```
RRRRRRRRRRR      UUU          UUU  NNN          NNN     000000000     FFFFFFFFFFFFFFF  FFFFFFFFFFFFFFF
RRRRRRRRRRRR     UUU          UUU  NNN          NNN     000000000     FFFFFFFFFFFFFFF  FFFFFFFFFFFFFFF
RRRRRRRRRRRR     UUU          UUU  NNN          NNN     000000000     FFFFFFFFFFFFFFF  FFFFFFFFFFFFFFF
RRR          RRR UUU          UUU  NNN          NNN  000         000  FFF              FFF
RRR          RRR UUU          UUU  NNN          NNN  000         000  FFF              FFF
RRR          RRR UUU          UUU  NNN          NNN  000         000  FFF              FFF
RRR          RRR UUU          UUU  NNNNNN       NNN  000         000  FFF              FFF
RRR          RRR UUU          UUU  NNNNNN       NNN  000         000  FFF              FFF
RRR          RRR UUU          UUU  NNNNNN       NNN  000         000  FFF              FFF
RRRRRRRRRRRR     UUU          UUU  NNN  NNN     NNN  000         000  FFFFFFFFFFFF     FFFFFFFFFFFF
RRRRRRRRRRRR     UUU          UUU  NNN  NNN     NNN  000         000  FFFFFFFFFFFF     FFFFFFFFFFFF
RRRRRRRRRRRR     UUU          UUU  NNN  NNN     NNN  000         000  FFFFFFFFFFFF     FFFFFFFFFFFF
RRR    RRR       UUU          UUU  NNN       NNNNNN  000         000  FFF              FFF
RRR    RRR       UUU          UUU  NNN       NNNNNN  000         000  FFF              FFF
RRR    RRR       UUU          UUU  NNN       NNNNNN  000         000  FFF              FFF
RRR       RRR    UUU          UUU  NNN          NNN  000         000  FFF              FFF
RRR       RRR    UUU          UUU  NNN          NNN  000         000  FFF              FFF
RRR       RRR    UUU          UUU  NNN          NNN  000         000  FFF              FFF
RRR          RRR UUUUUUUUUUUUUUU  NNN          NNN     000000000     FFF              FFF
RRR          RRR UUUUUUUUUUUUUUU  NNN          NNN     000000000     FFF              FFF
RRR          RRR UUUUUUUUUUUUUUU  NNN          NNN     000000000     FFF              FFF
RRR          RRR UUUUUUUUUUUUUUU  NNN          NNN     000000000     FFF              FFF
RRR          RRR UUUUUUUUUUUUUUU  NNN          NNN     000000000     FFF              FFF
```

```
_$2

Sym
---
NDX
NDX
NUM
NUM
OPE
OUT
PAC
PAC
PAC
PAC
PAC
PAC
PAC
PAD
PAG
PAG
PAG
PAG
PAG
PAG
PAG
PER
PUT
RCO
RIN

RLI
RNO
RNO
RTY
SAV
STR
STR
STR
STR
STR
STR
STR

STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
STR
```

```
RRRRRRRR     UU      UU  NN      NN    000000    FFFFFFFFFF  FFFFFFFFFF
RRRRRRRR     UU      UU  NN      NN    000000    FFFFFFFFFF  FFFFFFFFFF
RR      RR   UU      UU  NN      NN  00      00  FF          FF
RR      RR   UU      UU  NN      NN  00      00  FF          FF
RR      RR   UU      UU  NNNN    NN  00      00  FF          FF
RR      RR   UU      UU  NNNN    NN  00      00  FF          FF
RRRRRRRR     UU      UU  NN  NN  NN  00      00  FFFFFFFF    FFFFFFFF
RRRRRRRR     UU      UU  NN  NN  NN  00      00  FFFFFFFF    FFFFFFFF
RR   RR      UU      UU  NN    NNNN  00      00  FF          FF
RR   RR      UU      UU  NN    NNNN  00      00  FF          FF
RR     RR    UU      UU  NN      NN  00      00  FF          FF
RR     RR    UU      UU  NN      NN  00      00  FF          FF          ....
RR       RR  UUUUUUUUUU  NN      NN    000000    FF          FF          ....
RR       RR  UUUUUUUUUU  NN      NN    000000    FF          FF          ....
```

```
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
```

```
;  1          0001  0 %TITLE 'Overall DSR process control'
;  2          0002  0 MODULE runoff ( IDENT = 'V04-000'
;  3        C 0003  0                 %BLISS32[, ADDRESSING_MODE (EXTERNAL    = general %(,
;  4          0004  0                                             NONEXTERNAL = long_relative)%)]
;  5          0005  0                 ) =
;  6          0006  1 BEGIN                                   !Start of module
;  7          0007  1
;  8          0008  1 !
;  9          0009  1 !*****************************************************************************
; 10          0010  1 !*                                                                           *
; 11          0011  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
; 12          0012  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
; 13          0013  1 !*  ALL RIGHTS RESERVED.                                                      *
; 14          0014  1 !*                                                                           *
; 15          0015  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
; 16          0016  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
; 17          0017  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
; 18          0018  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
; 19          0019  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
; 20          0020  1 !*  TRANSFERRED.                                                              *
; 21          0021  1 !*                                                                           *
; 22          0022  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
; 23          0023  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
; 24          0024  1 !*  CORPORATION.                                                              *
; 25          0025  1 !*                                                                           *
; 26          0026  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
; 27          0027  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
; 28          0028  1 !*                                                                           *
; 29          0029  1 !*                                                                           *
; 30          0030  1 !*****************************************************************************
; 31          0031  1 !
; 32          0032  1 !++
; 33          0033  1 ! FACILITY:      DSR (Digital Standard RUNOFF) / DSRPLUS
; 34          0034  1 !
; 35          0035  1 ! ABSTRACT:      Main program.
; 36          0036  1 !
; 37          0037  1 ! ENVIRONMENT:   Transportable
; 38          0038  1 !
; 39          0039  1 ! AUTHOR:        R.W.Friday      CREATION DATE:  April, 1978
; 40          0040  1 !
```

```
;    42     0041  1 !
;    43     0042  1 ! MODIFIED BY:
;    44     0043  1 !
;    45     0044  1 !       040     REM00040        Ray Marshall    25-Mar-1984
;    46     0045  1 !               Fixed the call to LIB$FIND_FILE.  The new user flags parameter
;    47     0046  1 !                 must be passed by reference, not value.
;    48     0047  1 !
;    49     0048  1 !       039     REM00039        Ray Marshall    16-Feb -> 15-Mar-1984
;    50     0049  1 !               Added routine INPUT_FILE_SCAN to make appropriate calls to
;    51     0050  1 !                 LIB$FIND_FILE.  This is code to support directory search
;    52     0051  1 !                 string logic implemented in VMS V4, and as such, is
;    53     0052  1 !                 conditionalized to only compile for VMS and will only
;    54     0053  1 !                 excute on a V4 or later system.  It will only be compiled
;    55     0054  1 !                 into BLISS32 implementations.
;    56     0055  1 !
;    57     0056  1 !       038     KFA00038        Ken Alden       12-Aug-1983
;    58     0057  1 !               Fixed a /auto bug that forces a second run if any crefs
;    59     0058  1 !                 were unresolved (usu. a forward reference).
;    60     0059  1 !
;    61     0060  1 !       037     REM00037        Ray Marshall    9-Aug-1983
;    62     0061  1 !               Modified the init-file logic to support DSRPLUS.INIT and
;    63     0062  1 !                 RUNOFF.INIT in the TOPS-20 implementation of DSRPLUS
;    64     0063  1 !                 (replacing the logicals used in the VMS implementation).
;    65     0064  1 !
;    66     0065  1 !       036     REM00036        Ray Marshall    22-Jul-1983
;    67     0066  1 !               Defined TTYSET and ENAPSI as NOVALUE to match the way they
;    68     0067  1 !                 are coded.
;    69     0068  1 !               Moved GCA_PASS_COUNT out of being BLISS32 specific because
;    70     0069  1 !                 it is looked at in the routine OUTCREF which is part of all
;    71     0070  1 !                 implementations of DSRPLUS.
;    72     0071  1 !
;    73     0072  1 !       035     KFA00035        Ken Alden       15-Jul-1983
;    74     0073  1 !               Tweaked the /auto logic to prevent a premature MEM file.
;    75     0074  1 !
;    76     0075  1 !       034     KFA00034        Ken Alden       28-Jun-1983
;    77     0076  1 !               Added formal parameter to the SETQUICK call.
;    78     0077  1 !
;    79     0078  1 !       033     KFA00033        Ken Alden       27-Jun-1983
;    80     0079  1 !               Added more logic to the setquick check using the cref_errcnt
;    81     0080  1 !                 as an additional check item.
;    82     0081  1 !
;    83     0082  1 !       032     KFA00032        Ken Alden       24-Jun-1983
;    84     0083  1 !               This edit change did exactly what 31 was supposed to do.
;    85     0084  1 !                 The big problem was the gca_black_box was getting turned
;    86     0085  1 !                 off after the first run, from RINIT.  Now, gca_black_box
;    87     0086  1 !                 is not check, the rno_cmd[automatic] is.
;    88     0087  1 !
;    89     0088  1 !       031     KAD00031        Keith Dawson    7-Jun-1983
;    90     0089  1 !               Minor tweak to the foregoing: make sure no .MEM is gen-
;    91     0090  1 !                 erated (QUICK is set) on reruns caused because a cross-
;    92     0091  1 !                 reference changed.
;    93     0092  1 !
;    94     0093  1 !       030     KAD00030        Keith Dawson    27-May-1983
;    95     0094  1 !               For DSRPLUS/AUTO, check number of iterations for Cref oscil-
;    96     0095  1 !                 lation; quit after the 4th pass.
;    97     0096  1 !
;    98     0097  1 !       029     KAD00029        Keith Dawson    17-May-1983
```

```
;   99          0098  1 !                For FLIP, add TCXINI record to initialization records
;  100          0099  1 !                  written to the .BFL file.
;  101          0100  1 !
;  102          0101  1 !        028     REM00028        Ray Marshall    10-May-1983
;  103          0102  1 !                RNFXFL has been removed, so the reference herein used to check
;  104          0103  1 !                  for parallelism of MSGTXT.REQ (via RNFERM.BLI) with
;  105          0104  1 !                  DSRMSG.MSG (or DPLMSG.MSG for DSRPLUS) has been changed to
;  106          0105  1 !                  refer to RNFWFR.
;  107          0106  1 !
;  108          0107  1 !        027     KAD00027        Keith Dawson    3-May-1983
;  109          0108  1 !                Fix bug: if /AUTO, any errors in Pass 1 would stop processing.
;  110          0109  1 !
;  111          0110  1 !        026     KAD00026        Keith Dawson    19-Apr-1983
;  112          0111  1 !                Fixed bug whereby the results of /DEVICE were not getting
;  113          0112  1 !                  picked up in DSR.
;  114          0113  1 !
;  115          0114  1 !        025     KAD00025        Keith Dawson    5-Apr-1983
;  116          0115  1 !                Full support for /CROSS and /AUTO.
;  117          0116  1 !
;  118          0117  1 !        024     KAD00024        Keith Dawson    23-March-1983
;  119          0118  1 !                Changed GCA_FLIP bit to (.gca_op_dev EQL op_dev_flip).
;  120          0119  1 !
;  121          0120  1 !        023     REM00023        Ray Marshall    07-Mar-1983
;  122          0121  1 !                Global edit of all modules. Updated module names, idents,
;  123          0122  1 !                copyright dates. Changed require files to BLISS library.
;  124          0123  1 !--
```

```
;   126          0124  1 !
;   127          0125  1 ! TABLE OF CONTENTS:
;   128          0126  1 !
;   129          0127  1
;   130          0128  1 REQUIRE 'REQ:RNODEF';                   ! RUNOFF variant definitions
;   131          0259  1
;   132          0260  1 FORWARD ROUTINE
;   133          0261  1     runoff,
;   134          0262  1     grab_resultant,
;   135        U 0263  1 %IF DSRPLUS %THEN
;   136        U 0264  1     init_files : NOVALUE,
;   137          0265  1 %FI
;   138          0266  1 %IF %BLISS(BLISS32) %THEN
;   139          0267  1     input_file_scan,
;   140          0268  1 %FI
;   141          0269  1     input_loop;
;   142          0270  1
;   143          0271  1 !
;   144          0272  1 ! INCLUDE FILES:
;   145          0273  1 !
;   146          0274  1
;   147          0275  1 LIBRARY 'SYS$LIBRARY:STARLET';
;   148          0276  1
;   149          0277  1 undeclare %quote $DESCRIPTOR;
;   150          0278  1 !undeclare %quote RMS$_FNF;
;   151          0279  1 !undeclare %quote RMS$_NORMAL;
;   152          0280  1
;   153          0281  1 LIBRARY 'NXPORT:XPORT';                 ! XPORT Library
;   154          0282  1
;   155        U 0283  1 %IF DSRPLUS %THEN
;   156        U 0284  1 LIBRARY 'REQ:DPLLIB';                   ! DSRPLUS BLISS Library
;   157          0285  1 %ELSE
;   158          0286  1 LIBRARY 'REQ:DSRLIB';                   ! DSR BLISS Library
;   159          0287  1 %FI
;   160          0288  1
;   161          0289  1 !
;   162          0290  1 ! MACROS:
;   163          0291  1 !
;   164          0292  1 ! This macro returns "1" if RUNOFF is running on TOPS-10 or TOPS-20.
;   165          0293  1 ! If we're running on VAX/VMS, the specified return code is used,
;   166          0294  1 ! with bit 28 set to inhibit generation of an error message.
;   167          0295  1 ! On other systems the specified code is returned without modification.
;   168          0296  1 !
;   169          0297  1 MACRO
;   170        M 0298  1     creturn (return_code) =
;   171        M 0299  1         %IF %BLISS (BLISS36) %THEN
;   172        M 0300  1             RETURN 1
;   173        M 0301  1         %ELSE
;   174        M 0302  1             RETURN return_code
;   175        M 0303  1                 + %IF %BLISS(BLISS32) %THEN 1^28 ! Set inhibit message bit
;   176        M 0304  1                   %ELSE 0
;   177        M 0305  1                   %FI
;   178        M 0306  1         %FI
;   179          0307  1                                 %,      !  End of macro CRETURN
;   180          0308  1
;   181        M 0309  1     close_i_o_and_return (delete, return_value) =
;   182        M 0310  1         BEGIN
```

```
;  183        M 0311  1          clh (clh_close_input);
;  184        M 0312  1          ! Close output file unless quick processing (no output file opened).
;  185        M 0313  1          IF  NOT .rno_cmd [rno$v_quick]  THEN
;  186        M 0314  1              ! Delete output file if user requested.
;  187        M 0315  1              BEGIN
;  188        M 0316  1              IF delete
;  189        M 0317  1              THEN  clh (clh_close_del_out)
;  190        M 0318  1              ELSE  clh (clh_close_out);
;  191        M 0319  1              END;
;  192        M 0320  1          creturn (return_value);
;  193        M 0321  1          END
;  194           0322  1                                  %;      !  End of macro CLOSE_I_O_AND_RETURN
;  195           0323  1 !
;  196           0324  1 ! OWN STORAGE:
;  197           0325  1 !
;  198           0326  1 OWN
;  199        U 0327  1 %IF DSRPLUS %THEN
;  200        U 0328  1     init_file_error,           ! Error flag for opening DSR[PLUS]$INIT
;  201           0329  1 %FI
;  202           0330  1     file_error,                ! Error flag for main file processing
;  203           0331  1     tempob : $XPO_IOB (),      ! IOB for output file
;  204           0332  1     tempib : $XPO_IOB ();      ! IOB for primary input file
;  205           0333  1
;  206           0334  1 !
;  207           0335  1 ! EXTERNAL REFERENCES:
;  208           0336  1 !
;  209           0337  1 EXTERNAL
;  210           0338  1     rneiob : REF $XPO_IOB (),
;  211           0339  1     rnaiob : REF $XPO_IOB (),
;  212           0340  1     rnoiob : REF $XPO_IOB (),
;  213           0341  1     tteiob : $XPO_IOB (),
;  214           0342  1     ffname : $STR_DESCRIPTOR (CLASS = dynamic), ! Failing file's name
;  215           0343  1     semcod,                                    ! Secondary Error Message CODe
;  216           0344  1     gca  : gca_definition,
;  217           0345  1     ira  : fixed_string,
;  218           0346  1     irac : irac_definition,
;  219           0347  1     fs01 : fixed_string,
;  220           0348  1     lffctx,                    ! LIB$FIND_FILE context cell
;  221           0349  1     ipftyp,                    ! InPut File TYPe index
;  222           0350  1     ipftop : VECTOR,           ! InPut File Type OPtions list
;  223           0351  1     khar;
;  224           0352  1
;  225           0353  1
;  226           0354  1 EXTERNAL ROUTINE
;  227           0355  1     clh,        doopts,        erm,          erme,
;  228           0356  1     erms,       pus,           rinit,        rterm
;  229           0357  1
;  230           0358  1 %IF %BLISS (BLISS32) %THEN  ,
;  231           0359  1
;  232        U 0360  1 %IF DSRPLUS %THEN
;  233        U 0361  1     setquick,
;  234           0362  1 %FI
;  235           0363  1     LIB$FIND_FILE              ! NEEDED to resolve search strings on VMS!
;  236           0364  1 %FI
;  237           0365  1
;  238        U 0366  1 %IF (%BLISS(BLISS36) AND DSRPLUS)  %THEN ,
;  239        U 0367  1     enapsi : NOVALUE,          ! Enable CTRL/C PSI interrupt.
```

```
;  240        U 0368  1      ttyset : NOVALUE              ! Set (save) terminal characteristics.
;  241          0369  1  %FI                        ;
;  242          0370  1
;  243          0371  1  EXTERNAL LITERAL
;  244          0372  1      rnfcoi,      ! F - Can't open input file "%S"
;  245          0373  1      rnfcoo,      ! F - Can't open output file "%S"
;  246          0374  1      rnffab,      ! W - File aborted
;  247          0375  1      rnfrtl,      ! W - Input record too long: truncated "%S"
;  248          0376  1      rnfure       ! W - Unrecoverable error processing record %C on page %I of input file "%F"
;  249          0377  1
;  250          0378  1  %IF %BLISS(BLISS32) %THEN
;  251          0379  1          ! Special error messages used to detect and signal bad error message
;  252          0380  1          ! pointers:
;  253          0381  1      rnfwfr,      ! Last error defined by the message utility.
;  254          0382  1      last_rnf,    ! Last error defined by RNFERM and MSGTXT.REQ
;  255          0383  1      rnfaaa,      ! Message used to signal the inequality of the previous two
;  256          0384  1
;  257        U 0385  1  %IF dsrplus %THEN
;  258        U 0386  1      rnfoft,      ! File type ".RNO" not found.  Processing file "%F"
;  259          0387  1  %FI
;  260          0388  1  !    RMS$_FNF,  ! VMS RMS error code:  File Not Found
;  261          0389  1  !    RMS$_NORMAL,! VMS RMS return code:  Normal return
;  262          0390  1  %FI
;  263          0391  1      ipftct;      ! Literal defining the lengths of IPFTOP and OPFTOP.
;  264          0392  1
```

```
;  266     0393  1 GLOBAL ROUTINE runoff (rno_cmd) =
;  267     0394  1
;  268     0395  1 !++
;  269     0396  1 ! FUNCTIONAL DESCRIPTION:
;  270     0397  1 !
;  271     0398  1 !       This is the main program of RUNOFF.  The basic processing involves
;  272     0399  1 !       calling CLH to open and close files, and read records that get passed
;  273     0400  1 !       to PUS.  RINIT and RTERM are only marginally related to the formatting
;  274     0401  1 !       processing, being used for initialization and termination.
;  275     0402  1 !
;  276     0403  1 ! FORMAL PARAMETERS:     None
;  277     0404  1 !
;  278     0405  1 ! IMPLICIT INPUTS:       None
;  279     0406  1 !
;  280     0407  1 ! IMPLICIT OUTPUTS:      None
;  281     0408  1 !
;  282     0409  1 ! ROUTINE VALUE:
;  283     0410  1 ! COMPLETION CODES:      None
;  284     0411  1 !
;  285     0412  1 ! SIDE EFFECTS:          None
;  286     0413  1 !--
;  287     0414  1
;  288     0415  2     BEGIN                                       !Start routine RUNOFF
;  289     0416  2
;  290     0417  2     MAP
;  291     0418  2         rno_cmd : REF $rno_cmd;
;  292     0419  2     LOCAL
;  293     0420  2 %IF %BLISS(BLISS32) %THEN
;  294     0421  2         status,                     ! Return value from INPUT_FILE_SCAN routine
;  295     0422  2         input_file_name :           ! Receiving string descriptor
;  296     0423  2                 $STR_DESCRIPTOR (CLASS = dynamic),
;  297     0424  2 %FI
;  298     0425  2         clh_result,                 ! Result of CLH operation
;  299     0426  2         keep_reading,               ! Controls input-reading loop
;  300     0427  2         hold_cref_errcnt,           ! Remembers gca_cref_errcnt over RINIT.
;  301     0428  2         hold_cref_changed;          ! Remembers gca_cref_changed over RINIT
;  302     0429  2
;  303   U 0430  2 %IF DSRPLUS %THEN
;  304   U 0431  2     !+
;  305   U 0432  2     ! Initialize automatic variables
;  306   U 0433  2     !-
;  307   U 0434  2     gca_pass_count = 1;                     ! Current pass over input file
;  308   U 0435  2 %IF %BLISS (BLISS32) %THEN
;  309   U 0436  2     gca_rerun_count = 0;                    ! Number of times input file must be reprocessed
;  310   U 0437  2     gca_cref_errcnt = 0;                    ! No cross-references have been unresolved yet.
;  311   U 0438  2     gca_cref_changed = false;               ! No cross-references have changed.
;  312   U 0439  2
;  313   U 0440  2     WHILE true DO
;  314   U 0441  2         BEGIN                                       ! Start of automatic-processing loop
;  315   U 0442  2         !
;  316   U 0443  2         ! Reprocess the input file until exit condition is met
;  317   U 0444  2         !
;  318   U 0445  2         hold_cref_errcnt = .gca_cref_errcnt;
;  319   U 0446  2         hold_cref_changed = .gca_cref_changed;
;  320     0447  2 %FI  %FI
;  321     0448  2     rneiob = tempib;
;  322     0449  2     rnaiob = tempob;
```

```
;  323          0450  2
;  324          0451  2
;  325          0452  2      $XPO_IOB_INIT (IOB = .rnaiob, FILE_SPEC = rno_cmd [rno$t_output]);
;  326          0453  2
;  327          0454  2      rinit ();                                   ! Initialize most everything.
;  328          0455  2
;  329          0456  2  %IF %BLISS(BLISS32) %THEN
;  330          0457  2      !+
;  331          0458  2      ! Now we must see if the VMS message utility using DSRMSG.MSG as input has
;  332          0459  2      ! defined the same number of messages as RNFERM.BLI has put into the message
;  333          0460  2      ! vector using MSGTXT.REQ as input.  This is done by verifying that a global
;  334          0461  2      ! defined in RNFERM.BLI (LAST_RNF) is equal to the last message code defined
;  335          0462  2      ! by the message utility (RNFWFR).  If not, signal the FATAL error and exit!
;  336          0463  2      !-
;  337          0464  2      IF rnfwfr NEQ last_rnf
;  338          0465  2      THEN
;  339          0466  3          BEGIN
;  340          0467  3          erm (rnfaaa, 0, 0);
;  341          0468  3          creturn (4)
;  342          0469  2          END;
;  343          0470  2  %FI
;  344          0471  2
;  345          0472  2  %IF %BLISS(BLISS32) %THEN
;  346        P 0473  2      $STR_DESC_INIT( DESCRIPTOR = input_file_name
;  347          0474  2                     ,CLASS = DYNAMIC);
;  348          0475  2                                                  ! Locate an input file
;  349          0476  2      IF (status = input_file_scan (.rno_cmd, input_file_name)) EQL 4
;  350          0477  2          THEN creturn(.status)
;  351          0478  2      ELSE
;  352          0479  2          $XPO_IOB_INIT (IOB = .rneiob, FILE_SPEC = input_file_name);
;  353        U 0480  2  %ELSE
;  354        U 0481  2      $XPO_IOB_INIT (IOB = .rneiob, FILE_SPEC = rno_cmd [rno$t_input]);
;  355          0482  2  %FI
;  356          0483  2      !
;  357          0484  2      ! Now proceed with normal processing.
;  358          0485  2      !
;  359          0486  2
;  360          0487  2      clh_result = clh (clh_open_input);  ! Try to open input file.
;  361          0488  2
;  362          0489  2      IF  .clh_result NEQ clh_normal
;  363          0490  2      THEN
;  364          0491  3          BEGIN                                   ! Start can't open input
;  365          0492  3          LOCAL
;  366          0493  3              spec_length,
;  367          0494  3              spec_ptr;
;  368          0495  3
;  369          0496  3          spec_length = .ffname [str$h_length];
;  370          0497  3          spec_ptr    = .ffname [str$a_pointer];
;  371          0498  3          !
;  372          0499  3          erme (rnfcoi, .spec_ptr, .spec_length, .semcod);
;  373          0500  3
;  374          0501  3          creturn (4)
;  375          0502  3          END                                     ! End can't open input
;  376          0503  2      ELSE
;  377          0504  2      ! Did user specify quick processing? If so, don't open an output file.
;  378          0505  2          IF  .rno_cmd [rno$v_quick]
;  379          0506  2          THEN
```

```
;  380     0507  2              clh_result = clh_normal
;  381     0508  2          ELSE
;  382     0509  2              BEGIN
;  383     0510  2              !
;  384     0511  2              ! Pick up command-line information about /DEVICE now,
;  385     0512  2              ! so that output file defaults can be set up properly.
;  386     0513  2              !
;  387     0514  2              gca_op_dev  = .rno_cmd [rno$v_4_out_format]; ! Pick up device type.
;  388     0515  2              gca_s_output = .rno_cmd [rno$v_s_output];    ! See if user said /OUTPUT=file.
;  389     0516  2              clh_result   = clh (clh_open_out);  ! Open output file.
;  390     0517  2
;  391   U 0518  2      %IF (%BLISS(BLISS36) AND DSRPLUS)  %THEN
;  392   U 0519  2              !+
;  393   U 0520  2              ! The following call modifies the Control Character Output
;  394   U 0521  2              ! Control (CCOC) words to allow escape sequences to be sent to
;  395   U 0522  2              ! the terminal without translation, and modifies the JFN mode
;  396   U 0523  2              ! word to prevent TOPS-20 from wrapping output lines at the
;  397   U 0524  2              ! terminal width.
;  398   U 0525  2              !-
;  399   U 0526  2              IF  (.gca_op_dev EQL op_dev_vt100)  ! User said /DEC=VT100, and did
;  400   U 0527  2                  AND NOT .gca_s_output           ! not say /OUTPUT=name.
;  401   U 0528  2              THEN
;  402   U 0529  2                  ttyset ();                      ! Set terminal characteristics.
;  403   U 0530  2              !
;  404   U 0531  2              ! Enable the PSI CTRL/C interrupt.
;  405   U 0532  2              !
;  406   U 0533  2              enapsi ();
;  407     0534  2      %FI
;  408     0535  2              END;
;  409     0536  2
;  410     0537  2          IF  .clh_result NEQ clh_normal
;  411     0538  2          THEN
;  412     0539  3              BEGIN                                   ! Start can't open output
;  413     0540  3              LOCAL
;  414     0541  3                  spec_length,
;  415     0542  3                  spec_ptr;
;  416     0543  3
;  417     0544  3              spec_length = .ffname [STR$H_LENGTH];
;  418     0545  3              spec_ptr    = .ffname [STR$A_POINTER];
;  419     0546  3              !
;  420     0547  3              ! Can't open output file.
;  421     0548  3              !
;  422     0549  3              erme (rnfcoo, .spec_ptr, .spec_length, .semcod);
;  423     0550  3              clh (clh_close_input);
;  424     0551  3              creturn (4)
;  425     0552  3              END                                     ! End can't open output
;  426     0553  2          ELSE
;  427     0554  3              BEGIN                                   ! Start normal file processing
;  428   U 0555  3      %IF FLIP  %THEN
;  429   U 0556  3              IF  (.gca_op_dev EQL op_dev_flip)       ! User said /DEVICE=FLIP
;  430   U 0557  3              THEN
;  431   U 0558  3                  BEGIN
;  432   U 0559  3                  LOCAL
;  433   U 0560  3                      init_record : $flip_init,
;  434   U 0561  3                      tcxini_record : $flip_tcxini;
;  435   U 0562  3
;  436   U 0563  3              ! Write FLIP initialization record.
```

```
;  437        U 0564  3          !
;  438        U 0565  3              init_record [init_code]     = flip$k_init;
;  439        U 0566  3              init_record [init_contents] = .rno_cmd [rno$v_intermediate];
;  440        U 0567  3              init_record [init_level]    = flip$k_version;
;  441        U 0568  3              rno_cmd [rno$v_intermediate]    = 1;
;  442        U 0569  3              $XPO_PUT (  IOB    = .rnoiob,
;  443        U 0570  3                          STRING = (flip$k_init_size,
;  444        U 0571  3                          CH$PTR (init_record)));
;  445        U 0572  3
;  446        U 0573  3          ! Write FLIP index-information initialization record.
;  447        U 0574  3          !
;  448        U 0575  3              tcxini_record[tcxini_code] = flip$k_tcxini;
;  449        U 0576  3              tcxini_record[tcxini_header] = index_format;
;  450        U 0577  3              $XPO_PUT (  IOB     = .rnoiob,
;  451        U 0578  3                          STRING  = (flip$k_tcxini_size,
;  452        U 0579  3                          CH$PTR (tcxini_record)));
;  453        U 0580  3              END;
;  454          0581  %FI
;  455          0582
;  456        U 0583  %IF (%BLISS (BLISS32) AND DSRPLUS) %THEN
;  457        U 0584                 ! This logic makes sure we don't create a .MEM file for an
;  458        U 0585                 ! intermediate run of /AUTO -- if the reason we need one more
;  459        U 0586                 ! run is because a cross-reference changed.
;  460        U 0587                 !
;  461        U 0588                 IF  .rno_cmd [rno$v_automatic]! gca_black_box  ! Are we doing /AUTO?
;  462        U 0589                     AND
;  463        U 0590                     ((.hold_cref_changed EQL true) OR
;  464        U 0591                     ((.hold_cref_errcnt GTR 0) AND (.gca_pass_count EQL 2)))  ! One changed or was unresolved on
;  465        U 0592                     AND .gca_pass_count NEQ 4 ! Last chance, oscillating cref's.
;  466        U 0593                 THEN
;  467        U 0594                     setquick (true);
;  468        U 0595                 !
;  469        U 0596                 ! The user may have had some forward crefs.
;  470        U 0597                 ! We give him the benefit of the doubt and run plus again.
;  471        U 0598                 !
;  472        U 0599                 IF (.hold_cref_errcnt GTR 0) AND (.gca_pass_count EQL 2)
;  473        U 0600                 THEN
;  474        U 0601                     gca_rerun_count = .gca_rerun_count + 1;
;  475          0602  %FI
;  476          0603                 IF  NOT doopts (.rno_cmd)
;  477          0604                 THEN
;  478          0605                     !
;  479          0606                     ! Close and delete output file; return 4.
;  480          0607  4                  close_i_o_and_return (true, 4)
;  481          0608  3              ELSE
;  482          0609  4                  BEGIN
;  483          0610  4                  !+
;  484          0611  4                  ! If the output file is the terminal, don't redirect error messages.  Otherwise, direct messages
;  485          0612  4                  ! specified on the /MESSAGES switch.  The effect is that the user can determine where error mess
;  486          0613  4                  ! go only if the output file is not the terminal; i.e., you cannot suppress error messages total
;  487          0614  4                  ! but you can prevent them from going into the output file.
;  488          0615  4                  !-
;  489          0616  5                  IF  NOT (.rnoiob [iob$v_terminal] AND
;  490          0617  5                      .tteiob [iob$v_terminal])
;  491          0618  4                  THEN
;  492          0619  4                      gca_err_dir =.gca_cmd_msg;
;  493        U 0620  4 %IF DSRPLUS %THEN
```

```
;  494        U 0621  4              ! Attempt to open initialization files DSR$INIT and/or
;  495        U 0622  4              ! DSRPLUS$INIT.
;  496        U 0623  4              !
;  497        U 0624  4              init_files (.rno_cmd);
;  498          0625  4 %FI
;  499          0626  3              END;
;  500          0627  3
;  501          0628  3 !+
;  502          0629  3 ! Here is RUNOFF's main processing loop.
;  503          0630  3 !-
;  504          0631  3              keep_reading = true;
;  505          0632  3              file_error = false;
;  506          0633  3
;  507          0634  3              WHILE .keep_reading DO
;  508          0635  3                  keep_reading = input_loop (file_error);
;  509          0636  3              !
;  510          0637  3              ! Check status of file errors, and finish up accordingly.
;  511          0638  3              !
;  512          0639  3              IF NOT .file_error
;  513          0640  3              THEN
;  514          0641  4                  BEGIN                                   ! Start normal termination
;  515          0642  4
;  516          0643  4                  rterm ();
;  517          0644  4                  !
;  518        U 0645  4 %IF (%BLISS (BLISS32) AND DSRPLUS) %THEN
;  519        U 0646  4                  IF .gca_pass_count LEQ .gca_rerun_count
;  520        U 0647  4                  THEN
;  521        U 0648  4                      BEGIN
;  522        U 0649  4                      ! Doing /AUTOMATIC processing and encountered either cross
;  523        U 0650  4                      ! references or a .DO CONTENTS that required more than one
;  524        U 0651  4                      ! pass over the input file.
;  525        U 0652  4                      !
;  526        U 0653  4                      ! Quit now if we have processed the input stream 4 times.
;  527        U 0654  4                      ! In this case, we assume that an oscillating cross-refer-
;  528        U 0655  4                      ! ence makes it impossible for us to resolve the file
;  529        U 0656  4                      ! automatically.
;  530        U 0657  4                      !
;  531        U 0658  4                      IF .gca_pass_count EQL 4
;  532        U 0659  4                      THEN
;  533        U 0660  4                          close_i_o_and_return (false, 2)
;  534        U 0661  4                      ELSE
;  535        U 0662  4                          ! Bump the pass counter and close the input file (the
;  536        U 0663  4                          ! output file has already been closed).
;  537        U 0664  4                          !
;  538        U 0665  4                          BEGIN
;  539        U 0666  4                          gca_pass_count = .gca_pass_count + 1;
;  540        U 0667  4                          clh (clh_close_input);
;  541        U 0668  4                          END;
;  542        U 0669  4                      END
;  543        U 0670  4                  ELSE
;  544        U 0671  4                      ! Either /AUTOMATIC was not asserted or all required passes
;  545        U 0672  4                      ! over the input file are complete. In any case, close the
;  546        U 0673  4                      ! input and output files and return the appropriate status.
;  547          0674  4 %FI
;  548          0675  4                  ! Close output file (do not delete); return either success
;  549          0676  4                  ! or failure.
;  550          0677  5                  BEGIN
```

```
;  551          0678  5                 IF .gca_fehler
;  552          0679  5                 THEN
;  553          0680  6                     close_i_o_and_return (false, 2)
;  554          0681  5                 ELSE
;  555          0682  5                     close_i_o_and_return (false, 1);
;  556          0683  4                 END;
;  557          0684  4
;  558          0685  4             END                                 ! End normal termination
;  559          0686  3         ELSE
;  560          0687  3             ! Close (don't delete) output file; return 4 (fatal condition).
;  561          0688  3             close_i_o_and_return (false, 4);
;  562          0689  3
;  563          0690  2         END;                                    ! End normal file processing
;  564          0691  2
;  565        U 0692  2 %IF (%BLISS (BLISS32) AND DSRPLUS) %THEN
;  566        U 0693  2         END;                                    ! End of automatic-processing loop
;  567        U 0694  2     true        !avoid compiler complaint
;  568          0695  2 %FI
;  569          0696  1     END;                                  ! End of RUNOFF


                                                        .TITLE  RUNOFF Overall DSR process control
                                                        .IDENT  \V04-000\

                                                        .PSECT  $OWN$,NOEXE,2

                                                00000 FILE_ERROR:
                                                                .BLKB   4
                                                00004 TEMPOB:   .BLKB   244
                                                000F8 TEMPIB:   .BLKB   244

                                                        .EXTRN  RNEIOB, RNAIOB, RNOIOB
                                                        .EXTRN  TTEIOB, FFNAME, SEMCOD
                                                        .EXTRN  GCA, IRA, IRAC, FS01
                                                        .EXTRN  LFFCTX, IPFTYP, IPFTOP
                                                        .EXTRN  KHAR, CLH, DOOPTS
                                                        .EXTRN  ERM, ERME, ERMS
                                                        .EXTRN  PUS, RINIT, RTERM
                                                        .EXTRN  LIB$FIND_FILE, RNFCOI
                                                        .EXTRN  RNFCOO, RNFFAB, RNFRTL
                                                        .EXTRN  RNFURE, RNFWFR, LAST_RNF
                                                        .EXTRN  RNFAAA, IPFTCT

                                                        .PSECT  $CODE$,NOWRT,2

                                            0FFC 00000          .ENTRY  RUNOFF, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-  ; 0393
                                                                        R11
                                5B      0000'   CF  9E 00002            MOVAB   FILE_ERROR, R11                     ;
                                5A  00000000G   00  9E 00007            MOVAB   FFNAME, R10                         ;
                                59  00000000G   00  9E 0000E            MOVAB   GCA+208, R9                         ;
                                58  00000000G   00  9E 00015            MOVAB   CLH, R8                             ;
                                5E              08  C2 0001C            SUBL2   #8, SP                              ;
                    00000000G   00      00F8    CB  9E 0001F            MOVAB   TEMPIB, RNEIOB                      ; 0448
                    00000000G   00        04    AB  9E 00028            MOVAB   TEMPOB, RNAIOB                      ; 0449
                                50  00000000G   00  D0 00030            MOVL    RNAIOB, R6                          ; 0452
                                57          04  AC  D0 00037            MOVL    RNO_CMD, R7                         ;
00F4    8F          00          6E              00  2C 0003B            MOVC5   #0, (SP), #0, #244, (R6)            ;
```

```
                                        66       00042
                      66 0301003D       8F   D0  00043        MOVL     #50397245, (R6)                      ;
              04      A6       08       A7   9E  0004A        MOVAB    8(R7), 4(R6)                         ;
              1E      A6     020E       8F   B0  0004F        MOVW     #526, 30(R6)                         ;
       00000000G      00                00   FB  00055        CALLS    #0, RINIT                            ; 0454
       00000000G      8F 00000000G      8F   D1  0005C        CMPL     #RNFWFR, #LAST_RNF                   ; 0464
                                        11   13  00067        BEQL     1$                                   ;
                                        7E   7C  00069        CLRQ     -(SP)                                ; 0467
                         00000000G      8F   DD  0006B        PUSHL    #RNFAAA                              ;
       00000000G      00                03   FB  00071        CALLS    #3, ERM                              ;
                                        69   11  00078        BRB      3$                                   ; 0468
                      6E 020E0000       8F   D0  0007A 1$:    MOVL     #34471936, $STR$DESC                 ; 0474
                               04       AE   D4  00081        CLRL     $STR$DESC+4                          ;
                             4080       8F   BB  00084        PUSHR    #^M<R7,SP>                           ; 0476
           0000V      CF                02   FB  00088        CALLS    #2, INPUT_FILE_SCAN                  ;
                      04                50   D1  0008D        CMPL     STATUS, #4                           ;
                                        08   12  00090        BNEQ     2$                                   ;
                      50 10000000       E0   9E  00092        MOVAB    268435456(R0), R0                    ; 0477
                                             04  00099        RET                                          ;
                      56 00000000G      00   D0  0009A 2$:    MOVL     RNEIOB, R6                           ; 0479
   00F4    8F          00               6E   00  2C  000A1    MOVC5    #0, (SP), #0, #244, (R6)             ;
                                        66       000A8
                      66 0301003D       8F   D0  000A9        MOVL     #50397245, (R6)                      ;
              04      A6                6E   9E  000B0        MOVAB    $IOB$FILE_SPEC, 4(R6)                ;
              1E      A6     020E       8F   B0  000B4        MOVW     #526, 30(R6)                         ;
                                        01   DD  000BA        PUSHL    #1                                   ; 0487
                      68                01   FB  000BC        CALLS    #1, CLH                              ;
                      52                50   D0  000BF        MOVL     R0, CLH_RESULT                       ;
                      01                52   D1  000C2        CMPL     CLH_RESULT, #1                       ; 0489
                                        1F   13  000C5        BEQL     4$                                   ;
                      51                6A   3C  000C7        MOVZWL   FFNAME, SPEC_LENGTH                  ; 0496
                      50       04       AA   D0  000CA        MOVL     FFNAME+4, SPEC_PTR                   ; 0497
                         00000000G      00   DD  000CE        PUSHL    SEMCOD                               ; 0499
                                        03   BB  000D4        PUSHR    #^M<R0,R1>                           ;
                         00000000G      8F   DD  000D6        PUSHL    #RNFCOI                              ;
       00000000G      00                04   FB  000DC        CALLS    #4, ERME                             ;
                                      00D8   31  000E3 3$:    BRW      19$                                  ; 0501
              05      50                A7   E9  000E6 4$:    BLBC     80(R7), 5$                           ; 0505
                      52                01   D0  000EA        MOVL     #1, CLH_RESULT                       ; 0507
                                        1E   11  000ED        BRB      6$                                   ;
   50       4E        A7       04       06   EF  000EF 5$:    EXTZV    #6, #4, 78(R7), R0                   ; 0514
   69                 04       04       50   F0  000F5        INSV     R0, #4, #4, GCA+208                  ;
   50       4F        A7       01       03   EF  000FA        EXTZV    #3, #1, 79(R7), R0                   ; 0515
   69                 01       03       50   F0  00100        INSV     R0, #3, #1, GCA+208                  ;
                                        02   DD  00105        PUSHL    #2                                   ; 0516
                      68                01   FB  00107        CALLS    #1, CLH                              ;
                      52                50   D0  0010A        MOVL     R0, CLH_RESULT                       ;
                      01                52   D1  0010D 6$:    CMPL     CLH_RESULT, #1                       ; 0537
                                        20   13  00110        BEQL     7$                                   ;
                      51                6A   3C  00112        MOVZWL   FFNAME, SPEC_LENGTH                  ; 0544
                      50       04       AA   D0  00115        MOVL     FFNAME+4, SPEC_PTR                   ; 0545
                         00000000G      00   DD  00119        PUSHL    SEMCOD                               ; 0549
                                        03   BB  0011F        PUSHR    #^M<R0,R1>                           ;
                         00000000G      8F   DD  00121        PUSHL    #RNFCOO                              ;
       00000000G      00                04   FB  00127        CALLS    #4, ERME                             ;
                                        03   DD  0012E        PUSHL    #3                                   ; 0550
                                        17   11  00130        BRB      8$                                   ;
```

```
                                        57  DD 00132 7$:     PUSHL   R7                                      ; 0603
                 00000000G              00  01  FB 00134     CALLS   #1, DOOPTS
                                        0D  50  E8 00138     BLBS    R0, 9$
                                            03  DD 0013E     PUSHL   #3                                      ; 0607
                                        68  01  FB 00140     CALLS   #1, CLH
                                        77      50  A7  E8 00143 BLBS  80(R7), 19$
                                            0C  DD 00147     PUSHL   #12
                                        70  11 00149 8$:     BRB     18$
                                        50 00000000G  00  D0 0014B 9$: MOVL  RNOIOB, R0                      ; 0616
                          08        32  A0  04  E1 00152     BBC     #4, 50(R0), 10$
                          06 00000000G  00  04  E0 00157     BBS     #4, TTEIOB+50, 11$                     ; 0617
                                    F8  A9      FF64  C9  D0 0015F 10$: MOVL  GCA+52, GCA+200               ; 0619
                                        50  01  D0 00165 11$: MOVL  #1, KEEP_READING                        ; 0631
                                            6B  D4 00168     CLRL    FILE_ERROR                             ; 0632
                                        09  50  E9 0016A 12$: BLBC  KEEP_READING, 13$                       ; 0634
                                            5B  DD 0016D     PUSHL   R11                                    ; 0635
                                  0000V CF  01  FB 0016F     CALLS   #1, INPUT_LOOP
                                        F4  11 00174     BRB     12$
                                        37  6B  E8 00176 13$: BLBS  FILE_ERROR, 17$                         ; 0639
                 00000000G              00  00  FB 00179     CALLS   #0, RTERM                              ; 0643
                                        16      F0  A9  E9 00180 BLBC  GCA+192, 15$                         ; 0678
                                            03  DD 00184     PUSHL   #3                                     ; 0680
                                        68  01  FB 00186     CALLS   #1, CLH
                                        05      50  A7  E8 00189 BLBS  80(R7), 14$
                                            04  DD 0018D     PUSHL   #4
                                        68  01  FB 0018F     CALLS   #1, CLH
                                        50 10000002  8F  D0 00192 14$: MOVL  #268435458, R0                  ; 0682
                                                 04 00199     RET
                                            03  DD 0019A 15$: PUSHL   #3
                                    68  01  FB 0019C     CALLS   #1, CLH
                                    05      50  A7  E8 0019F BLBS  80(R7), 16$
                                            04  DD 001A3     PUSHL   #4
                                    68  01  FB 001A5     CALLS   #1, CLH
                                    50 10000001  8F  D0 001A8 16$: MOVL  #268435457, R0
                                                 04 001AF     RET                                            ; 0688
                                            03  DD 001B0 17$: PUSHL   #3
                                    68  01  FB 001B2     CALLS   #1, CLH
                                    05      50  A7  E8 001B5 BLBS  80(R7), 19$
                                            04  DD 001B9     PUSHL   #4
                                    68  01  FB 001BB 18$: CALLS   #1, CLH
                                    50 10000004  8F  D0 001BE 19$: MOVL  #268435460, R0
                                                 04 001C5     RET                                            ; 0696

; Routine Size:  454 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  571     0697  1 GLOBAL ROUTINE grab_resultant (
;  572     0698  1                  function_code          ! For failure action routine
;  573     0699  1                 ,primary_code           ! 1st completion code
;  574     0700  1                 ,secondary_code         ! 2nd completion code
;  575     0701  1                 ,iob : REF $XPO_IOB ()  ! Address of IOB
;  576     0702  1                                 ) =
;  577     0703  1 !++
;  578     0704  1 ! FUNCTIONAL DESCRIPTION:
;  579     0705  1 !
;  580     0706  1 !       Moves resultant filename string from the IOB to FFNAME so it
;  581     0707  1 !       can be picked up and used when the error is signaled.
;  582     0708  1 !
;  583     0709  1 ! FORMAL PARAMETERS:
;  584     0710  1 !
;  585     0711  1 !       As defined by XPORT.  See above ROUTINE declaration for the list and
;  586     0712  1 !       meanings of the current formal parameters.
;  587     0713  1 !
;  588     0714  1 ! IMPLICIT INPUTS:      None
;  589     0715  1 !
;  590     0716  1 ! IMPLICIT OUTPUTS:
;  591     0717  1 !
;  592     0718  1 !       FFNAME  String descriptor defined in GLBDAT.BLI that points to the
;  593     0719  1 !               resultant filename string from the IOB pointed to by the
;  594     0720  1 !               fourth parameter passed to this routine.
;  595     0721  1 !
;  596     0722  1 ! ROUTINE VALUE:
;  597     0723  1 ! COMPLETION CODES:
;  598     0724  1 !
;  599     0725  1 !       The PRIMARY_CODE passed to this routine will also be used as its
;  600     0726  1 !       completion code.
;  601     0727  1 !
;  602     0728  1 ! SIDE EFFECTS:         None
;  603     0729  1 !--
;  604     0730  1
;  605     0731  2     BEGIN                                       ! Start grab_resultant
;  606     0732  2         $STR_DESC_INIT (DESCRIPTOR = ffname, CLASS = dynamic);
;  607     0733  2         !
;  608     0734  2         ! Pick of the name and length of the file spec.
;  609     0735  2         !
;  610     0736  2         $STR_COPY ( STRING = iob[iob$t_resultant] ,TARGET = ffname );
;  611     0737  2         semcod = .primary_code;             ! Pick up and save the error code
;  612     0738  2                                             ! explaining why this file couldn't
;  613     0739  2                                             ! be opened.
;  614     0740  2         .primary_code
;  615     0741  1     END;                                    ! End of GRAB_RESULTANT


                                                    .EXTRN  XST$COPY, STR$FAILURE

                                      0004 00000    .ENTRY  GRAB_RESULTANT, Save R2                 ; 0697
                 52 00000000G   00     9E 00002    MOVAB   $STR$DESC, R2                           ;
                 62 020E0000    8F     D0 00009    MOVL    #34471936, $STR$DESC                    ; 0732
                       04       A2     D4 00010    CLRL    $STR$DESC+4                             ;
          50       10   AC       1C     C1 00013    ADDL3   #28, IOB, R0                            ; 0736
                    00000000G   EF     9F 00018    PUSHAB  STR$FAILURE                             ;
                                7E     D4 0001E    CLRL    -(SP)                                   ;
```

```
                                    05  BB 00020        PUSHR   #^M<R0,R2>                                ;
                                    7E  D4 00022        CLRL    -(SP)                                    ;
                    00000000G   EF  05  FB 00024        CALLS   #5, XST$COPY                             ;
                    00000000G   00  08  AC  D0 0002B    MOVL    PRIMARY_CODE, SEMCOD                     ; 0737
                                50  08  AC  D0 00033    MOVL    PRIMARY_CODE, R0                         ; 0741
                                        04  00037      RET                                              ;

; Routine Size:  56 bytes,    Routine Base:  $CODE$ + 01C6

;  616        U 0742  1 %IF DSRPLUS  %THEN
```

```
;  618     U 0743  1 ROUTINE init_files (rno_cmd) : NOVALUE =
;  619     U 0744  1
;  620     U 0745  1 !++
;  621     U 0746  1 ! FUNCTIONAL DESCRIPTION:
;  622     U 0747  1 !
;  623     U 0748  1 !       INIT_FILES processes either or both of the initialization files
;  624     U 0749  1 !       that may be present -- DSR$INIT and DSRPLUS$INIT. It attempts to
;  625     U 0750  1 !       open the files in order and, if found, takes input from them before
;  626     U 0751  1 !       the main input file is read.
;  627     U 0752  1 !
;  628     U 0753  1 ! FORMAL PARAMETERS:
;  629     U 0754  1 !
;  630     U 0755  1 !       RNO_CMD is the address of the command-line block, passed down from
;  631     U 0756  1 !       RUNOFF.
;  632     U 0757  1 !
;  633     U 0758  1 ! IMPLICIT INPUTS:      None
;  634     U 0759  1 !
;  635     U 0760  1 ! IMPLICIT OUTPUTS:     None
;  636     U 0761  1 !
;  637     U 0762  1 ! ROUTINE VALUE:
;  638     U 0763  1 ! COMPLETION CODES:     None
;  639     U 0764  1 !
;  640     U 0765  1 ! SIDE EFFECTS:         None
;  641     U 0766  1 !--
;  642     U 0767  1
;  643     U 0768  1     BEGIN                                         ! Start of init_files
;  644     U 0769  1     MAP
;  645     U 0770  1         rno_cmd : REF $rno_cmd;
;  646     U 0771  1     LOCAL
;  647     U 0772  1         clh_result,
;  648     U 0773  1         dsrplus_init,
;  649     U 0774  1         keep_reading,
;  650     U 0775  1         ptr;
;  651     U 0776  1
;  652     U 0777  1     dsrplus_init = false;
;  653     U 0778  1     !
;  654     U 0779  1     ! Save the original input file.
;  655     U 0780  1     !
;  656     U 0781  1     clh (clh_push);
;  657     U 0782  1     !
;  658     U 0783  1     ! Try to open logical file DSRPLUS$INIT. CLH expects to find the filespec in fs01.
;  659     U 0784  1     !
;  660     U 0785  1     ptr = CH$PTR (UPLIT (%IF %BLISS (BLISS32) %THEN 'DSRPLUS$INIT'
;  661     U 0786  1                          %ELSE                      'DSRPLUS.INIT' %FI ));
;  662     U 0787  1     fs_init  (fs01);
;  663     U 0788  1     !
;  664     U 0789  1     INCR i FROM 1 TO 12 DO
;  665     U 0790  1         fs_wchar (fs01, CH$RCHAR_A (ptr));
;  666     U 0791  1     !
;  667     U 0792  1     clh_result = clh (clh_open_init);
;  668     U 0793  1     !
;  669     U 0794  1     ! If CLH can open file, save it and try to get next one.
;  670     U 0795  1     !
;  671     U 0796  1     IF  .clh_result EQL clh_normal
;  672     U 0797  1     THEN
;  673     U 0798  1         BEGIN
;  674     U 0799  1         clh (clh_push);
```

```
;  675     U 0800  1          dsrplus_init = true;
;  676     U 0801  1          END;
;  677     U 0802  1      !
;  678     U 0803  1      ! Try to open logical file DSR$INIT. CLH expects to find the filespec in fs01.
;  679     U 0804  1      !
;  680     U 0805  1      ptr = CH$PTR (UPLIT (%IF %BLISS (BLISS32) %THEN 'DSR$INIT'
;  681     U 0806  1                           %ELSE                      'RUNOFF.INIT' %FI ));
;  682     U 0807  1      !
;  683     U 0808  1      fs_init  (fs01);
;  684     U 0809  1      INCR i FROM 1 TO 8 DO
;  685     U 0810  1          fs_wchar (fs01, CH$RCHAR_A (ptr));
;  686     U 0811  1      !
;  687     U 0812  1      clh_result = clh (clh_open_init);
;  688     U 0813  1      !
;  689     U 0814  1      ! If CLH can open file, start reading from it. Otherwise, try next one.
;  690     U 0815  1      !
;  691     U 0816  1      IF  .clh_result EQL clh_normal
;  692     U 0817  1      THEN
;  693     U 0818  1          BEGIN                                  ! Start processing DSR$INIT file.
;  694     U 0819  1
;  695     U 0820  1          keep_reading = true;
;  696     U 0821  1          init_file_error = false;
;  697     U 0822  1
;  698     U 0823  1          WHILE .keep_reading DO
;  699     U 0824  1              keep_reading = input_loop (init_file_error);
;  700     U 0825  1          !
;  701     U 0826  1          ! Close initialization file. (Same as closing .REQUIRE file.)
;  702     U 0827  1          !
;  703     U 0828  1          clh (clh_close_req);
;  704     U 0829  1          !
;  705     U 0830  1          ! If an error occurred, recover and close the original source file
;  706     U 0831  1          ! and the output "file".
;  707     U 0832  1          !
;  708     U 0833  1          IF  .init_file_error
;  709     U 0834  1          THEN
;  710     U 0835  1              BEGIN                              ! Start pop DSRPLUS$INIT
;  711     U 0836  1              !
;  712     U 0837  1              ! Pop DSRPLUS$INIT if it was pushed.
;  713     U 0838  1              !
;  714     U 0839  1              IF  .dsrplus_init THEN
;  715     U 0840  1                  clh (clh_pop);
;  716     U 0841  1              !
;  717     U 0842  1              clh (clh_pop);                 ! Recover and close original source file.
;  718     U 0843  1              !
;  719     U 0844  1              ! Close (don't delete) output file; return 4.
;  720     U 0845  1              close_i_o_and_return (false, 4);
;  721     U 0846  1              !
;  722     U 0847  1              END;                               ! End pop DSRPLUS$INIT
;  723     U 0848  1
;  724     U 0849  1          END;                                   ! End processing DSR$INIT file
;  725     U 0850  1
;  726     U 0851  1      IF  .dsrplus_init
;  727     U 0852  1      THEN
;  728     U 0853  1          BEGIN                                  ! Start processing DSRPLUS$INIT file
;  729     U 0854  1          clh (clh_pop);
;  730     U 0855  1
;  731     U 0856  1          keep_reading = true;
```

```
;  732          U 0857  1          init_file_error = false;
;  733          U 0858  1
;  734          U 0859  1          WHILE .keep_reading DO
;  735          U 0860  1              keep_reading = input_loop (init_file_error);
;  736          U 0861  1          !
;  737          U 0862  1          ! Close initialization file.
;  738          U 0863  1          !
;  739          U 0864  1          clh (clh_close_req);
;  740          U 0865  1          !
;  741          U 0866  1          ! If an error occurred, recover and close original source file
;  742          U 0867  1          ! and output destination.
;  743          U 0868  1          !
;  744          U 0869  1          IF  .init_file_error
;  745          U 0870  1          THEN
;  746          U 0871  1              BEGIN
;  747          U 0872  1              !
;  748          U 0873  1              clh (clh_pop);                  ! Recover and close original source file.
;  749          U 0874  1              !
;  750          U 0875  1              ! Close (don't delete) output file; return 4.
;  751          U 0876  1              close_i_o_and_return (false, 4);
;  752          U 0877  1              !
;  753          U 0878  1              END;
;  754          U 0879  1
;  755          U 0880  1          END;                                    ! End processing DSRPLUS$INIT file
;  756          U 0881  1      !
;  757          U 0882  1      ! Recover original source file.
;  758          U 0883  1      !
;  759          U 0884  1      clh (clh_pop);
;  760          U 0885  1
;  761          U 0886  1      END;                                    ! End of INIT_FILES
;  762            0887  1 %FI
```

```
;  764     0888  1 ROUTINE input_loop (error_condition) =
;  765     0889  1
;  766     0890  1 !++
;  767     0891  1 ! FUNCTIONAL DESCRIPTION:
;  768     0892  1 !
;  769     0893  1 !       INPUT_LOOP reads and processes a single record from the input file.
;  770     0894  1 !
;  771     0895  1 ! FORMAL PARAMETERS:
;  772     0896  1 !
;  773     0897  1 !       ERROR_CONDITION is set to TRUE before return if a file-processing
;  774     0898  1 !       error occurred.
;  775     0899  1 !
;  776     0900  1 ! IMPLICIT INPUTS:      None
;  777     0901  1 !
;  778     0902  1 ! IMPLICIT OUTPUTS:     None
;  779     0903  1 !
;  780     0904  1 ! ROUTINE VALUE:
;  781     0905  1 ! COMPLETION CODES:
;  782     0906  1 !
;  783     0907  1 !       The routine returns TRUE while there is more input to read; it
;  784     0908  1 !       returns FALSE when end-of-file is reached, and also when a file
;  785     0909  1 !       error occurs. (in this latter case, the formal ERROR_CONDITION is
;  786     0910  1 !       also set.)
;  787     0911  1 !
;  788     0912  1 ! SIDE EFFECTS:         None
;  789     0913  1 !--
;  790     0914  1
;  791     0915  2     BEGIN                                       ! Start of input_loop
;  792     0916  2     LOCAL
;  793     0917  2         clh_result;
;  794     0918  2
;  795     0919  2     clh_result = clh (clh_read_input);
;  796     0920  2
;  797     0921  2     SELECT .clh_result OF
;  798     0922  2         SET
;  799     0923  2
;  800     0924  2         [clh_normal] :          ! Something was read successfully.
;  801     0925  2             !
;  802     0926  3             BEGIN
;  803     0927  3             kcns ();
;  804     0928  3             pus  (true);
;  805     0929  2             END;
;  806     0930  2
;  807     0931  2         [clh_end_file] :        ! End of initialization file.
;  808     0932  2             !
;  809     0933  2             RETURN false;
;  810     0934  2
;  811     0935  2         [clh_too_long] :        ! Record would not fit in input area (ira).
;  812     0936  2             !
;  813     0937  3             BEGIN
;  814     0938  3             erms (rnfrtl, .fs_start (ira), .fs_length (ira));
;  815     0939  3             kcns ();
;  816     0940  3             pus  ();
;  817     0941  2             END;
;  818     0942  2
;  819     0943  2         [otherwise] :           ! Unrecoverable error. Inform user.
;  820     0944  2             !
```

```
;  821     0945  3            BEGIN
;  822     0946  3            erm (rnfure, 0, 0);
;  823     0947  3            erm (rnffab, 0, 0);
;  824     0948  3            .error_condition = true;
;  825     0949  3            RETURN false;
;  826     0950  2            END;
;  827     0951  2
;  828     0952  2        TES;
;  829     0953  2
;  830     0954  2    RETURN true;
;  831     0955  2
;  832     0956  1    END;                                ! End of INPUT_LOOP


                                                       .EXTRN  RINTES

                                            00FC 00000 INPUT_LOOP:
                                                       .WORD   Save R2,R3,R4,R5,R6,R7                  ; 0888
                          57 00000000G   00   9E 00002         MOVAB   ERM, R7                         ;
                          56 00000000G   00   9E 00009         MOVAB   PUS, R6                         ;
                          55 00000000G   00   9E 00010         MOVAB   KHAR, R5                        ;
                          54 00000000G   00   9E 00017         MOVAB   IRA+12, R4                      ;
                                         05   DD 0001E         PUSHL   #5                              ; 0919
                00000000G 00             01   FB 00020         CALLS   #1, CLH                         ;
                          53             50   D0 00027         MOVL    R0, CLH_RESULT                  ;
                          52             01   D0 0002A         MOVL    #1, R2                          ; 0921
                          01             53   D1 0002D         CMPL    CLH_RESULT, #1                  ; 0924
                                         20   12 00030         BNEQ    3$                              ;
                                         52   D4 00032         CLRL    R2                              ;
                                         64   D5 00034         TSTL    IRA+12                          ; 0927
                                         09   14 00036         BGTR    1$                              ;
                          65       00G   8F   9A 00038         MOVZBL  #RINTES, KHAR                   ;
                          64             01   CE 0003C         MNEGL   #1, IRA+12                      ;
                                         0C   11 0003F         BRB     2$                              ;
                          50       F8    A4   D0 00041 1$:     MOVL    IRA+4, R0                       ;
                          65             60   9A 00045         MOVZBL  (R0), KHAR                      ;
                                   F8    A4   D6 00048         INCL    IRA+4                           ;
                                         64   D7 0004B         DECL    IRA+12                          ;
                                         01   DD 0004D 2$:     PUSHL   #1                              ; 0928
                          66             01   FB 0004F         CALLS   #1, PUS                         ;
                          02             53   D1 00052 3$:     CMPL    CLH_RESULT, #2                  ; 0931
                                         04   12 00055         BNEQ    4$                              ;
                                         52   D4 00057         CLRL    R2                              ;
                                         58   11 00059         BRB     9$                              ; 0933
                          04             53   D1 0005B 4$:     CMPL    CLH_RESULT, #4                  ; 0935
                                         30   12 0005E         BNEQ    7$                              ;
                                         52   D4 00060         CLRL    R2                              ;
                                         64   DD 00062         PUSHL   IRA+12                          ; 0938
                                   F4    A4   DD 00064         PUSHL   IRA                             ;
                             00000000G   8F   DD 00067         PUSHL   #RNFRTL                         ;
                00000000G 00             03   FB 0006D         CALLS   #3, ERMS                        ;
                                         64   D5 00074         TSTL    IRA+12                          ; 0939
                                         09   14 00076         BGTR    5$                              ;
                          65       00G   8F   9A 00078         MOVZBL  #RINTES, KHAR                   ;
                          64             01   CE 0007C         MNEGL   #1, IRA+12                      ;
                                         0C   11 0007F         BRB     6$                              ;
```

```
                    50      F8  A4  D0 00081 5$:    MOVL    IRA+4, R0                       ;
                    65          60  9A 00085       MOVZBL  (R0), KHAR                      ;
                        F8      A4  D6 00088       INCL    IRA+4                           ;
                                64  D7 0008B       DECL    IRA+12                          ;
                    66          00  FB 0008D 6$:    CALLS   #0, PUS                         ; 0940
                    1C          52  E9 00090 7$:    BLBC    R2, 8$                          ; 0943
                                7E  7C 00093       CLRQ    -(SP)                           ; 0946
                00000000G       8F  DD 00095       PUSHL   #RNFURE                         ;
                    67          03  FB 0009B       CALLS   #3, ERM                         ;
                                7E  7C 0009E       CLRQ    -(SP)                           ; 0947
                00000000G       8F  DD 000A0       PUSHL   #RNFFAB                         ;
                    67          03  FB 000A6       CALLS   #3, ERM                         ;
                04  BC          01  D0 000A9       MOVL    #1, @ERROR_CONDITION            ; 0948
                                04  11 000AD       BRB     9$                              ; 0949
                    50          01  D0 000AF 8$:    MOVL    #1, R0                          ; 0954
                                04  000B2         RET                                     ;
                                50  D4 000B3 9$:    CLRL    R0                              ; 0956
                                04  000B5         RET                                     ;

; Routine Size:  182 bytes,    Routine Base:  $CODE$ + 01FE

;  837          0957  1
```

```
;  835     0958  1 %IF %BLISS(BLISS32) %THEN
;  836     0959  1 ROUTINE input_file_scan ( rno_cmd
;  837     0960  1                          ,input_file_name : $STR_DESCRIPTOR (CLASS = dynamic)
;  838     0961  1                          ) =
;  839     0962  1 !++
;  840     0963  1 ! FUNCTIONAL DESCRIPTION:
;  841     0964  1 !
;  842     0965  1 !       This routine makes the needed calls to the VMS routine LIB$FIND_FILE.
;  843     0966  1 !       Also, for DSRPLUS, it does the file type resolution.  What this means
;  844     0967  1 !       is that if the user didn't specify an extension and the first FIND_FILE
;  845     0968  1 !       doesn't locate a file with a .RN_ extension, then we will try our own
;  846     0969  1 !       list of extensions starting with .RNO.  Data structure IPFTOP contains
;  847     0970  1 !       the list of known extensions in the order in which we will look for
;  848     0971  1 !       them.  But, for the sake of documentation, here is that list:
;  849     0972  1 !
;  850     0973  1 !               Input file      Maps to this output
;  851     0974  1 !               filetype        file filetype
;  852     0975  1 !
;  853     0976  1 !                .RNO                .MEM
;  854     0977  1 !                .RTB                .MET        (DSRPLUS only)
;  855     0978  1 !                .RNT                .MEC
;  856     0979  1 !                .RNX                .MEX
;  857     0980  1 !                .RND                .DOC
;  858     0981  1 !                .RNH                .HLP
;  859     0982  1 !                .RNB                .BLB
;  860     0983  1 !                .RNC                .CCO
;  861     0984  1 !                .RNE                .ERR
;  862     0985  1 !                .RNL                .PLM
;  863     0986  1 !                .RNM                .MAN
;  864     0987  1 !                .RNP                .OPR
;  865     0988  1 !                .RNS                .STD
;  866     0989  1 !                .***                Wild card  (DSRPLUS only)
;  867     0990  1 !
;  868     0991  1 !       Note that this routine is conditionalized to be compiled only if
;  869     0992  1 !       the VMS implementation is being built.  As for the TOPS-20 implementa-
;  870     0993  1 !       tion, the input and output file extensions are defaulted within the
;  871     0994  1 !       command line interface in RNOT20.  However, this will not take care
;  872     0995  1 !       of the TOPS-10 (or any future PDP-11) implementation.
;  873     0996  1 !
;  874     0997  1 ! FORMAL PARAMETERS:
;  875     0998  1 !
;  876     0999  1 !       RNO_CMD         Data structure from the CLI.  This is where we will
;  877     1000  1 !                       get the filespec string that the user entered.
;  878     1001  1 !
;  879     1002  1 !       INPUT_FILE_NAME Address of a string descriptor into which we will
;  880     1003  1 !                       put the filespec resulting from our efforts.
;  881     1004  1 !
;  882     1005  1 ! IMPLICIT INPUTS:      None
;  883     1006  1 !
;  884     1007  1 ! IMPLICIT OUTPUTS:     None
;  885     1008  1 !
;  886     1009  1 ! ROUTINE VALUE:
;  887     1010  1 !
;  888     1011  1 !       1 = Normal return
;  889     1012  1 !       4 = Fatal return -- file not found
;  890     1013  1 !
;  891     1014  1 ! SIDE EFFECTS:        None
```

```
;  892          1015  1 !--
;  893          1016  1
;  894          1017  2     BEGIN                                   ! Start of input_loop
;  895          1018  2     MAP
;  896          1019  2         rno_cmd : REF $rno_cmd;
;  897          1020  2
;  898          1021  2     BIND
;  899          1022  2         passed_file_spec = rno_cmd [rno$t_input] : $STR_DESCRIPTOR(),
;  900          1023  2         ip_file_spec = .input_file_name : $STR_DESCRIPTOR();
;  901          1024  2
;  902          1025  2     LOCAL
;  903          1026  2         rms_status,
;  904          1027  2         rms_stv_adr : INITIAL(-1),
;  905          1028  2         rms_user_flags : INITIAL(2),
;  906          1029  2         default_file_specification : $STR_DESCRIPTOR (CLASS = dynamic);
;  907          1030  2
;  908          1031  2     default_file_specification [str$b_dtype] = str$k_dtype_t;   ! ASCII text (8-bit)
;  909          1032  2     default_file_specification [str$b_class] = str$k_class_f;   ! Fixed (Scalar) String Descriptor
;  910          1033  2     default_file_specification [STR$H_LENGTH] = 4;
;  911          1034  2     default_file_specification [STR$A_POINTER] = .ipftop[0];
;  912          1035  2
;  913          1036  2     ipftyp = -1;        ! To indicate that we haven't mapped against IPFTOP
;  914          1037  2
;  915          1038  2     rms_status = LIB$FIND_FILE ( passed_file_spec
;  916          1039  2                                 ,.input_file_name
;  917          1040  2                                 ,lffctx
;  918          1041  2                                 ,default_file_specification
;  919          1042  2                                 ,0                ! No related file specification
;  920          1043  2                                 ,rms_stv_adr      ! Secondary RMS error code
;  921          1044  2                                 ,rms_user_flags ! User flags
;  922          1045  2                                 );
;  923          1046  2
;  924        U 1047  2 %IF dsrplus %THEN
;  925        U 1048  2     IF .rms_status EQL RMS$_FNF                                ! If a file wasn't
;  926        U 1049  2                  AND                                       !   found and a
;  927        U 1050  2     CH$FAIL( CH$FIND_CH( .passed_file_spec[str$h_length] !  filetype was not
;  928        U 1051  2                         ,.passed_file_spec[str$a_pointer] ! specified,
;  929        U 1052  2                         ,%C'.' ) )
;  930        U 1053  2     THEN
;  931        U 1054  2         BEGIN                     ! We will try our own default extension(s).
;  932        U 1055  2
;  933        U 1056  2 !!!     lffctx = 0;     ! To start searching all over again.
;  934        U 1057  2
;  935        U 1058  2         ! Search all input filetypes until we find one:
;  936        U 1059  2
;  937        U 1060  2         INCR ipftyp FROM 0 TO (ipftct - 1) DO
;  938        U 1061  2             BEGIN
;  939        U 1062  2
;  940        U 1063  2             ! Point to the currently indexed input filetype option.  The
;  941        U 1064  2             ! length field has already been initialized to 4 outside of
;  942        U 1065  2             ! this INCR loop.
;  943        U 1066  2
;  944        U 1067  2             default_file_specification [STR$A_POINTER] = .ipftop[.ipftyp];
;  945        U 1068  2
;  946        U 1069  2             ! Now see if there is a file with the specified filename and
;  947        U 1070  2             ! the currently indexed input filetype option.
;  948        U 1071  2
```

```
;  949        U 1072  2            rms_status = LIB$FIND_FILE ( passed_file_spec
;  950        U 1073  2                                       ,.input_file_name
;  951        U 1074  2                                       ,lffctx
;  952        U 1075  2                                       ,default_file_specification
;  953        U 1076  2                                       ,0      ! No related file specification
;  954        U 1077  2                                       ,rms_stv_adr  ! Secondary RMS error code
;  955        U 1078  2                                       ,2                ! User flags
;  956        U 1079  2                                       );
;  957        U 1080  2
;  958        U 1081  2            IF .rms_status EQL RMS$_NORMAL THEN EXITLOOP
;  959        U 1082  2            END                        !end of INCR loop.
;  960        U 1083  2        END;                ! End of what we do if first test didn't find a file
;  961          1084  2    %FI
;  962          1085  2
;  963          1086  2        IF  .rms_status EQL RMS$_NORMAL       ! Succeeded in finding (any type)?
;  964          1087  2        THEN
;  965          1088  3            BEGIN
;  966          1089  3
;  967          1090  3            ! Pick off the name and length of the file spec for ERROR.BLI
;  968          1091  3            irac_fspecp = .ip_file_spec [STR$A_POINTER];
;  969          1092  3            irac_fspecc = .ip_file_spec [STR$H_LENGTH];
;  970        U 1093  3    %IF dsrplus %THEN
;  971        U 1094  3            IF  (.ipftyp GTR 0)  ! If we didn't find an RNO file,  tell the
;  972        U 1095  3            THEN                 ! user just what we did find.
;  973        U 1096  3                erm (rnfoft); ! File type ".RNO" not found.  Processing file xx
;  974          1097  3    %FI
;  975          1098  3            true                 ! Exit saying we found an input file.
;  976          1099  3            END
;  977          1100  2        ELSE
;  978          1101  3            BEGIN
;  979          1102  3            LOCAL
;  980          1103  3                spec_length,
;  981          1104  3                spec_ptr;
;  982          1105  3
;  983          1106  3            spec_ptr    = .passed_file_spec [str$a_pointer];
;  984          1107  3            spec_length = .passed_file_spec [str$h_length];
;  985          1108  3
;  986          1109  3            erme (rnfcoi, .spec_ptr, .spec_length, .semcod);
;  987          1110  3            IF .rms_stv_adr EQL -1 THEN
;  988          1111  3                SIGNAL ( .rms_status )
;  989          1112  3            ELSE
;  990          1113  3                SIGNAL ( .rms_status, .rms_stv_adr );
;  991          1114  3
;  992          1115  3            4       ! Exit saying we couldn't find an input file.
;  993          1116  3            END
;  994          1117  1        END;                                ! End of routine INPUT_FILE_SCAN


                                 003C 00000 INPUT_FILE_SCAN:
                                                .WORD   Save R2,R3,R4,R5                        ; 0959
                      55 00000000G   00  9E 00002       MOVAB   LIB$SIGNAL, R5                  ;
                      5E             08  C2 00009       SUBL2   #8, SP                          ;
                      52       04    AC  7D 0000C       MOVQ    RNO_CMD, R2                     ; 1022
                      7E             01  CE 00010       MNEGL   #1, RMS_STV_ADR                 ; 1023
```

```
                                   02  DD 00013          PUSHL   #2                                       :
             08   AE 010E0004      8F  D0 00015          MOVL    #17694724, DEFAULT_FILE_SPECIFICATION    : 1033
             0C   AE 00000000G     00  D0 0001D          MOVL    IPFTOP, DEFAULT_FILE_SPECIFICATION+4     : 1034
      00000000G   00               01  CE 00025          MNEGL   #1, IPFTYP                               : 1036
                                   5E  DD 0002C          PUSHL   SP                                       : 1038
                          08       AE  9F 0002E          PUSHAB  RMS_STV_ADR                              :
                                   7E  D4 00031          CLRL    -(SP)                                    :
                          14       AE  9F 00033          PUSHAB  DEFAULT_FILE_SPECIFICATION               :
                   00000000G       00  9F 00036          PUSHAB  LFFCTX                                   :
                          08       AC  DD 0003C          PUSHL   INPUT_FILE_NAME                          : 1039
                                   52  DD 0003F          PUSHL   R2                                       : 1038
      00000000G   00               07  FB 00041          CALLS   #7, LIB$FIND_FILE                        :
                  54               50  D0 00048          MOVL    R0, RMS_STATUS                           :
      00010001    8F               54  D1 0004B          CMPL    RMS_STATUS, #65537                       : 1086
                                   13  12 00052          BNEQ    1$                                       :
      00000000G   00      04       A3  D0 00054          MOVL    4(R3), IRAC+16                           : 1091
      00000000G   00               63  3C 0005C          MOVZWL  (R3), IRAC+20                            : 1092
                  50               01  D0 00063          MOVL    #1, R0                                   : 1088
                                       04 00066          RET                                              :
                  51      04       A2  D0 00067 1$:      MOVL    4(R2), SPEC_PTR                          : 1106
                  50               62  3C 0006B          MOVZWL  (R2), SPEC_LENGTH                        : 1107
                   00000000G       00  DD 0006E          PUSHL   SEMCOD                                   : 1109
                                   50  DD 00074          PUSHL   SPEC_LENGTH                              :
                                   51  DD 00076          PUSHL   SPEC_PTR                                 :
                   00000000G       8F  DD 00078          PUSHL   #RNFCOI                                  :
      00000000G   00               04  FB 0007E          CALLS   #4, ERME                                 :
      FFFFFFFF    8F      04       AE  D1 00085          CMPL    RMS_STV_ADR, #-1                         : 1110
                                   07  12 0008D          BNEQ    2$                                       :
                                   54  DD 0008F          PUSHL   RMS_STATUS                               : 1111
                  65               01  FB 00091          CALLS   #1, LIB$SIGNAL                           :
                                   08  11 00094          BRB     3$                                       :
                          04       AE  DD 00096 2$:      PUSHL   RMS_STV_ADR                              : 1113
                                   54  DD 00099          PUSHL   RMS_STATUS                               :
                  65               02  FB 0009B          CALLS   #2, LIB$SIGNAL                           :
                  50               04  D0 0009E 3$:      MOVL    #4, R0                                   : 1101
                                       04 000A1          RET                                              : 1117

; Routine Size:  162 bytes,    Routine Base:  $CODE$ + 02B4


;  995          1118  1 %FI
;  996          1119  1 END                                    ! End of module
;  997          1120  0 ELUDOM



                                                          .EXTRN  LIB$SIGNAL

;                         PSECT SUMMARY
;
;       Name                       Bytes                      Attributes
;
;   $OWN$                            492  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   $CODE$                           854  NOVEC,NOWRT,  RD ,  EXE,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          2          0         581        00:01.0
;  _$255$DUA28:[SYSLIB]XPORT.L32;1         590        117         19         252        00:00.6
;  _$255$DUA28:[RUNOFF.SRC]DSRLIB.L32;1   1248        127         10          86        00:01.0




;                                       COMMAND QUALIFIERS

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

; Size:         854 code + 492 data bytes
; Run Time:        00:28.3
; Elapsed Time:    00:58.5
; Lines/CPU Min:    2371
; Lexemes/CPU-Min: 42594
; Memory Used:  251 pages
; Compilation Complete
```

RSDERM
LIS

RTERM
LIS

RNODEF
LIS

RUNOFF
LIS

RNOVMS
LIS

RSKIPS
LIS

RNODAT
LIS

SAVRES
LIS

RNOURS
LIS

RNFERM
LIS