  0      MACRO
;  1959  0          $inhibit_msg (status)
;M 1960  0                               =
;M 1961  0                               BEGIN
;M 1962  0                               BIND
;M 1963  0                                   status2 = status : BLOCK [4, BYTE];
;M 1964  0                               status2 [sts$v_inhib_msg] = 1;          ! Inhibit $EXIT signalling
;M 1965  0                               .status2                                ! Value of block is new code
;M 1966  0                               END
;  1967  0                               %,
;  1968  0
;  1969  0          $inhibit_msg_copy (status)
;M 1970  0                               =
;M 1971  0                               BEGIN
;M 1972  0                               LOCAL
;M 1973  0                                   status2 : BLOCK [4, BYTE];
;M 1974  0                               status2 [0,0,32,0] = status;            ! Copy the whole code
;M 1975  0                               status2 [sts$v_inhib_msg] = 1;          ! Inhibit $EXIT signalling
;M 1976  0                               .status2                                ! Value of block is new code
;M 1977  0                               END
;  1978  0                               %,
;  1979  0
;  1980  0          $warning_stat (status)
```

```
;  M 1981  0                        =
;  M 1982  0                        BEGIN
;  M 1983  0                        BIND
;  M 1984  0                            status2 = status : BLOCK [4, BYTE];
;  M 1985  0                        status2 [sts$v_severity] = sts$k_warning;       ! Force status to warning
;  M 1986  0                        .status2                                        ! Value of block is new code
;  M 1987  0                        END
;    1988  0                        %,
;    1989  0
;    1990  0          $warning_stat_copy (status)
;  M 1991  0                        =
;  M 1992  0                        BEGIN
;  M 1993  0                        LOCAL
;  M 1994  0                            status2 : BLOCK [4, BYTE];
;  M 1995  0                        status2 [0,0,32,0] = status;                    ! Copy the whole code
;  M 1996  0                        status2 [sts$v_severity] = sts$k_warning;       ! Force status to warning
;  M 1997  0                        .status2                                        ! Value of block is new code
;  M 1998  0                        END
;    1999  0                        %,
;    2000  0
;    2001  0          $success_stat (status)
;  M 2002  0                        =
;  M 2003  0                        BEGIN
;  M 2004  0                        BIND
;  M 2005  0                            status2 = status : BLOCK [4, BYTE];
;  M 2006  0                        status2 [sts$v_severity] = sts$k_success;       ! Force status to success
;  M 2007  0                        .status2                                        ! Value of block is new code
;  M 2008  0                        END
;    2009  0                        %,
;    2010  0
;    2011  0          $success_stat_copy (status)
;  M 2012  0                        =
;  M 2013  0                        BEGIN
;  M 2014  0                        LOCAL
;  M 2015  0                            status2 : BLOCK [4, BYTE];
;  M 2016  0                        status2 [0,0,32,0] = status;                    ! Copy the whole code
;  M 2017  0                        status2 [sts$v_severity] = sts$k_success;       ! Force status to success
;  M 2018  0                        .status2                                        ! Value of block is new code
;  M 2019  0                        END
;    2020  0                        %,
;    2021  0
;    2022  0          $error_stat (status)
;  M 2023  0                        =
;  M 2024  0                        BEGIN
;  M 2025  0                        BIND
;  M 2026  0                            status2 = status : BLOCK [4, BYTE];
;  M 2027  0                        status2 [sts$v_severity] = sts$k_error; ! Force status to error
;  M 2028  0                        .status2                                ! Value of block is new code
;  M 2029  0                        END
;    2030  0                        %,
;    2031  0
;    2032  0          $error_stat_copy (status)
;  M 2033  0                        =
;  M 2034  0                        BEGIN
;  M 2035  0                        LOCAL
;  M 2036  0                            status2 : BLOCK [4, BYTE];
;  M 2037  0                        status2 [0,0,32,0] = status;                    ! Copy the whole code
```

```
;  M 2038  0                        status2 [sts$v_severity] = sts$k_error; ! Force status to error
;  M 2039  0                        .status2                                ! Value of block is new code
;  M 2040  0                        END
;    2041  0                        %,
;    2042  0
;    2043  0          $info_stat (status)
;  M 2044  0                        =
;  M 2045  0                        BEGIN
;  M 2046  0                        BIND
;  M 2047  0                            status2 = status : BLOCK [4, BYTE];
;  M 2048  0                        status2 [sts$v_severity] = sts$k_info;  ! Force status to info
;  M 2049  0                        .status2                                ! Value of block is new code
;  M 2050  0                        END
;    2051  0                        %,
;    2052  0
;    2053  0          $info_stat_copy (status)
;  M 2054  0                        =
;  M 2055  0                        BEGIN
;  M 2056  0                        LOCAL
;  M 2057  0                            status2 : BLOCK [4, BYTE];
;  M 2058  0                        status2 [0,0,32,0] = status;            ! Copy the whole code
;  M 2059  0                        status2 [sts$v_severity] = sts$k_info;  ! Force status to info
;  M 2060  0                        .status2                                ! Value of block is new code
;  M 2061  0                        END
;    2062  0                        %,
;    2063  0
;    2064  0          $severe_stat (status)
;  M 2065  0                        =
;  M 2066  0                        BEGIN
;  M 2067  0                        BIND
;  M 2068  0                            status2 = status : BLOCK [4, BYTE];
;  M 2069  0                        status2 [sts$v_severity] = sts$k_severe;        ! Force status to severe
;  M 2070  0                        .status2                                ! Value of block is new code
;  M 2071  0                        END
;    2072  0                        %,
;    2073  0
;    2074  0              $severe_stat_copy (status)
;  M 2075  0                        =
;  M 2076  0                        BEGIN
;  M 2077  0                        LOCAL
;  M 2078  0                            status2 : BLOCK [4, BYTE];
;  M 2079  0                        status2 [0,0,32,0] = status;            ! Copy the whole code
;  M 2080  0                        status2 [sts$v_severity] = sts$k_severe;        ! Force status to severe
;  M 2081  0                        .status2                                ! Value of block is new code
;  M 2082  0                        END
;    2083  0                        %;
;    2084  0
;    2085  0
;    2086  0          ! Special debug and trace macros
;    2087  0          !
;    2088  0          MACRO
;    2089  0              $dbgtrc_prefix (string)                     ! Declare a nested macro with the value of the string
;  M 2090  0                        =
;  M 2091  0                        MACRO   $dbgtrc_prefix_string = string %QUOTE %
;    2092  0                        %,
;    2093  0
;    2094  0              $check_call (level, routine_addr)           ! Call the routine depending on variant level
```

```
;  M 2095  0                    =
;  M 2096  0                    %IF switch_variant GEQ (level)
;  M 2097  0                    %THEN
;  M 2098  0                            BEGIN
;  M 2099  0                            EXTERNAL ROUTINE routine_addr : ADDRESSING_MODE (GENERAL);
;  M 2100  0                            routine_addr (%REMAINING)
;  M 2101  0                            END;
;  M 2102  0                    %FI
;    2103  0                    %;
```

```
;    2104  0      ! Message print routines
;    2105  0      !
;    2106  0      MACRO
;    2107  0          $print_lit (string)
;M   2108  0                              =
;M   2109  0                              lib$put_output (%ASCID string)
;    2110  0                              %,
;    2111  0
;    2112  0          $trace_print_lit (string)
;M   2113  0                              =
;M   2114  0                              %IF switch_trace
;M   2115  0                              %THEN
;M   2116  0                                      lib$put_output (%ASCID %STRING ($dbgtrc_prefix_string, string))
;M   2117  0                              %FI ! switch_trace
;    2118  0                              %,
;    2119  0
;    2120  0          $debug_print_lit (string)
;M   2121  0                              =
;M   2122  0                              %IF switch_debug
;M   2123  0                              %THEN
;M   2124  0                                      lib$put_output (%ASCID %STRING ($dbgtrc_prefix_string, string))
;M   2125  0                              %FI ! switch_debug
;    2126  0                              %,
;    2127  0
;    2128  0          $print_desc (desc)
;M   2129  0                              =
;M   2130  0                              lib$put_output (desc)
;    2131  0                              %,
;    2132  0
;    2133  0          $trace_print_desc (desc)
;M   2134  0                              =
;M   2135  0                              %IF switch_trace
;M   2136  0                              %THEN
;M   2137  0                                      BEGIN
;M   2138  0                                      EXTERNAL ROUTINE  exch$util_fao_buffer;
;M   2139  0                                      lib$put_output (
;M   2140  0                                          exch$util_fao_buffer (%ASCID %STRING ($dbgtrc_prefix_string, '!AS'), desc))
;M   2141  0                                      END
;M   2142  0                              %FI ! switch_trace
;    2143  0                              %,
;    2144  0
;    2145  0          $debug_print_desc (desc)
;M   2146  0                              =
;M   2147  0                              %IF switch_debug
;M   2148  0                              %THEN
;M   2149  0                                      BEGIN
;M   2150  0                                      EXTERNAL ROUTINE  exch$util_fao_buffer;
;M   2151  0                                      lib$put_output (
;M   2152  0                                          exch$util_fao_buffer (%ASCID %STRING ($dbgtrc_prefix_string, '!AS'), desc));
;M   2153  0                                      END
;M   2154  0                              %FI ! switch_debug
;    2155  0                              %,
;    2156  0
;    2157  0          $print_fao (string)
;M   2158  0                              =
;M   2159  0                              BEGIN
;M   2160  0                              EXTERNAL ROUTINE  exch$util_fao_buffer;
```

```
;  M 2161  0                            lib$put_output (
;  M 2162  0                                exch$util_fao_buffer (%ASCID string
;  M 2163  0                                    %IF %LENGTH GTR 1 %THEN ,%REMAINING %FI))
;  M 2164  0                            END
;    2165  0                            %,
;    2166  0
;    2167  0          $trace_print_fao (string)
;  M 2168  0                            =
;  M 2169  0                            %IF switch_trace
;  M 2170  0                            %THEN
;  M 2171  0                                    BEGIN
;  M 2172  0                                    EXTERNAL ROUTINE  exch$util_fao_buffer;
;  M 2173  0                                    lib$put_output (
;  M 2174  0                                        exch$util_fao_buffer (%ASCID %STRING ($dbgtrc_prefix_string, string)
;  M 2175  0                                            %IF %LENGTH GTR 1 %THEN ,%REMAINING %FI))
;  M 2176  0                                    END
;  M 2177  0                            %FI ! switch_trace
;    2178  0                            %,
;    2179  0
;    2180  0          $debug_print_fao (string)
;  M 2181  0                            =
;  M 2182  0                            %IF switch_debug
;  M 2183  0                            %THEN
;  M 2184  0                                    BEGIN
;  M 2185  0                                    EXTERNAL ROUTINE  exch$util_fao_buffer;
;  M 2186  0                                    lib$put_output (
;  M 2187  0                                        exch$util_fao_buffer (%ASCID %STRING ($dbgtrc_prefix_string, string)
;  M 2188  0                                            %IF %LENGTH GTR 1 %THEN ,%REMAINING %FI))
;  M 2189  0                                    END
;  M 2190  0                            %FI ! switch_debug
;    2191  0                            %;
```

```
;    2192  0      ! Macros to manipulate queues
;    2193  0      !
;    2194  0      MACRO
;    2195  0
;    2196  0          ! Initialize the header of a queue.  This means make each of the 2 pointers in the header point to the header.
;    2197  0          !
;    2198  0          $queue_initialize (q_header)
;M   2199  0                            =
;M   2200  0                            BEGIN
;M   2201  0
;M   2202  0                            BIND
;M   2203  0                                _qh_ = (q_header) : VECTOR [2, LONG];
;M   2204  0
;M   2205  0                            _qh_ [0] = _qh_;
;M   2206  0                            _qh_ [1] = _qh_;
;M   2207  0
;M   2208  0                            END
;    2209  0                            %,
;    2210  0
;    2211  0          ! Insert an element at the head of a queue.
;    2212  0          !
;    2213  0          $queue_insert_head (item, q_header)
;M   2214  0                            =
;M   2215  0                            BEGIN
;M   2216  0
;M   2217  0                            BUILTIN
;M   2218  0                                INSQUE;
;M   2219  0                            BIND
;M   2220  0                                _qh_ = (q_header) : VECTOR [2, LONG];
;M   2221  0
;M   2222  0                            INSQUE ((item), _qh_ [0])
;M   2223  0
;M   2224  0                            END
;    2225  0                            %,
;    2226  0
;    2227  0          ! Insert an element at the tail of a queue.
;    2228  0          !
;    2229  0          $queue_insert_tail (item, q_header)
;M   2230  0                            =
;M   2231  0                            BEGIN
;M   2232  0
;M   2233  0                            BUILTIN
;M   2234  0                               INSQUE;
;M   2235  0                            BIND
;M   2236  0                                _qh_ = (q_header) : VECTOR [2, LONG];
;M   2237  0
;M   2238  0                            INSQUE ((item), ._qh_ [1])
;M   2239  0
;M   2240  0                            END
;    2241  0                            %,
;    2242  0          !+
;    2243  0          ! Remove the indicated element from a queue.  The first parameter is the address of the element.  The second
;    2244  0          ! parameter is optional.
;    2245  0          !
;    2246  0          ! If supplied, it is the address of a longword in which to store the element removed from the queue or 0 if
;    2247  0          ! no element was present in the queue.  The value of the expression is TRUE is a element was removed from the
;    2248  0          ! queue and FALSE otherwise.
```

```
;    2249  0          !
;    2250  0          ! If the second parameter is not supplied, the value of the expression is the address of the element removed
;    2251  0          ! from the queue or 0 if no element was present in the queue.
;    2252  0          !-
;    2253  0          $queue_remove (q_element, element)
;  M 2254  0                        =
;  M 2255  0                        BEGIN
;  M 2256  0
;  M 2257  0                        BIND
;  M 2258  0                            qhead_ = (q_element) : VECTOR [2, LONG];
;  M 2259  0                        BUILTIN
;  M 2260  0                            REMQUE;
;  M 2261  0
;  M 2262  0                        %IF (%NULL (element))
;  M 2263  0                        %THEN
;  M 2264  0                            LOCAL
;  M 2265  0                                _T_ : REF VECTOR [2, LONG];
;  M 2266  0                        %ELSE
;  M 2267  0                            BIND
;  M 2268  0                                _T_ = (element) : REF VECTOR [2, LONG];
;  M 2269  0                        %FI
;  M 2270  0
;  M 2271  0                        IF (REMQUE (_qhead_, _T_))
;  M 2272  0                        THEN
;  M 2273  0                            BEGIN
;  M 2274  0
;  M 2275  0                            ! queue was empty
;  M 2276  0                            !
;  M 2277  0                            IF (%NULL (element))
;  M 2278  0                            THEN
;  M 2279  0                                0
;  M 2280  0                            ELSE
;  M 2281  0                                (_T_ = 0; FALSE)
;  M 2282  0                            END
;  M 2283  0                        ELSE
;  M 2284  0                            BEGIN
;  M 2285  0
;  M 2286  0                            IF (%NULL (element))
;  M 2287  0                            THEN
;  M 2288  0                                _T_
;  M 2289  0                            ELSE
;  M 2290  0                                true
;  M 2291  0                            END
;  M 2292  0                        END
;    2293  0                        %,
;    2294  0          !+
;    2295  0          ! Remove an element from the head of a queue.  The first parameter is the address of the queue header.  The
;    2296  0          ! second parameter is optional.
;    2297  0          !
;    2298  0          ! If supplied, it is the address of a longword in which to store the element removed from the queue or 0 if
;    2299  0          ! no element was present in the queue.  The value of the expression is TRUE is a element was removed from the
;    2300  0          ! queue and FALSE otherwise.
;    2301  0          !
;    2302  0          ! If the second parameter is not supplied, the value of the expression is the address of the element removed
;    2303  0          ! from the queue or 0 if no element was present in the queue.
;    2304  0          !-
;    2305  0          $queue_remove_head (q_header, element)
```

```
;  M 2306  0                        =
;  M 2307  0                        BEGIN
;  M 2308  0
;  M 2309  0                        BIND
;  M 2310  0                            _qh_ = (q_header) : VECTOR [2, LONG];
;  M 2311  0
;  M 2312  0                        %IF (%NULL (element))
;  M 2313  0                        %THEN
;  M 2314  0                            $queue_remove (._qh_ [0])
;  M 2315  0                        %ELSE
;  M 2316  0                            $queue_remove (._qh_ [0], element)
;  M 2317  0                        %FI
;  M 2318  0
;  M 2319  0                        END
;     2320  0                        %,
;     2321  0            !+
;     2322  0            ! Remove an element from the tail of a queue.  The first parameter is the address of the queue header.  The
;     2323  0            ! second parameter is optional.
;     2324  0            !
;     2325  0            ! If supplied, it is the address of a longword in which to store the element removed from the queue or 0 if
;     2326  0            ! no element was present in the queue.  The value of the expression is TRUE is a element was removed from the
;     2327  0            ! queue and FALSE otherwise.
;     2328  0            !
;     2329  0            ! If the second parameter is not supplied, the value of the expression is the address of the element removed
;     2330  0            ! from the queue or 0 if no element was present in the queue.
;     2331  0            !-
;     2332  0           $queue_remove_tail (q_header, element)
;  M 2333  0                        =
;  M 2334  0                        BEGIN
;  M 2335  0
;  M 2336  0                        BIND
;  M 2337  0                            _qh_ = (q_header) : VECTOR [2, LONG];
;  M 2338  0
;  M 2339  0                        %IF (%NULL (element))
;  M 2340  0                        %THEN
;  M 2341  0                            $queue_remove (._qh_ [1])
;  M 2342  0                        %ELSE
;  M 2343  0                            $queue_remove (._qh_ [1], element)
;  M 2344  0                        %FI
;  M 2345  0
;  M 2346  0                        END
;     2347  0                        %,
;     2348  0
;     2349  0            ! Test a queue for emptiness.  TRUE if the queue is empty, FALSE if the queue is not empty
;     2350  0            !
;     2351  0           $queue_empty (q_header)
;  M 2352  0                        =
;  M 2353  0                        BEGIN
;  M 2354  0
;  M 2355  0                        BIND
;  M 2356  0                            _qh_ = (q_header) : VECTOR [2, LONG];
;  M 2357  0
;  M 2358  0                        _qh_ EQLA ._qh_ [0]
;  M 2359  0
;  M 2360  0                        END
;     2361  0                        %;
```

```
;    2362  0      !
;    2363  0      ! Literal definitions:
;    2364  0      !
;    2365  0
;    2366  0      ! define literals for BLISS true and false values
;    2367  0      !
;    2368  0      LITERAL
;    2369  0          true = 1,
;    2370  0          false = 0
;    2371  0          .
;    2372  0
;    2373  0      ! Define values of some ASCII characters
;    2374  0      !
;    2375  0      LITERAL
;    2376  0          NUL = 0,                                    ! null
;    2377  0          LF  = 10,                                   ! line feed
;    2378  0          VT  = 11,                                   ! vertical tab
;    2379  0          FF  = 12,                                   ! form feed
;    2380  0          CR  = 13,                                   ! carriage return
;    2381  0          CTRLZ = 26,                                 ! control z
;    2382  0          ESC = 27,                                   ! escape
;    2383  0          DEL = 127                                   ! rubout
;    2384  0          ;
;    2385  0
;    2386  0      ! Define the Radix-50 equivalents for FILE.BAD
;    2387  0      !
;    2388  0      LITERAL
;    2389  0          R50_EMPTY = %RAD50_11 'EMPTY ',             ! longword "EMPTY "
;    2390  0          R50_FIL   = %RAD50_11 'FIL',                ! word "FIL"
;    2391  0          R50_FILE = %X '1F4026F4',                   ! longword "FILE  "
;    2392  0          R50_BAD  = %X '0CAC',                       ! word "BAD"
;    2393  0          R50_SYS  = %X '7ABB'                        ! word "SYS"
;    2394  0          ;
;    2395  0
;    2396  0      !
;    2397  0      ! Linkage definitions:
;    2398  0      !
;    2399  0      LINKAGE
;    2400  0          jsb_r0r1    = JSB (REGISTER=0, REGISTER=1)
;    2401  0                              : NOPRESERVE(0,1)       NOTUSED(2,3,4,5,6,7,8,9,10,11),
;    2402  0          jsb_r0r1r2  = JSB (REGISTER=0, REGISTER=1, REGISTER=2)
;    2403  0                              : NOPRESERVE(0,1,2)     NOTUSED(3,4,5,6,7,8,9,10,11),
;    2404  0          jsb_r1      = JSB (REGISTER=1)
;    2405  0                              : NOPRESERVE(0,1)       NOTUSED(2,3,4,5,6,7,8,9,10,11),
;    2406  0          jsb_r1r2    = JSB (REGISTER=1, REGISTER=2)
;    2407  0                              : NOPRESERVE(0,1,2)     NOTUSED(3,4,5,6,7,8,9,10,11),
;    2408  0          jsb_r1r2r3  = JSB (REGISTER=1, REGISTER=2, REGISTER=3)
;    2409  0                              : NOPRESERVE(0,1,2,3)   NOTUSED(4,5,6,7,8,9,10,11),
;    2410  0          jsb_r2r3    = JSB (REGISTER=2, REGISTER=3)
;    2411  0                              : NOPRESERVE(0,1,2,3)   NOTUSED(4,5,6,7,8,9,10,11),
;    2412  0          jsb_r3r4    = JSB (REGISTER=3, REGISTER=4)
;    2413  0                              : NOPRESERVE(0,1,2,3,4) NOTUSED(5,6,7,8,9,10,11),
;    2414  0          jsb_get     = JSB (REGISTER=5, REGISTER=6, REGISTER=7)
;    2415  0                              : NOPRESERVE(0,1,2,3,4,5,6,7) NOTUSED(8,9,10,11),
;    2416  0          jsb_put     = JSB (REGISTER=9, REGISTER=10)
;    2417  0                              : NOPRESERVE(0,1,2,3,4,5,6,7,8,9,10) NOTUSED(11)
;    2418  0          ;
```

```
;  2419  0      ! Run-time library and other routines external to the facility
;  2420  0      !
;  2421  0      EXTERNAL ROUTINE
;  2422  0          cli$dcl_parse : ADDRESSING_MODE (GENERAL),  ! Command parsing routine
;  2423  0          cli$dispatch  : ADDRESSING_MODE (GENERAL),  ! Action routine dispatch
;  2424  0          cli$get_value : ADDRESSING_MODE (GENERAL),  ! Entity value fetch
;  2425  0          cli$present   : ADDRESSING_MODE (GENERAL),  ! Entity presence boolean
;  2426  0          lib$find_file : ADDRESSING_MODE (GENERAL),  ! Wildcard files-11 processing
;  2427  0          lib$free_vm   : ADDRESSING_MODE (GENERAL),  ! Releases memory
;  2428  0          lib$get_input : ADDRESSING_MODE (GENERAL),  ! Get a line from SYS$INPUT
;  2429  0          lib$get_vm    : ADDRESSING_MODE (GENERAL),  ! Gets memory
;  2430  0          lib$put_output: ADDRESSING_MODE (GENERAL),  ! Display a line on SYS$OUTPUT
;  2431  0          ots$cvt_ti_l  : ADDRESSING_MODE (GENERAL),  ! ASCII decimal to longword
;  2432  0          ots$cvt_to_l  : ADDRESSING_MODE (GENERAL),  ! ASCII octal to longword
;  2433  0          ots$cvt_tz_l  : ADDRESSING_MODE (GENERAL),  ! ASCII hexadecimal to longword
;  2434  0          str$copy_dx   : ADDRESSING_MODE (GENERAL),  ! Copy string of any class
;  2435  0          str$free1_dx  : ADDRESSING_MODE (GENERAL)   ! Release dynamic string
;  2436  0          ;
;  2437  0
;  2438  0      ! Define the lengths of control blocks here - Many of these need to be adjusted by system block sizes, so it
;  2439  0      ! can't be completely done in the SDL definition.
;  2440  0      !
;  2441  0      LITERAL
;  2442  0
;  2443  0          ! An $EXCG is the global environment for the facility, the SDL block plus two RMS work areas
;  2444  0          !
;  2445  0          exchblk$s_excg = excg$k_length + 2*(fab$k_bln + rab$k_bln + nam$k_bln + (2*nam$c_maxrss)),
;  2446  0
;  2447  0          ! An $RMSB describes an RMS file, the SDL block plus one RMS work area
;  2448  0          !
;  2449  0          exchblk$s_rmsb = rmsb$k_length + fab$k_bln + rab$k_bln + nam$k_bln + (2*nam$c_maxrss),
;  2450  0
;  2451  0          ! A $VOLB contains the structures for a volume, the SDL block plus one RMS work area
;  2452  0          !
;  2453  0          exchblk$s_volb = volb$k_length + fab$k_bln + rab$k_bln + nam$k_bln + (2*nam$c_maxrss),
;  2454  0
;  2455  0          ! The following don't need adjusting, but we want to keep all the  EXCHBLK$S_ definitions in one place
;  2456  0          !
;  2457  0          exchblk$s_copy = copy$k_length,             ! Size of the work area for the COPY command
;  2458  0          exchblk$s_dire = dire$k_length,             ! Size of the work area for the DIRECTORY command
;  2459  0          exchblk$s_dos11 = dos11$k_length,           ! Size of the DOS-11 specific extension to the volb
;  2460  0          exchblk$s_dos11ctx = dos1Tctx$k_length,     ! Size of the DOS-11 file context block
;  2461  0          exchblk$s_filb = filb$k_length,             ! A $FILB is a structure which describes an open file
;  2462  0          exchblk$s_init = init$k_length,             ! Size of the work area for the INIT command
;  2463  0          exchblk$s_namb = namb$k_length,             ! A $NAMB is a structure which stores a fully parsed file name
;  2464  0          exchblk$s_moun = moun$k_length,             ! Size of the work area for the MOUNT command
;  2465  0          exchblk$s_rt11 = rt11$k_length,             ! Size of the RT-11 specific extension to the volb
;  2466  0          exchblk$s_rt11ctx = rt1Tctx$k_length,       ! Size of the RT-11 file context block
;  2467  0          exchblk$s_rtnam = rtnam$k_length            ! Size of the work area for the DIRECTORY command
;  2468  0          ;
```

```
;  2469  0      ! Message codes defined in SRC$:EXCMSG.MSG
;  2470  0      !
;  2471  0      EXTERNAL LITERAL
;  2472  0          exch$_accessfail,           ! failed to access volume ($GETDVI service failure)
;  2473  0          exch$_badfilename,          ! File name not valid for given volume
;  2474  0          exch$_badlogic,             ! Adds error number to shared message
;  2475  0          exch$_badpad,               ! Improper /RECORD_FORMAT=PAD option
;  2476  0          exch$_binchksum,            ! Bad formatted binary record
;  2477  0          exch$_binrecfmt,            ! Bad formatted binary record
;  2478  0          exch$_blockcheck,           ! Block check failed
;  2479  0          exch$_blockcheck0,          ! Block check failed because block address is 0
;  2480  0          exch$_canceled,             ! Command canceled
;  2481  0          exch$_closeerr,             ! Error closing file
;  2482  0          exch$_closeforeign,         ! Error closing foreign devire
;  2483  0          exch$_copied,               ! Log message for copy command
;  2484  0          exch$_copyboot,             ! Log message for copy /boot command
;  2485  0          exch$_copnewname,           ! File copied with new name
;  2486  0          exch$_createvirt,           ! Error creating virtual volume
;  2487  0          exch$_deleted,              ! Deleted copy of a file
;  2488  0          exch$_deleteprev,           ! Deleted previous copy of a file
;  2489  0          exch$_devonly,              ! Device spec only, other parts of file name ignored
;  2490  0          exch$_devnotsuit,           ! Device is not suitable for EXCHANGE
;  2491  0          exch$_dire_error,           ! Error writing directory
;  2492  0          exch$_dismounted,           ! Device has been dismounted
;  2493  0          exch$_dos11_badlabel,       ! Invalid label found on dos11 tape
;  2494  0          exch$_dos11_blocksize,      ! Invalid block (>512 bytes) found on dos11 tape
;  2495  0          exch$_dos11_ioerror,        ! Error during I/O on dos11 tape
;  2496  0          exch$_dos11_position,       ! Rewinding tape to find correct position
;  2497  0          exch$_filenotfound,         ! Unable to locate file
;  2498  0          exch$_fil11_norec,          ! No /RECORD for files-11
;  2499  0          exch$_ignore_dire,          ! Ignoring directory specification
;  2500  0          exch$_ignore_vers,          ! Ignoring file version number
;  2501  0          exch$_illmtcopy,            ! Illegal magtape copy, input and output on same device
;  2502  0          exch$_initialized,          ! Device has been initialized
;  2503  0          exch$_invrecfmt,            ! Record format not valid for volume type
;  2504  0          exch$_invvolfmt,            ! Volume format not valid for operation
;  2505  0          exch$_many_to_one,          ! Multiple input files were given but only one output file
;  2506  0          exch$_mounted,              ! Volume mounted                (success)
;  2507  0          exch$_mounterror,           ! Error performing VMS $mount service
;  2508  0          exch$_mountvir,             ! Virtual volume mounted        (success)
;  2509  0          exch$_noalloc,              ! /ALLOCATE ignored on tape output
;  2510  0          exch$_nocarriage,           ! /CARRIAGE ignored on output
;  2511  0          exch$_nocopbad,             ! Couldn't create, .BAD file with wildcarded names
;  2512  0          exch$_nocopbaddel,          ! Couldn't create, have to delete .BAD file
;  2513  0          exch$_nocopdup,             ! Couldn't create, already created same name
;  2514  0          exch$_nocoplock,            ! Couldn't create, volume is writelocked
;  2515  0          exch$_nocopnodel,           ! Couldn't create, file of same name and /NODELETE given
;  2516  0          exch$_nocopprot,            ! Couldn't create, file of same name protected against modification
;  2517  0          exch$_nocopsamdev,          ! Illegal copy to same device
;  2518  0          exch$_nocopsysdel,          ! Illegal copy of .SYS when existing .SYS present
;  2519  0          exch$_nocopyboot,           ! Unable to copy boot info
;  2520  0          exch$_nodellock,            ! File not deleted, volume locked
;  2521  0          exch$_nodevice,             ! Device spec missing
;  2522  0          exch$_noremote,             ! Device spec cannot have node field
;  2523  0          exch$_norendev,             ! Illegal rename to different device
;  2524  0          exch$_norenexists,          ! Not renamed, already exits
;  2525  0          exch$_norenlock,            ! Files not renamed, volume locked
```

```
;    2526  0          exch$_nosysact,                 ! No action on .SYS files
;    2527  0          exch$_notcopied,                ! File not copied
;    2528  0          exch$_notcop_retry,             ! File not copied, will retry
;    2529  0          exch$_notdeleted,               ! File not deleted
;    2530  0      !\    exch$_notimplement,                 ! Feature not yet implemented
;    2531  0          exch$_notmounted,               ! Device is not mounted on EXCHANGE
;    2532  0          exch$_notsamedev,               ! Input and output not same device (copy /boot)
;    2533  0          exch$_notvallen,                ! /REC=LEN requires FIXED
;    2534  0          exch$_novolumes,                ! No volumes are mounted
;    2535  0          exch$_openforeign,              ! Open failed on a foreign volume
;    2536  0          exch$_openvirtual,              ! Open failed on a virtual volume
;    2537  0          exch$_opnotperdos,              ! Operation not permitted on DOS-11 volume
;    2538  0          exch$_opnotperfil,              ! Operation not permitted on Files-11 volume
;    2539  0      !\  exch$_opnotperrtl,              ! Operation not permitted on RT-11 volume (not yet needed)
;    2540  0      !\  exch$_opnotperrtmt,             ! Operation not permitted on RT-11 magtape volume
;    2541  0          exch$_parseerr,                 ! Bad file parameter syntax
;    2542  0          exch$_partcopied,               ! File partially copied
;    2543  0          exch$_readcheck,                ! Error detected during read check
;    2544  0          exch$_readcheckrec,             ! Error detected during read check was recovered
;    2545  0          exch$_readerrrec,               ! Error detected during read was recovered
;    2546  0          exch$_recover,                  ! Directory recovery message
;    2547  0          exch$_rectoobig,                ! Bad formatted binary record
;    2548  0          exch$_renamed,                  ! File renamed log message
;    2549  0          exch$_rtl1_baddirect,           ! RT-11 directory error
;    2550  0          exch$_rtl1_badfile,             ! Bad block file created
;    2551  0          exch$_rtl1_bigbadfile,          ! Bad block file contains some good blocks
;    2552  0          exch$_rtl1_dirsize,             ! Device size disagrees with directory size
;    2553  0          exch$_rtl1_errlock,             ! RT-11 directory error
;    2554  0          exch$_rtl1_extra,               ! Too many extra words
;    2555  0          exch$_rtl1_noend,               ! RT-11 directory error
;    2556  0          exch$_rtl1_overflow,            ! RT-11 directory error
;    2557  0          exch$_rtl1_stblock,             ! RT-11 directory error
;    2558  0          exch$_rtl1_toomanyblk,          ! RT-11 directory error
;    2559  0          exch$_rtl1_toomanyseg,          ! RT-11 directory error
;    2560  0          exch$_rtl1_unkent,              ! RT-11 directory error
;    2561  0          exch$_rtouteof,                 ! End of file on output file
;    2562  0          exch$_rtprotect,                ! File protected against modification
;    2563  0          exch$_stmrecfmt,                ! Bad stream record format
;    2564  0          exch$_stnotavail,               ! Start block not available
;    2565  0          exch$_strtnomulti,              ! Can't say /START with multiple input files
;    2566  0          exch$_toomanycol,               ! Too many columns requested
;    2567  0          exch$_trace,                    ! Header for a status trace
;    2568  0          exch$_typed,                    ! Log message for type command
;    2569  0          exch$_virtnochange,             ! Cannot change size of virtual devices
;    2570  0          exch$_vmsmount,                 ! Volume has been mounted on VMS
;    2571  0          exch$_volmount,                 ! Volume is already mounted
;    2572  0          exch$_volume_full,              ! Output volume is full
;    2573  0          exch$_waiterr,                  ! Error waiting for RMS operation
;    2574  0          exch$_writecache,               ! Writing modified directory segments
;    2575  0          exch$_writecheck,               ! Error detected during write check
;    2576  0          exch$_writecheckrec,            ! Error detected during write check was recovered
;    2577  0          exch$_writeerrrec,              ! Error detected during write was recovered
;    2578  0          exch$_writelock                 ! Volume is write-locked
;    2579  0          ;
;    2580  0
;    2581  0      ! Shared message definitions
;    2582  0      !
```

```
;  P 2583  0     $shr_msgdef
;  P 2584  0         (exch, 248, local,
;  P 2585  0     !       (badlogic, warning),                    ! Using private message so can add error number
;  P 2586  0             (badvalue, warning),
;  P 2587  0             (closeout, warning),
;  P 2588  0             (confqual, warning),
;  P 2589  0             (insvirmem, warning),
;  P 2590  0             (openin, warning),
;  P 2591  0             (openout, warning),
;  P 2592  0             (readerr, warning),
;  P 2593  0             (writeerr, warning)
;     2594  0         );
;     2595  0
;  P 2596  0     $shr_msgdef                                     ! Message from CLI that syntax error occurred
;  P 2597  0         (msg, 3, local,
;  P 2598  0             (syntax, severe)
;     2599  0         );
;     2600  0
;     2601  0     ! Other symbols which need explicit declarations
;     2602  0     !
;     2603  0     EXTERNAL LITERAL
;     2604  0         cli$_comma,                                 ! Parameter ended with a comma
;     2605  0         cli$_concat,                                ! Parameter ended with a plus sign
;     2606  0         cli$_locneg,                                ! An explicit /NOqual for local qual
;     2607  0         cli$_locpres,                               ! An explicit /qual for local qual
;     2608  0         cli$_nocomd,                                ! CLI saw a blank line and burped
;     2609  0         cli$_negated,                               ! An explicit /NOqual was given
;     2610  0         cli$_present,                               ! An explicit /qual was given
;     2611  0         cli$_facility;                              ! CLI facility code
;     2612  0
;     2613  0     ! Storage external to all modules
;     2614  0     !
;     2615  0     EXTERNAL
;     2616  0         exch$cld_table : ADDRESSING_MODE (LONG_RELATIVE)        ! Command table for CLI$DCL_PARSE
;     2617  0         ;
;     2618  0
;     2619  0     ! External data - defined in EXCH$MAIN module
;     2620  0     !
;     2621  0     EXTERNAL
;     2622  0         exch$gq_dyn_str_template : $desc_block ADDRESSING_MODE (LONG_RELATIVE),     ! An initialized, null dynamic strin
;     2623  0         exch$a_gbl  : REF BLOCK [,BYTE] ADDRESSING_MODE (LONG_RELATIVE)             ! The pointer to the known world
;     2624  0         ;
;     2625  0
;     2626  0     ! END                                           ! End of module EXCLIB
;     2627  0     ! ELUDOM


;                          Library Statistics
;
;                                    -------- Symbols --------      Pages       Processing
;      File                          Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1      18619        33         0      1000          00:01.8
```

```
;                                       COMMAND QUALIFIERS

;       BLISS/LIBRARY=LIB$:/LIST=LIS$: SRC$:EXCLIB

; Run Time:         00:21.2
; Elapsed Time:     01:09.7
; Lines/CPU Min:     7434
; Lexemes/CPU-Min: 40975
; Memory Used:  279 pages
; Library Precompilation Complete
```

EXCFIL11
LIS

EXCINIT
LIS

EXCLIB
LIS

EXCIO
LIS

EXCRT11.
LIS

EXCMAIN
LIS

EXCMSG
LIS

EXCPDP.
LIS

EXCMOUN
LIS