```
                                                       M 7

   CCCCCCCCCCCC     OOOOOOOOO     BBBBBBBBBBBB     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
   CCCCCCCCCCCC     OOOOOOOOO     BBBBBBBBBBBB     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
   CCCCCCCCCCCC     OOOOOOOOO     BBBBBBBBBBBB     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
CCC              OOO         OOO  BBB         BBB  RRR         RRR        TTT        LLL
CCC              OOO         OOO  BBB         BBB  RRR         RRR        TTT        LLL
CCC              OOO         OOO  BBB         BBB  RRR         RRR        TTT        LLL
CCC              OOO         OOO  BBB         BBB  RRR         RRR        TTT        LLL
CCC              OOO         OOO  BBB         BBB  RRR         RRR        TTT        LLL
CCC              OOO         OOO  BBB         BBB  RRR         RRR        TTT        LLL
CCC              OOO         OOO  BBBBBBBBBBBB     RRRRRRRRRRRR           TTT        LLL
CCC              OOO         OOO  BBBBBBBBBBBB     RRRRRRRRRRRR           TTT        LLL
CCC              OOO         OOO  BBBBBBBBBBBB     RRRRRRRRRRRR           TTT        LLL
CCC              OOO         OOO  BBB         BBB  RRR   RRR              TTT        LLL
CCC              OOO         OOO  BBB         BBB  RRR   RRR              TTT        LLL
CCC              OOO         OOO  BBB         BBB  RRR   RRR              TTT        LLL
CCC              OOO         OOO  BBB         BBB  RRR      RRR           TTT        LLL
CCC              OOO         OOO  BBB         BBB  RRR      RRR           TTT        LLL
CCC              OOO         OOO  BBB         BBB  RRR      RRR           TTT        LLL
   CCCCCCCCCCCC     OOOOOOOOO     BBBBBBBBBBBB     RRR         RRR        TTT        LLLLLLLLLLLLLLL
   CCCCCCCCCCCC     OOOOOOOOO     BBBBBBBBBBBB     RRR         RRR        TTT        LLLLLLLLLLLLLLL
   CCCCCCCCCCCC     OOOOOOOOO     BBBBBBBBBBBB     RRR         RRR        TTT        LLLLLLLLLLLLLLL
```

```
**FILE**ID**COBACCEPT                                L  1
```

```
  CCCCCCCC    OOOOOO    BBBBBBBB      AAAAAA      CCCCCCCC    CCCCCCCC  EEEEEEEEEE  PPPPPPPP    TTTTTTTTTT
  CCCCCCCC    OOOOOO    BBBBBBBB      AAAAAA      CCCCCCCC    CCCCCCCC  EEEEEEEEEE  PPPPPPPP    TTTTTTTTTT
CC          OO      OO  BB      BB  AA      AA  CC          CC          EE          PP      PP      TT
CC          OO      OO  BB      BB  AA      AA  CC          CC          EE          PP      PP      TT
CC          OO      OO  BB      BB  AA      AA  CC          CC          EE          PP      PP      TT
CC          OO      OO  BB      BB  AA      AA  CC          CC          EE          PP      PP      TT
CC          OO      OO  BBBBBBBB    AA      AA  CC          CC          EEEEEEEE    PPPPPPPP        TT
CC          OO      OO  BBBBBBBB    AA      AA  CC          CC          EEEEEEEE    PPPPPPPP        TT
CC          OO      OO  BB      BB  AAAAAAAAAA  CC          CC          EE          PP              TT
CC          OO      OO  BB      BB  AAAAAAAAAA  CC          CC          EE          PP              TT
CC          OO      OO  BB      BB  AA      AA  CC          CC          EE          PP              TT          ....
CC          OO      OO  BB      BB  AA      AA  CC          CC          EE          PP              TT          ....
  CCCCCCCC    OOOOOO    BBBBBBBB    AA      AA    CCCCCCCC    CCCCCCCC  EEEEEEEEEE  PP              TT          ....
  CCCCCCCC    OOOOOO    BBBBBBBB    AA      AA    CCCCCCCC    CCCCCCCC  EEEEEEEEEE  PP              TT          ....

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
```

```
0 %TITLE 'COB$ACCEPT - VAX COBOL ACCEPT Statement'
0 MODULE COB$ACCEPT (                                   !
0                       IDENT = '1-018'                 ! File: COBACCEPT.B32 EDIT:LGB1018
0                                       ) =
1 BEGIN
1
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1 !
1
1 !++
1 ! FACILITY:  COBOL SUPPORT
1 !
1 ! ABSTRACT:
1 !
1 !       Supports the COBOL ACCEPT statement.
1 !
1 !       Contains COB$$OPEN_IN  to open an RMS file for input.
1 !
1 !
1 ! ENVIRONMENT:  VAX-11 User Mode
1 !
1 ! AUTHOR: Rich Reichert, CREATION DATE: 16-JULY-79
1 !
1 ! MODIFIED BY:
1 !
1 ! 1-001 - Original.  RKR 16-JULY-79
1 ! 1-002 - Make COB$$OPEN_IN stop instead of signal on open error.
1 !         RKR 4-SEPT-79
1 ! 1-003 - Make COB$$READ_RMS signal COB$_EOFON_ACC if an EOF is
1 !         encountered during reading.
1 !         Do string copy into caller's buffer via CH$COPY instead of
1 !         STR$COPY to avoid dependency on STR$ routines.
1 !         RKR 14-SEPT-79
1 ! 1-004 - Identify file name on bad RMS  status other than EOF.
1 !         RKR 25-SEPT-79
1 ! 1-005 - Change name of symbolic LIBRARY file. RKR 1-OCT-79
```

```
1 ! 1-006 - Make module name match entry point.  RKR 20-OCT-79
1 ! 1-007 - Change references to LIB$_INVARG to COB$_INVARG.
1 ! 1-008 - Make sensitive to names in REQUIRE file.  RKR 21-OCT-79
1 ! 1-009 - Improve errors signaled.  RKR 21-OCT-79
1 !         Cosmetic changes.  RKR 21-OCT-79
1 ! 1-010 - Imperative clean-ups, also try SYS$ logicals.
1 !         PDG 00-FEB-81
1 ! 1-011 - Fix call to $TRNLOG to test for SS$_NORMAL
1 !                 (since SS$_NOTRAN is a success status).
1 !         Add COB$ACCEPT_EOF to allow ACCEPT ... AT END imp-statement.
1 !         Allow MAX(COB$K_ACC_SIZE, .STRING[DSC$W_LENGTH]) bytes for ACCEPT.
1 !         PDG 24-Jul-1981
1 ! 1-012 - Updated copyright date.  LB 9-Aug-81
1 ! 1-013 - Removed COB$ACCEPT_EOF.  This functionality is provided by a flag
1 !         passed to COB$ACCEPT.
1 ! 1-014 - Add code in COB$ACCEPT to check the STV2 field in the RAB to
1 !         determine if the terminator is an escape sequence, and if so,
1 !         to return the escape sequence in the user's buffer.  This was
1 !         done in response to an SPR regarding incompatibilities between
1 !         COBOL-74 and VAX-11 COBOL.  LEB  16-FEB-82
1 ! 1-015 - Version 3 ACCEPT with screen enhancements.     LGB 15-AUG-83
1 ! 1-016 - Code converted from QIO calls to RMS.          LGB 20-JAN-84
1 ! 1-017 - Reset RAB[RAB$V_ETO] bit in COB$ACCEPT.
1 !         Added code to COB$$ILLEGAL_TERM and COB$$DELETE_KEY to handle
1 !         a Control Z situation.
1 !         Added a condition to the IF statement in COB$$ILLEGAL_TERM that
1 !         handles bell ringing for an illegal terminator.
1 !         Put an RMS workaround in routine COB$$PARTIAL_SEQ. LGB 11-JUL-84
1 ! 1-018 - Bug fix to COB$ACCEPT - reverse IF stmt within the CH$COPY stmt.
1 !                                                          LBG 10-SEP-84
1 !--
```

```
1 !
1 ! PROLOGUE FILE
1 !
1 REQUIRE 'RTLIN:COBPROLOG' ;                               ! Switches, Psects, Include
1                                                          ! files
1 !
1 ! TABLE OF CONTENTS:
1 !
1 FORWARD ROUTINE
1         COB$ACCEPT,                                      ! Perform ACCEPT
1         COB$ACC_SCR,                                     ! Perform ACCEPT with screen enhancements
1         COB$$ACC_SCR_FILE,                               ! Dealing with files not terminals,
1                                                          ! RMS call is different
1         COB$$OPEN_IN              :  NOVALUE,            ! Open for input
1         COB$$RMS_GET              :  NOVALUE,            ! Perform an RMS $GET call
1         COB$$RMS_PUT_BYTE         :  NOVALUE,            ! Perform a one byte RMS $PUT
1         COB$$RMS_PUT_BUFFER       :  NOVALUE,            ! Perform a buffer RMS $PUT
1         COB$$CONTROL_Z            :  NOVALUE,            ! Handle Control Z
1         COB$$PARTIAL_SEQ          :  NOVALUE,            ! Put entire escape seq in buffer
1         COB$$DELETE_KEY           :  NOVALUE,            ! Delete Key processing
1         COB$$ILLEGAL_TERM         :  NOVALUE,            ! Look for valid terminator
1         COB$$CLEAN_UP             :  NOVALUE,            ! Clean up before return to COBOL
1         COB$$RPG_CLEAN_UP         :  NOVALUE,            ! Clean up before return to RPG
1         COB$$FORMAT_FOUR ;                               ! Format four CONTROL KEY routine
1 !
1 ! EQUATED SYMBOLS:
1 !
1 LITERAL
1         NUM_UNITS = COB$K_UNIT_MAX - COB$K_UNIT_MIN + 1 ;
1                                                          ! Number of units
1 LITERAL
1         DISP       = 0,                                  ! Code for Display
1         DNA        = 1,                                  ! Code for Display no Advancing
1         POS        = 2,                                  ! Code for Positioning
1         POS_DNA    = 3,                                  ! Code for Positioning no Advancing
1         ACC_ADV    = 4,                                  ! Code for Accept (COBOL V3)
1         ACC_DNA    = 5,                                  ! Code for Accept no Adv (COBOL V3)
1         FLAG_MASK = 15,                                  ! Mask for first four bits of
1                                                          ! parameter FLAGS (0-3)
1         V_BELL     = 16,                                 ! Bit flag for terminal bell
1         V_CONV     = 32,                                 ! Bit flag for conversion
1         V_DEC_PT   = 64,                                 ! Bit flag for 'Decimal Point
1                                                          ! is Comma'
1         V_NO_SIGN = 128,                                 ! Bit flag yes/no include sign
1         V_PROTECT = 256,                                 ! Bit flag for protection
1         V_NO_ECHO = 512,                                 ! Bit flag for no-echo
1         V_ADV      = 1024,                               ! Bit flag for advancing(1)/
1                                                          ! no advancing (0)
1         V_COB_RPG = 2048,                                ! Bit flag for VAX COBOL /
1                                                          ! VAX RPG
1         DEL_KEY    = %X'7F',                             ! Delete key
1         CZ         = %X'1A',                             ! Control Z
1         CARR_RET   = 0,                                  ! Parameters for routine
1         LINE_FD    = 1,                                  ! COB$$RMS_PUT_BYTE
1         RING_BELL = 2,
1         RMS_HEADER = 14 ;                                ! RMS uses up 14 bytes of
1 !                                                        ! the buffer for header info
```

```
1 ! GUARDS:
1 !
1 !     Since the code assumes that COB$K_UNIT_MIN equals 0, and COB_TABLE
1 !         has only 7 items in it, we safeguard this module.
1 !
1 %IF COB$K_UNIT_MIN NEQ 0 %THEN %ERROR('Unexpected COB$K_UNIT_MIN value') %FI
1 %IF COB$K_UNIT_MAX GTR 6 %THEN %ERROR('Unexpected COB$K_UNIT_MAX value') %FI
1
1 GLOBAL
1     ACC_SCR       :  INITIAL (0) ;                    ! Flag for COB$ACCEPT
1 OWN
1     XABTRM        :  $XABTRM_DECL,                    ! RMS XABTRM Control Block
1     XAB_ITMLST    :  $ITMLST_DECL (ITEMS=2),          ! Item list for XABTRM
1
1     !+
1     !
1     !   Terminator mask.
1     !   Leave Control C, Y, S, and Q for VMS.
1     !   Control I is Tab.  Control M is Carriage Return.
1     !   ENTER key has the same value as <CR> i.e. Control M.
1     !   Escape key is a terminator for a VMS V4 system.
1     !   Delete key is processed via routine COB$$DELETE_KEY.
1     !-
1
1     MASK_VECTOR :  BITVECTOR [160]                    ! 20 bytes by 8 bits
1                        PRESET ( [TAB] = 1,
1                                 [CR ] = 1,
1                                 [26 ] = 1,            ! Control z
1                                 [01 ] = 1,            ! Control a
1                                 [02 ] = 1,            ! Control b
1                                 [04 ] = 1,            ! Control d
1                                 [05 ] = 1,            ! Control e
1                                 [06 ] = 1,            ! Control f
1                                 [07 ] = 1,            ! Control g
1                                 [08 ] = 1,            ! Control h
1                                 [10 ] = 1,            ! Control j
1                                 [11 ] = 1,            ! Control k
1                                 [12 ] = 1,            ! Control l
1                                 [14 ] = 1,            ! Control n
1                                 [15 ] = 1,            ! Control o
1                                 [16 ] = 1,            ! Control p
1                                 [18 ] = 1,            ! Control r
1                                 [20 ] = 1,            ! Control t
1                                 [21 ] = 1,            ! Control u
1                                 [22 ] = 1,            ! Control v
1                                 [23 ] = 1,            ! Control w
1                                 [24 ] = 1,            ! Control x
1                                 [27 ] = 1,            ! Escape key for Arrow & PF Keys
1                                                       ! and Alternate Keypad Mode
1                                 [127] = 1,            ! Delete key
1                                 [143] = 1,            ! SS3 for Professional Keys
1                                 [155] = 1             ! CSI for Professional Keys
1                               ) ;
1 !
1 !
1 !     MACROS:
1 !
1 MACRO  COB$$B_STVO_TERM = 12,0,8,0  %;                ! Location of terminator if it
```

```
;      1                                                   ! is not an escape sequence
;      1 MACRO  COB$$B_STV2_LEN  = 14,0,8,0  %;           ! Length of escape sequence
;      1
;      1 MACRO
;    M 1     $VERIFY_TERMINATOR =
;    M 1     !+
;    M 1     !   If parameter KEY not sent (.KEY = 0) then CR, TAB, CONTROL Z,
;    M 1     !   and DELETE KEY are the only legal terminators.
;    M 1     !
;    M 1     !   If parameter KEY not 0 then CR, TAB, CONTROL Z, DELETE KEY, PF,
;    M 1     !   ARROW and SPECIAL FUNCTION PROFESSIONAL Keys are legal terminators.
;    M 1     !   Copy terminator to KEY parameter if valid.
;    M 1     !   Flag LEGAL set to 1 if terminator is valid.
;    M 1     !-
;    M 1             BEGIN
;    M 1
;    M 1             IF .TERM_SIZE EQL 1
;    M 1             THEN
;    M 1                 BEGIN
;    M 1                 TERM_PTR = RAB [COB$$B_STV0_TERM] ;
;    M 1                 SELECTONE .RAB [COB$$B_STV0_TERM] OF
;    M 1                     SET
;    M 1                         [ CR,                           ! Carriage Return
;    M 1                           TAB ] :                       ! Tab
;    M 1
;    M 1                             BEGIN
;    M 1                             IF .KEY NEQ 0
;    M 1                             THEN
;    M 1                                 CH$MOVE ( 1, .TERM_PTR, .KEY [DSC$A_POINTER] ) ;
;    M 1                             LEGAL = 1 ;
;    M 1                             END ;
;    M 1
;    M 1                         [ CZ ] :                        ! Control z
;    M 1
;    M 1                             BEGIN
;    M 1                             !+
;    M 1                             !   CONTROL Z hit along with data
;    M 1                             !-
;    M 1                             IF (.FLAGS AND V_COB_RPG) NEQ 0
;    M 1                             THEN
;    M 1                                 BEGIN                   ! Control Z is illegal
;    M 1                                 LEGAL = 0 ;             ! for VAX RPG
;    M 1                                 TERM_SIZE = 0 ;
;    M 1                                 END
;    M 1                             ELSE                        ! Special meaning for
;    M 1                                 BEGIN                   ! VAX COBOL
;    M 1                                 COB$$CLEAN_UP ( .PARAMETERS, .FLAGS ) ;
;    M 1                                 COB$$CONTROL_Z ( .UNIT, .KEY ) ;
;    M 1                                 RETURN 0 ;
;    M 1                                 END ;
;    M 1                             END ;
;    M 1
;    M 1                         [ DEL_KEY ] :                   ! Delete key
;    M 1
;    M 1                             BEGIN
;    M 1                             COB$$DELETE_KEY ( .PARAMETERS, .UNIT, .FLAGS ) ;
;    M 1                             NO_BELL = 1 ;               ! Special processing for
```

```
                END ;                               ! the DELETE KEY.

            [OTHERWISE] :                           ! Error - key not a
                                                    ! legal terminator
                BEGIN
                LEGAL = 0 ;
                TERM_SIZE = 0 ;
                END ;
        TES ;
    END

ELSE
    IF .CHARS_READ EQL 0  AND  .RAB [RAB$L_STS] EQL RMS$_EOF
    THEN
        !+
        !  CONTROL Z hit alone
        !-
        BEGIN
        IF (.FLAGS AND V_COB_RPG) NEQ 0
        THEN
            BEGIN                       ! Control Z is illegal
            LEGAL = 0 ;                 ! for VAX RPG
            TERM_SIZE = 0 ;
            END
        ELSE                            ! Special meaning for
            BEGIN                       ! VAX COBOL
            COB$$CLEAN_UP ( .PARAMETERS, .FLAGS ) ;
            COB$$CONTROL_Z ( .UNIT, .KEY ) ;
            RETURN 0 ;
            END ;
        END
    ELSE
        !+
        !  Escape Sequence as Terminator.
        !-
        IF .KEY NEQ 0
        THEN
            !+
            !  COB$$CONTROL_KEY converts terminator sequences to COBOL
            !  defined sequences and fills in KEY parameter if terminator
            !  is legal.
            !-
            BEGIN
            IF NOT ( COB$$CONTROL_KEY (TERM_PTR, .TERM_SIZE, .KEY) )
            THEN
                BEGIN
                LEGAL = 0 ;
                TERM_SIZE = 0 ;
                END
            ELSE
                LEGAL = 1 ;
            END
        ELSE
        !+
        !  KEY parameter not passed.  Escape sequences are not
        !  legal terminators.
        !-
```

```
                    BEGIN
                    LEGAL = 0 ;
                    TERM_SIZE = 0 ;
                    END ;
            END ;                               ! End $VERIFY_TERMINATOR macro
    % ;

MACRO
    $ERROR_REPROMPT =
    !+
    !  Ring terminal bell to signal a CONVERSION  error was been made during
    !  data input, restore cursor to original position and perform another
    !  $GET to look for valid data.
    !-

    BEGIN                                       ! Begin $ERROR_REPROMPT macro

    LOCAL
        PUT_TOTAL        :  INITIAL (0),        ! # of chars to $PUT
        INDEX            :  INITIAL (0),        ! Pointer to RESTORE_CURSOR
        RESTORE_CURSOR   :  VECTOR [5000, BYTE]; ! Holds sequence for restoring
                                                ! cursor position.
    COB$$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ;

    !+
    !  If NO ECHO was set no need to do any erasing of input data
    !-

    IF .YES_NO_ECHO EQL 0
    THEN
        BEGIN

        IF (.PUT_FLAG NEQ 0) AND (.YES_PROTECT  EQL  0)
        THEN

        !+
        !  If no protection set and attributes were turned on - turn them off.
        !  If protection was set, leave the FIELD on the screen.  OFF_BUF holds
        !  escape sequence to turn off attributes.  OFF_LEN - length of that
        !  sequence.
        !-

            BEGIN
            CH$MOVE ( .OFF_LEN, OFF_BUF [0], RESTORE_CURSOR [0] ) ;
            PUT_TOTAL = .OFF_LEN ;                      ! Total for $PUT so far
            END ;

        !+
        !  Sequence for reprompting - Backspace, Space, Backspace for each input
        !  character.
        !-

        INDEX = .PUT_TOTAL ;                    ! PUT_TOTAL = 0 or .OFF_LEN
        INCR P FROM .PUT_TOTAL TO (.PUT_TOTAL+(.CHARS_READ-1)) DO
            BEGIN
            RESTORE_CURSOR [.INDEX]   = BS ;            ! Backspace
            RESTORE_CURSOR [.INDEX+1] = BLANK ;         ! Space
        RESTORE_CURSOR [.INDEX+2] = BS ;            ! Backspace
        INDEX = .INDEX + 3 ;
        END ;
    PUT_TOTAL = .PUT_TOTAL + (.CHARS_READ*3) ;      ! Total for $PUT so far

    IF (.PUT_FLAG NEQ 0) AND (.YES_PROTECT  EQL  0)
    THEN

    !+
    !  If no protection set and attributes used - turn them on again.
    !  (after deleting all characters from screen).  ON_BUF holds escape
    !  sequence to turn on attributes.  ON_LEN - length of that sequence.
    !-

        BEGIN
        CH$MOVE ( .ON_LEN, ON_BUF [0], RESTORE_CURSOR [.PUT_TOTAL] ) ;
        PUT_TOTAL = .PUT_TOTAL + .ON_LEN ;          ! Total for $PUT
        END ;

    END ;

!+
!  Max for $PUT buffer is 1024 (can be increased by changing the max on
!  a SYSGEN parameter).  If user input 500 characters the total sequence
!  for reprompting would be 1500 bytes plus possible sequences for turning
!  attributes off and on again, therefore perform a $PUT in sets of 1024
!  until the whole buffer RESTORE_CURSOR has been written to terminal.
!-

BEGIN
    LOCAL
        P_TOT,                                      ! Length of $PUT.
        LAST_WRITE : INITIAL (0) ;                  ! = 1 for final $PUT -
                                                    ! $PUT less than 1024 bytes
WHILE .LAST_WRITE EQL 0 DO
    BEGIN
    IF .PUT_TOTAL GTR (COB$K_ACC_SIZE - RMS_HEADER) !  COB$K_ACC_SIZE = 1024
    THEN
        BEGIN                                       ! Need multiple $PUTs.
        P_TOT = COB$K_ACC_SIZE - RMS_HEADER ;       ! # to Write to screen this time.
        PUT_TOTAL = .PUT_TOTAL - .P_TOT ;           ! # still to Write.
        END
    ELSE
        BEGIN                                       ! Final $PUT
        P_TOT = .PUT_TOTAL ;
        LAST_WRITE = 1 ;
        END ;
    END;

    !+
    !  Clear screen of invalid input
    !-

    COB$$RMS_PUT_BUFFER ( RESTORE_CURSOR [0], .P_TOT, .FLAGS ) ;
END ;

!+
    !   Perform another $GET - looking for valid input
    !-

    RAB = .COB$$AL_WRITE_RAB [ .UNIT[0] ] ;
    COB$$RMS_GET ( .RAB, .FUNC_VAL, .ACC_SIZE, .PUT_HERE [DSC$A_POINTER] ) ;

    REPROMPT_DONE = 1 ;                         ! Signal that REPROMPT has been
                                                ! done.
    END ;                                       ! End of $ERROR_REPROMPT macro
  % ;

MACRO
    $BIND_PARAMETERS =
    !+
    !   Put data used by many of the subroutines in a vector of data.
    !   BIND all the separate names that can be used to identify the
    !   various elements of the vector.
    !-
    BIND
        PUT_HERE        = PARAMETERS [0]  :  BLOCK [,BYTE],  ! Buffer to hold input
        NEXT_CHAR       = PARAMETERS [3]  :  VECTOR [,BYTE], ! Buffer used for
                                                             ! PROTECTION check
        ACC_SIZE        = PARAMETERS [6]  :  WORD,          ! Length for RMS $GET
        CHARS_READ      = PARAMETERS [7],               ! Number of input characters
        FUNC_VAL        = PARAMETERS [8],               ! QIO Function Modifiers used
                                                        ! in the item list for RMS $GET
        TERM_SIZE       = PARAMETERS [9],               ! Size of terminator
        TERM_LOC        = PARAMETERS [10],              ! Location of terminator
        TERM_PTR        = PARAMETERS [11],              ! Pointer to terminator in buffer
        TERM_IN_NEXT    = PARAMETERS [12],              ! = 1 if terminator in NEXT_CHAR
        TERM_FROM_DEL   = PARAMETERS [13],              ! Flag from COB$$DELETE_KEY to
                                                        ! COB$$ILLEGAL_TERM
        LEGAL           = PARAMETERS [14],              ! = 0 if illegal terminator hit
        YES_PROTECT     = PARAMETERS [15],              ! = 1 if PROTECTED requested
        YES_DEFAULT     = PARAMETERS [16],              ! = 1 if DEFAULT used as input
        PUT_FLAG        = PARAMETERS [17],              ! Flag for turning on attributes
        OFF_BUF         = PARAMETERS [18]  :  VECTOR [,BYTE], ! Holds esc seq to
                                                              ! turn off attributes
        OFF_LEN         = PARAMETERS [21] ;             ! Length of esc seq in OFF_BUF
    % ;
!
!   The following tables convert the UNIT number into a logical name.
!
MACRO
    DESC_(A) = UPLIT BYTE(%ASCIC A) - BASE %;
BIND
    BASE = UPLIT(REP 0 OF (0)),
    COB_TABLE = UPLIT(
        DESC_('COB$INPUT'),
        DESC_('COB$OUTPUT'),
        DESC_('COB$CONSOLE'),
        DESC_('COB$CARDREADER'),
        DESC_('COB$PAPERTAPEREADER'),
        DESC_('COB$LINEPRINTER'),
        DESC_('COB$PAPERTAPEPUNCH')):   VECTOR[NUM_UNITS],
    SYS_TABLE = UPLIT(
        DESC_('SYS$INPUT'),
;  489          2004  1             DESC_('SYS$OUTPUT'),
;  490          2005  1             DESC_('SYS$ERROR'),
;  491          2006  1             DESC_('SYS$INPUT'),
;  492          2007  1             DESC_('SYS$INPUT'),
;  493          2008  1             DESC_('SYS$OUTPUT'),
;  494          2009  1             DESC_('SYS$OUTPUT')):                      VECTOR[NUM_UNITS];
;  495          2010  1
;  496          2011  1   !
;  497          2012  1   ! EXTERNAL REFERENCES:
;  498          2013  1   !
;  499          2014  1   EXTERNAL ROUTINE
;  500          2015  1
;  501          2016  1             COB$$CONTROL_KEY,                                  ! Translate terminator
;  502          2017  1             COB$$ACC_CONVERT,                                  ! Conversion routine
;  503          2018  1             COB$$OPEN_OUT    :  NOVALUE,                       ! Open for output
;  504          2019  1             LIB$STOP : NOVALUE,                                ! Signals fatal error
;  505          2020  1             LIB$GET_VM,                                        ! Get virtual memory
;  506          2021  1             LIB$FREE_VM,                                       ! Free virtual memory
;  507          2022  1             STR$GET1_DX,                                       ! Allocate a string
;  508          2023  1             STR$DUPL_CHAR,                                     ! Duplicate character n times
;  509          2024  1             STR$FREE1_DX,                                      ! Deallocate a string
;  510          2025  1             STR$COPY_R,                                        ! Copy a string by ref
;  511          2026  1             COB$$SETUP_TERM_TYPE,                              ! Setup terminal type
;  512          2027  1             COB$$SET_ATTRIBUTES_ONLY ;                         ! Set bold, reverse, blink,
;  513          2028  1                                                                ! underline
;  514          2029  1   EXTERNAL LITERAL
;  515          2030  1             COB$_ERRDURACC,                                    ! Error during DISPLAY
;  516          2031  1             COB$_FAIGET_VM,                                    ! Failure to get VM
;  517          2032  1             COB$_EOFON_ACC,                                    ! EOF on ACCEPT
;  518          2033  1             COB$_INVDEFVAL,                                    ! DEFAULT value too large
;  519          2034  1             COB$_INVARG ;                                      ! Invalid Argument(s)
;  520          2035  1
;  521          2036  1   EXTERNAL
;  522          2037  1             COB$$AL_WRITE_RAB        :  VECTOR,                ! Address of RAB
;  523          2038  1             COB$$AW_WRITE_IFI        :  VECTOR [,WORD],        ! Internal file identifiers
;  524          2039  1             COB$$AB_USPCODE          :  VECTOR [,BYTE],        ! Prefix and Post upspacing
;  525          2040  1             COB$$AB_PREV             :  VECTOR [,BYTE],        ! History of previous call
;  526          2041  1             COB$ACC_TERM_TYPE,                                 ! Terminal type for ACCEPT
;  527          2042  1             COB$TERM_TYPE;                                     ! Terminal type for DISPLAY
```

```
1 %SBTTL 'COB$ACCEPT - Version 1 ACCEPT Statement'
1 GLOBAL ROUTINE COB$ACCEPT (UNIT, STRING) =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Reads a record from specified unit and delivers record to
1 !       caller's string.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       UNIT.rbu.va     Byte integer unit number designating the unit from
1 !                       which the string is to be read, followed by byte
1 !                       flag indicating whether routine should (false) abort,
1 !                       or (true) return status on RMS$_EOF.
1 !
1 !       STRING.wt.ds    The address of a fixed-string descriptor to
1 !                       receive the string read.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       Status of whether the file in question is currently open.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       Updated status of the file just used.
1 !
1 ! ROUTINE VALUE:
1 !
1 !       If .UNIT[1] is false:
1 !           Unspecified.
1 !
1 !       If .UNIT[1] is true:
1 !           Either true or false, indicating success or EOF, respectively.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Reads a record from the designated unit.
1 !
1 !--
1
2     BEGIN
2     MAP
2       UNIT:   VECTOR[,BYTE],
2       STRING: REF BLOCK[8, BYTE];
2
2     LOCAL
2       RAB:    REF $RAB_DECL,
2       STATUS,
2       DESCR:  BLOCK [8, BYTE],
2       TERM_SIZE,
2       BUFFER: VECTOR [COB$K_ACC_SIZE, BYTE];
2
2     IF .UNIT[0] GTRU COB$K_UNIT_MAX
2     THEN
2       LIB$STOP(COB$_INVARG);
2
```

```
2      ! If this file is not open, open it.
2      ! 0 as second parameter to COB$$OPEN_IN signifies that VAX COBOL is used.
2      !
2      IF .COB$$AL_WRITE_RAB[.UNIT[0]] EQL 0
2      THEN
2          COB$$OPEN_IN(.UNIT[0], 0);
2
2      !  Perform a Linefeed when a Version 1 ACCEPT statement follows a
2      !  Version 3 ACCEPT statement with advancing.
2      !
2      IF .COB$$AB_PREV [0] EQL ACC_ADV
2      THEN COB$$RMS_PUT_BYTE ( LINE_FD, 0 ) ;
2
2      ! Read a record into our buffer or caller's buffer, whichever is larger.
2      !
2      RAB = .COB$$AL_WRITE_RAB[.UNIT[0]];
2      IF .STRING[DSC$W_LENGTH] GTRU COB$K_ACC_SIZE
2      THEN
3          BEGIN
3          RAB[RAB$W_USZ] = .STRING[DSC$W_LENGTH];
3          RAB[RAB$L_UBF] = .STRING[DSC$A_POINTER];
3          END
2      ELSE
3          BEGIN
3          RAB[RAB$W_USZ] = COB$K_ACC_SIZE;
3          RAB[RAB$L_UBF] = BUFFER;
2          END;
2
2      !+
2      !  Turn off RAB [RAB$V_ETO] just in case a 'screen enhancement ACCEPT'
2      !  was performed before this one.  COB$$RMS_GET set RAB [RAB$V_ETO]
2      !  to signal to RMS to expect an 'extented terminal' $GET.  Here we
2      !  only want a simple $GET with no bells and whistles.  If RAB [RAB$V_ETO]
2      !  is not turned off unwanted behavior will result.
2      !-
2
2      RAB [RAB$V_ETO] = 0 ;
2
2      ! Read the record.
2      !
2      WHILE $GET(RAB = .RAB) EQL RMS$_RSA DO $WAIT(RAB = .RAB);
2
2
2      IF NOT .RAB[RAB$L_STS]
2      THEN
2          LIB$STOP(
3              (IF .RAB[RAB$L_STS] EQL RMS$_EOF
3                  THEN
3                      IF .UNIT[1]
3                      THEN RETURN 0
3                      ELSE COB$_EOFON_ACC
2                  ELSE COB$_ERRDURACC),
2              1, .RAB+RAB$C_BLN, .RAB[RAB$L_STS], .RAB[RAB$L_STV]);
2
2  !+
2  ! Check if the terminator size is greater than 1.  If it is,
2  ! this indicates that the terminator string is an escape sequence.
```

```
2 ! Return the entire escape sequence in the user's buffer.
2 !-
2
2      TERM_SIZE = .RAB[RAB$W_STV2];                               ! Get terminator size
2
2      CH$COPY( (IF .TERM_SIZE GTR 1
3                 THEN .RAB[RAB$W_RSZ] + .TERM_SIZE
3                 ELSE .RAB[RAB$W_RSZ] ),
2          .RAB[RAB$L_UBF], %C' ', .STRING[DSC$W_LENGTH], .STRING[DSC$A_POINTER]);
2
2      !+
2      !  VAX COBOL Version 1 / Version 3 interaction.
2      !  Interaction with COB$ACC_SCR - Perform a Carriage Return if necessary
2      !  and signal that this is an ACCEPT with advancing.
2      !-
2
2      IF .ACC_SCR
2      THEN
2          COB$$RMS_PUT_BYTE ( CARR_RET, 0 ) ;
2      COB$$AB_PREV[0] = ACC_ADV ;
2
2      RETURN 1;
1      END;                                                         ! End COB$ACCEPT

                                                        .TITLE  COB$ACCEPT COB$ACCEPT - VAX COBOL ACCEPT Statem
;                                                                                  ent
                                                        .IDENT  \1-018\

                                                        .PSECT  _COB$DATA,NOEXE,  PIC,2

                               00000000  00000 ACC_SCR::
                                                        .LONG   0                                               ;
                                         00004 XABTRM:  .BLKB   36
                                         00028 XAB_ITMLST:
                                                        .BLKB   28
                         0D   F5   FF   F6  00044 MASK_VECTOR:
                                                        .BYTE   -10, -1, -11, 13                                ;
                                       00#  00048       .BYTE   0[11]                                           ;
                                        80  00053       .BYTE   -128                                            ;
                                        00  00054       .BYTE   0                                               ;
                                        80  00055       .BYTE   -128                                            ;
                                        00  00056       .BYTE   0                                               ;
                                        08  00057       .BYTE   8                                               ;

                                                        .PSECT  _COB$CODE,NOWRT,  SHR,  PIC,2

                                            00000 P.AAA:  .BLKB   0
               54  55  50  4E  49  24  42  4F  43  09  00000 P.AAC:  .ASCII  <9>\CCB$INPUT\                          ;
           54  55  50  54  55  4F  24  42  4F  43  0A  0000A P.AAD:  .ASCII  <10>\COB$OUTPUT\                        ;
       45  4C  4F  53  4E  4F  43  24  42  4F  43  0B  00015 P.AAE:  .ASCII  <11>\COB$CONSOLE\                       ;
52  45  44  41  45  52  44  52  41  43  24  42  4F  43  0E  00021 P.AAF:  .ASCII  <14>\COB$CARDREADER\                    ;
52  45  50  41  54  52  45  50  41  50  24  42  4F  43  13  00030 P.AAG:  .ASCII  <19>\COB$PAPERTAPEREADER\               ;
                                        52  45  44  41  45  0003F                                                     ;
45  54  4E  49  52  50  45  4E  49  4C  24  42  4F  43  0F  00044 P.AAH:  .ASCII  <15>\COB$LINEPRINTER\                   ;
                                                        52  00053                                                     ;
50  45  50  41  54  52  45  50  41  50  24  42  4F  43  12  00054 P.AAI:  .ASCII  <18>\COB$PAPERTAPEPUNCH\                ;
```

```
                                             48  43  4E  55  00063                                                                    ;
                                                             00067            .BLKB   1
00000044  00000030  00000021  00000015  0000000A  00000000  00068 P.AAB:   .LONG   0, 10, 21, 33, 48, 68, 84                          ;
                                                  00000054  00080                                                                     ;
                    54  55  50  4E  49  24  53  59  53  09  00084 P.AAK:   .ASCII  <9>\SYS$INPUT\                                     ;
                54  55  50  54  55  4F  24  53  59  53  0A  0008E P.AAL:   .ASCII  <10>\SYS$OUTPUT\                                   ;
                    52  4F  52  52  45  24  53  59  53  09  00099 P.AAM:   .ASCII  <9>\SYS$ERROR\                                     ;
                    54  55  50  4E  49  24  53  59  53  09  000A3 P.AAN:   .ASCII  <9>\SYS$INPUT\                                     ;
                    54  55  50  4E  49  24  53  59  53  09  000AD P.AAO:   .ASCII  <9>\SYS$INPUT\                                     ;
                54  55  50  54  55  4F  24  53  59  53  0A  000B7 P.AAP:   .ASCII  <10>\SYS$OUTPUT\                                   ;
                54  55  50  54  55  4F  24  53  59  53  0A  000C2 P.AAQ:   .ASCII  <10>\SYS$OUTPUT\                                   ;
                                                             000CD            .BLKB   3
000000B7  000000AD  000000A3  00000099  0000008E  00000084  000D0 P.AAJ:   .LONG   132, 142, 153, 163, 173, 183, 194                  ;
                                                  000000C2  000E8                                                                     ;

                                                                  BASE=                   P.AAA
                                                                  COB_TABLE=              P.AAB
                                                                  SYS_TABLE=              P.AAJ
                                                                           .EXTRN  COB$$CONTROL_KEY
                                                                           .EXTRN  COB$$ACC_CONVERT
                                                                           .EXTRN  COB$$OPEN_OUT, LIB$STOP
                                                                           .EXTRN  LIB$GET_VM, LIB$FREE_VM
                                                                           .EXTRN  STR$GET1_DX, STR$DUPL_CHAR
                                                                           .EXTRN  STR$FREE1_DX, STR$COPY_R
                                                                           .EXTRN  COB$$SETUP_TERM_TYPE
                                                                           .EXTRN  COB$$SET_ATTRIBUTES_ONLY
                                                                           .EXTRN  COB$_ERRDURACC, COB$_FAIGET_VM
                                                                           .EXTRN  COB$_EOFON_ACC, COB$_INVDEFVAL
                                                                           .EXTRN  COB$_INVARG, COB$$AL_WRITE_RAB
                                                                           .EXTRN  COB$$AW_WRITE_IFI
                                                                           .EXTRN  COB$$AB_USPCODE
                                                                           .EXTRN  COB$$AB_PREV, COB$ACC_TERM_TYPE
                                                                           .EXTRN  COB$TERM_TYPE, SYS$GET
                                                                           .EXTRN  SYS$WAIT

                                                      00FC  00000          .ENTRY  COB$ACCEPT, Save R2,R3,R4,R5,R6,R7             ; 2044
                                  57 00000000G   00   9E  00002          MOVAB   COB$$AB_PREV, R7                               ;
                                  56 00000000G   00   9E  00009          MOVAB   LIB$STOP, R6                                   ;
                                  5E      FBF8   CE   9E  00010          MOVAB   -1032(SP), SP                                  ;
                                  52        04   AC   9A  00015          MOVZBL  UNIT, R2                                       ; 2096
                                  06             52   91  00019          CMPB    R2, #6                                         ;
                                                 09   1B  0001C          BLEQU   1$                                             ;
                                     00000000G   8F   DD  0001E          PUSHL   #COB$_INVARG                                   ; 2098
                                  66             01   FB  00024          CALLS   #1, LIB$STOP                                   ;
                                  53 00000000G0042   DE   00027 1$:      MOVAL   COB$$AL_WRITE_RAB[R2], R3                      ; 2103
                                                 63   D5  0002F          TSTL    (R3)                                           ;
                                                 09   12  00031          BNEQ    2$                                             ;
                                                 7E   D4  00033          CLRL    -(SP)                                          ; 2105
                                                 52   DD  00035          PUSHL   R2                                             ;
                           0000V  CF             02   FB  00037          CALLS   #2, COB$$OPEN_IN                               ;
                                  04             67   91  0003C 2$:      CMPB    COB$$AB_PREV, #4                               ; 2110
                                                 08   12  0003F          BNEQ    3$                                             ;
                                  7E             01   7D  00041          MOVQ    #1, -(SP)                                      ; 2111
                           0000V  CF             02   FB  00044          CALLS   #2, COB$$RMS_PUT_BYTE                          ;
                                  52             63   D0  00049 3$:      MOVL    (R3), RAB                                      ; 2115
                                  53        08   AC   D0  0004C          MOVL    STRING, R3                                     ; 2116
                           0400   8F             63   B1  00050          CMPW    (R3), #1024                                    ;
```

```
                                             0B  1B 00055        BLEQU    4$                                       ;
                           20 A2             63  B0 00057        MOVW     (R3), 32(RAB)                            ; 2119
                           24 A2          04 A3  D0 0005B        MOVL     4(R3), 36(RAB)                           ; 2120
                                             0A  11 00060        BRB      5$                                       ; 2116
                           20 A2        0400 8F  B0 00062 4$:    MOVW     #1024, 32(RAB)                           ; 2124
                           24 A2             6E  9E 00068        MOVAB    BUFFER, 36(RAB)                          ; 2125
                           07 A2             10  8A 0006C 5$:    BICB2    #16, 7(RAB)                              ; 2136
                                             52  DD 00070 6$:    PUSHL    RAB                                      ; 2140
                    00000000G 00             01  FB 00072        CALLS    #1, SYS$GET                              ;
                     000182DA 8F             50  D1 00079        CMPL     R0, #99034                               ;
                                             0B  12 00080        BNEQ     7$                                       ;
                                             52  DD 00082        PUSHL    RAB                                      ;
                    00000000G 00             01  FB 00084        CALLS    #1, SYS$WAIT                             ;
                                             E3  11 0008B        BRB      6$                                       ;
                              2B          08 A2  E8 0008D 7$:    BLBS     8(RAB), 10$                              ; 2143
                              7E          08 A2  7D 00091        MOVQ     8(RAB), -(SP)                            ; 2152
                                          44 A2  9F 00095        PUSHAB   68(RAB)                                  ;
                                             01  DD 00098        PUSHL    #1                                       ; 2145
                     0001827A 8F          08 A2  D1 0009A        CMPL     8(RAB), #98938                           ; 2146
                                             0F  12 000A2        BNEQ     8$                                       ;
                              47          05 AC  E8 000A4        BLBS     UNIT+1, 14$                              ; 2148
                              50   00000000G 8F  D0 000A8        MOVL     #COB$_EOFON_ACC, R0                      ;
                                             50  DD 000AF        PUSHL    R0                                       ;
                                             06  11 000B1        BRB      9$                                       ;
                                   00000000G 8F  DD 000B3 8$:    PUSHL    #COB$_ERRDURACC                          ; 2146
                              66             05  FB 000B9 9$:    CALLS    #5, LIB$STOP                             ;
                              50          0E A2  3C 000BC 10$:   MOVZWL   14(RAB), TERM_SIZE                       ; 2160
                              01             50  D1 000C0        CMPL     TERM_SIZE, #1                            ; 2162
                                             09  15 000C3        BLEQ     11$                                      ;
                              51          22 A2  3C 000C5        MOVZWL   34(RAB), R1                              ; 2163
                              50             51  C0 000C9        ADDL2    R1, R0                                   ;
                                             04  11 000CC        BRB      12$                                      ;
                              50          22 A2  3C 000CE 11$:   MOVZWL   34(RAB), R0                              ; 2164
       63        20        24 B2             50  2C 000D2 12$:   MOVC5    R0, @36(RAB), #32, (R3), @4(R3)          ; 2165
                                          04 B3     000D8                                                          ;
                              07   00000000' EF  E9 000DA        BLBC     ACC_SCR, 13$                             ; 2173
                                             7E  7C 000E1        CLRQ     -(SP)                                    ; 2175
                        0000V CF             02  FB 000E3        CALLS    #2, COB$$RMS_PUT_BYTE                    ;
                              67             04  90 000E8 13$:   MOVB     #4, COB$$AB_PREV                         ; 2176
                              50             01  D0 000EB        MOVL     #1, R0                                   ; 2178
                                                 04 000EE        RET                                               ;
                                             50  D4 000EF 14$:   CLRL     R0                                       ; 2179
                                                 04 000F1        RET                                               ;

; Routine Size:  242 bytes,    Routine Base:  _COB$CODE + 00EC
```

```
%SBTTL 'COB$ACC_SCR - ACCEPT with screen enhancements'
GLOBAL ROUTINE COB$ACC_SCR ( UNIT           :  VECTOR [2,BYTE],
                             STRING_DEST    :  REF $STR$DESCRIPTOR,
                             FLAGS,
                             DEFAULT        :  REF $STR$DESCRIPTOR,
                             SIZE,
                             KEY            :  REF $STR$DESCRIPTOR,
                             LENGTH
                           ) =

!++
! FUNCTIONAL DESCRIPTION:
!
!       Performs COBOL ACCEPT statement with screen enhancements.
!       Reads a record from a specified UNIT and deposits record in
!       STRING_DEST.
!       A call to COB$POS_ACCEPT is made by the VAX COBOL Compiler
!       prior to the call to COB$ACC_SCR to set cursor position and
!       perform any screen or line erasing.
!
! CALLING SEQUENCE:
!
!       RETURN_STATUS.wlc.v = COB$ACC_SCR ( UNIT.rbu.va, STRING_DEST.mt.ds,
!                                           [FLAGS.rlu.v], [DEFAULT.rt.dx],
!                                           [SIZE.rlu.v], [KEY.wt.ds],
!                                           [LENGTH.wlu.r] )
! FORMAL PARAMETERS:
!
!       UNIT.rbu.va     Array of two unsigned byte integers.
!                       The first byte is the unit number designating the
!                       device from which the string is to be read.
!                       The second byte indicates whether the routine should
!                       abort or return to the calling program.
!                         Byte 2 = 0  -  routine will abort on control z
!                                        and reprompt on conversion errors.
!                                = 1  -  ( AT END )
!                                        routine will return to calling program
!                                        on control z and reprompt on conversion
!                                        errors.
!                                = 2  -  ( ON EXCEPTION )
!                                        routine will return to calling program
!                                        on control z and conversion errors.
!
!       STRING_DEST.mt.ds   Address of descriptor to receive the read input.
!
!       FLAGS.rlu.v     Screen enhancement flag;
!
!                               bit 0  -  bold
!                               bit 1  -  reverse
!                               bit 2  -  blink
!                               bit 3  -  underline
!                               bit 4  -  bell
!                               bit 5  -  conversion
!                               bit 6  -  decimal point is comma
!                               bit 7  -  0 to allow space for sign in PROTECTED
!                                         ACCEPT, 1 no allowance for sign
!                               bit 8  -  protect
```

```
1 ! !                                 bit 9  -  no-echo
1 ! !                                 bit 10 -  0 advancing, 1 no advancing
1 ! !                                 bit 11 -  0 for VAX COBOL, 1 for VAX RPG
1 ! !
1 ! !      DEFAULT.rt.dx    Default source moved to destination descriptor
1 ! !                       (STRING_DEST) in the event of null input.
1 ! !
1 ! !      SIZE.rlu.v       Size of protected field.  Only applicable if the
1 ! !                       protected flag is set.
1 ! !
1 ! !      KEY.wt.ds        Destination of the receiving field of the control key
1 ! !
1 ! !      LENGTH.wlu.r     Destination of the number of characters read
1 ! !
1 ! IMPLICIT INPUTS:
1 ! !
1 ! !      Status of whether the input file is currently open.
1 ! !
1 ! IMPLICIT OUTPUTS:
1 ! !
1 ! !      Updated status of file
1 ! !
1 ! ROUTINE VALUE:
1 ! !
1 ! !      If .UNIT[1] is false :  Unspecified.
1 ! !      If .UNIT[1] is true  :  Either true or false, indicating success or
1 ! !                              EOF, respectively.
1 ! !
1 ! SIDE EFFECTS:
1 ! !
1 ! !      Reads a record from a designated uint.
1 ! !
1 !--
1
2    BEGIN
2
2    LOCAL
2         !+
2         !  Note; other declarations are in the macro $BIND_PARAMETERS.
2         !-
2         RAB              :  REF $RAB_DECL,
2         PUT_SIZE         :  WORD,                ! ACC_SIZE plus 5 (for escape
2                                                  ! sequences)
2         ON_BUF           :  VECTOR [20,BYTE],    ! Holds escape seq to turn on
2                                                  ! terminal attributes
2         ON_LEN           :  INITIAL (0),         ! Length of ON_BUF
2         FUNC_VAL_2,                              ! QIO Function Modifiers used
2                                                  ! in the item list for RMS $GET
2         PROT_OK          :  INITIAL (0),         ! = 1 if no Protection errors
2         CONV_OK          :  INITIAL (0),         ! = 1 if no Conversion errors
2         REPROMPT_DONE    :  INITIAL (0),         ! = 1 if reprompt performed in
2                                                  ! response to a Conversion error
2         YES_CONV         :  INITIAL (0),         ! = 1 if Conversion requested
2         YES_NO_ECHO      :  INITIAL (0),         ! = 1 if No-Echo requested
2         YES_SIGN         :  INITIAL (1),         ! = 1 if no allowance for sign
2                                                  ! given. NOTE - initialized to 1
2         P_DATA_TYPE      :  INITIAL (0),         ! PP99 or 99PP data types
```

```
2            ZEROES,                                   ! %X'0' filler
2            BLANKS,                                   ! %C' ' filler
2            PARAMETERS       :  VECTOR [22]           ! Buffer to hold data to be
2                                INITIAL (REP 22 OF (0)) ; ! passed to subroutines
2        BUILTIN
2            NULLPARAMETER ;
2
2         LITERAL
2            F_PROT_SIZE = 13,                         ! # of chars allowed for input
2            D_PROT_SIZE = 22 ;                        ! when PROTECTED is requested
2   !+                                                 ! for floating and double fl.
2   !    Bind PARAMETERS to other names.
2   !-
2         $BIND_PARAMETERS ;
2
2   !+
2   !    Fillers - used by STR$DUPL_CHAR, therefore they cannot be literals
2   !-
2         ZEROES = %X'0' ;
2         BLANKS = %C' ' ;
2
2   !+
2   !    Put ACCEPTed data from RMS $GET in this buffer.
2   !-
2
2         PUT_HERE [DSC$W_LENGTH]  = 0 ;
2         PUT_HERE [DSC$B_DTYPE]   = DSC$K_DTYPE_NL ;
2         PUT_HERE [DSC$B_CLASS]   = DSC$K_CLASS_D ;
2         PUT_HERE [DSC$A_POINTER] = 0 ;
2
2   !+
2   !    Determine if PROTECTION has been requested.
2   !    If so, set the size of the field by either the value of the SIZE parameter
2   !    or the length field of the STRING_DEST descriptor.
2   !    If no PROTECTION requested, use COB$K_ACC_SIZE (1024 - same as
2   !    V1 Accept).
2   !    Also make adjustments if both PROTECTION and CONVERSION are requested -
2   !    add room for sign and a decimal point, in some cases look at DSC$B_DIGITS
2   !    instead of DSC$W_LENGTH.
2   !    'P' data types need special handling.
2   !    Use STR$GET1_DX to allocate space for dynamic string PUT_HERE.
2   !-
2
2         IF ( .FLAGS AND V_CONV ) NEQ 0 THEN YES_CONV = 1 ;      !  Avoid BLISS
2         IF ( .FLAGS AND V_NO_SIGN ) NEQ 0 THEN YES_SIGN = 0 ;   !  optimization problems
2
2         IF ( .FLAGS AND V_PROTECT ) NEQ 0
2         THEN
3             BEGIN                                            ! Begin Protect Size
3             YES_PROTECT = 1 ;
3             IF .SIZE NEQ 0
3             THEN ACC_SIZE = .SIZE                            ! Use SIZE
3             ELSE
4                 BEGIN                                        ! Begin no SIZE param
4                     LOCAL
4                         pp99  :  initial (0) ;               ! Scale for PP99 data
4                                                              ! type
```

```
4
4              pp99 = .string_dest [dsc$b_digits] + .string_dest [dsc$b_scale] ;
4              ACC_SIZE = .STRING_DEST [DSC$W_LENGTH] ;     ! Use STRING_DEST
4
4   !+
4   !  Special case "P" data types (each "P" specifies an assumed scaling position).
4   !  NOTE:  All code pertaining to the "P" data type is in lowercase.  Since "P"
4   !  data types are such an off the wall issue, leaving this code in lowercase is
4   !  the best way to avoid "P" code interfering with "normal" data types.
4   !-
6              if ((.string_dest [dsc$b_class] eql dsc$k_class_sd )
7                  and  ((.pp99 lss 0)                      ! P Picture of PP99
5                  or (.string_dest[dsc$b_scale] gtr 0)))   ! P Picture of 99PP.
4              then
5                  begin                                    ! begin P data types
5                  p_data_type = 1 ;
5                  if .pp99 lss 0                           ! P Picture of PP99
5                  then
5                      acc_size = abs (.string_dest[dsc$b_scale])
5                  else
5                      acc_size = .pp99 ;
5
5                  if .yes_conv
5                  then
6                      begin
6                      !+
6                      !  Allow space for a decimal point for PP99 but not 99pp.
6                      !-
6                      if .pp99 lss 0
6                      then
6                          acc_size = .acc_size + 1 ;      ! decimal point for pp99
6
6                      !+
6                      !  Because we are reading the digits and scale fields,
6                      !  all numeric data types will need an extra space for
6                      !  the sign - except Numeric Unsigned.
6                      !-
6
6                      if .string_dest [dsc$b_dtype] neq dsc$k_dtype_nu
6                      then
6                          acc_size = .acc_size + 1 ;
6                      !+
6                      !  Additional check for VAX_11 COBOL COMP and COMP3
6                      !  data types - if YES_SIGN = 0 then do not include
6                      !  space for sign.
6                      !-
8                      if (((.string_dest [dsc$b_dtype] eql dsc$k_dtype_w  ) or
8                           (.string_dest [dsc$b_dtype] eql dsc$k_dtype_wu ) or
8                           (.string_dest [dsc$b_dtype] eql dsc$k_dtype_l  ) or
8                           (.string_dest [dsc$b_dtype] eql dsc$k_dtype_lu ) or
8                           (.string_dest [dsc$b_dtype] eql dsc$k_dtype_q  ) or
8                           (.string_dest [dsc$b_dtype] eql dsc$k_dtype_qu ) or
8                           (.string_dest [dsc$b_dtype] eql dsc$k_dtype_p ))
7                                                             and .yes_sign eql 0 )
6                      then
6                          acc_size = .acc_size - 1 ;
5                      end ;
```

```
5                 end                                        ! end P data types
5
4               else
4                 !+
4                 ! Non P data type
4                 !-
5                 begin                                      ! Begin non P data type
5                 IF .YES_CONV                               ! Adjust ACC_SIZE
5                 THEN
5                     !+
5                     ! Make room for overpunch sign.
5                     ! Packed data type - check to see if sign should be
5                     ! included.
5                     !-
6                     BEGIN
7                     IF ((.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_NRO ) OR
7                         (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_NLO ) OR
8                         (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_P  AND
7                                                         .YES_SIGN ))
6                     THEN
6                         ACC_SIZE = .ACC_SIZE + 1 ;
6
6                     !+
6                     ! COMP - look at digits field plus one for sign, only if
6                     ! conversion is requested.
6                     ! VAX_COBOL always sends an SD descriptor for W, L, Q when
6                     ! Conversion is used.
6                     ! Check to see if sign should be included.
6                     !-
6
7                     IF (.STRING_DEST [DSC$B_CLASS] EQL DSC$K_CLASS_SD )
6                     THEN
8                         IF (((.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_W  ) OR
8                              (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_WU ) OR
8                              (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_L  ) OR
8                              (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_LU ) OR
8                              (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_Q  ) OR
8                              (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_QU ))
7                                                          AND .YES_SIGN )
6                         THEN
6                             ACC_SIZE = .STRING_DEST [DSC$B_DIGITS] + 1 ;
6
6                     !+
6                     ! Floating pt - 13 for Floating, 22 for Double Floating.
6                     !-
6
7                     IF (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_F )
6                     THEN
6                         ACC_SIZE = F_PROT_SIZE ;
7                     IF (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_D )
6                     THEN
6                         ACC_SIZE = D_PROT_SIZE ;
6
6                     !+
6                     ! Make room for decimal point
6                     !-
6
```

```
7                       IF (.STRING_DEST [DSC$B_CLASS] EQL DSC$K_CLASS_SD )
6                       THEN
6                           ACC_SIZE = .ACC_SIZE + 1 ;
5                       END ;
4                   end ;                                   ! End non P data type
3               END ;                                       ! End no SIZE param
3           END                                             ! End Protect Size
2       ELSE
2           ACC_SIZE = COB$K_ACC_SIZE - RMS_HEADER ;        ! 1024 - 14 is same
2                                                           ! limit as a DISPLAY
2   !+
2   !  Allocate enough room in PUT_HERE to hold the terminator escape sequences.
2   !  Most sequences are 4 bytes or less.  (PUT_SIZE is 5 more than ACC_SIZE.)
2   !  Note: PUT_SIZE used, not ACC_SIZE.
2   !-
2
2       PUT_SIZE = .ACC_SIZE ;
2       IF .ACC_SIZE LSS 920 THEN PUT_SIZE = .ACC_SIZE + 5 ;
2
3       IF NOT ( STR$GET1_DX ( %REF (.PUT_SIZE ), PUT_HERE ))
2       THEN LIB$STOP ( COB$_ERRDURACC ) ;
2
2   !+
2   !  Check first byte of UNIT param.
2   !  If this file is not open, open it.  (Note: only first byte of UNIT is
2   !                                             sent to COB$$OPEN_IN)
2   !-
2
2       IF .UNIT[0] GTRU COB$K_UNIT_MAX
2       THEN
2           LIB$STOP ( COB$_INVARG ) ;
2
2       IF .COB$$AL_WRITE_RAB [ .UNIT[0] ] EQL 0
2       THEN
2           !+
2           !  Second parameter tells COB$$OPEN_IN whether VAX COBOL (0)
2           !  or VAX RPG (1) is the caller.
2           !-
2           COB$$OPEN_IN ( .UNIT[0],
2                           IF ( .FLAGS AND V_COB_RPG ) NEQ 0
2                           THEN 1
2                           ELSE 0 ) ;
2
2       RAB = .COB$$AL_WRITE_RAB [ .UNIT[0] ] ;
2
2   !+
2   !  Find out if the device is a terminal.
2   !-
2
3       BEGIN
3           LOCAL
3               STATUS,
3               NAM_DSC  :  REF BLOCK [8,BYTE] ;
3
3       NAM_DSC = .RAB + RAB$C_BLN ;
3
3       IF .COB$ACC_TERM_TYPE EQL 0
```

```
3        THEN
4            IF NOT ( COB$$SETUP_TERM_TYPE ( .NAM_DSC [DSC$A_POINTER],
4                                            .NAM_DSC [DSC$W_LENGTH],
4                                            COB$ACC_TERM_TYPE ) )
3            THEN LIB$STOP ( COB$_ERRDURACC ) ;
3
3        IF .COB$ACC_TERM_TYPE EQL UNKNOWN
3        THEN
3            !+
3            !  If terminal is UNKNOWN then it can be assumed we are working
3            !  with files rather than terminals.  Pull out of this routine
3            !  and go to COB$$ACC_SCR_FILE which uses a slightly different
3            !  variation of the RMS $GET Service.
3            !-
4            BEGIN
4            STATUS = COB$$ACC_SCR_FILE ( .UNIT, .STRING_DEST, .FLAGS, .DEFAULT,
4                                           .LENGTH, .ACC_SIZE, PUT_HERE, .YES_CONV,
4                                           .YES_PROTECT, .YES_SIGN ) ;
4            !+
4            !  Free local string PUT_HERE
4            !-
5            IF NOT ( STR$FREE1_DX ( PUT_HERE ))
4            THEN LIB$STOP ( COB$_ERRDURACC ) ;
4
5            IF (NOT .STATUS)
4            THEN
4                RETURN 0
4            ELSE
4                RETURN 1 ;
3            END ;
3
2        END ;
2
2        !+
2        !  Flag to COB$ACCEPT that COB$ACC_SCR has been called.  COB$ACCEPT will
2        !  have to perform a Carriage Return.
2        !-
2
2        ACC_SCR = 1 ;
2
3        BEGIN                                                   !  Begin $GET
3
3    !+
3    !  VAX COBOL Version 1 / Version 3 Interaction.
3    !  Advancing philosophy : <LF>    $GET    <CR>
3    !  <LF> based on previous call.
3    !  <CR> based on current ACCEPT using FLAGS bit 10.
3    !  If previous call requires advancing then perform a linefeed.  DISPLAY (DISP)
3    !  and ACCEPT (ACC_ADV) with advancing.  POS = call to module COB$POS_ERASE
3    !  remembers what previous call was, if advancing then POS, if no advancing
3    !  then POS_DNA
3    !-
3
4        IF (.COB$$AB_PREV[0] EQL DISP
4            OR  .COB$$AB_PREV[0] EQL POS
4            OR  .COB$$AB_PREV[0] EQL ACC_ADV )
3        THEN
```

```
3               !+
3               !  Echo linefeed to terminal
3               !-
3               COB$$RMS_PUT_BYTE ( LINE_FD, .FLAGS ) ;
3
3       !+
3       !  Did user request any terminal attributes (bold, blink, underline, reverse) ?
3       !  If so, call COB$$SET_ATTRIBUTES_ONLY to get escape sequence to turn
3       !  attributes on and off.
3       !  PUT_FLAG - first four bits (0-3) of FLAGS parameter.
3       !-
3
3           PUT_FLAG = .FLAGS  AND  FLAG_MASK ;
3
3           IF .PUT_FLAG NEQ 0
3           THEN
4               IF NOT ( COB$$SET_ATTRIBUTES_ONLY ( .COB$ACC_TERM_TYPE, .PUT_FLAG,
4                                                   ON_BUF [0], ON_LEN,
4                                                   OFF_BUF [0], OFF_LEN ) )
3               THEN LIB$STOP ( COB$_ERRDURACC ) ;
3
3       !+
3       !  If requested, add sequence to ON_BUF to ring terminal bell.
3       !-
3
3           IF ( .FLAGS AND V_BELL ) NEQ 0
3           THEN
4               BEGIN
4               ON_BUF [ .ON_LEN ] = BELL ;
4               ON_LEN = .ON_LEN + 1 ;
3               END ;
3
3       !+
3       !  Check parameters to see if the CONTROL KEY FORMAT 4 ACCEPT has been
3       !  requested.  If so, pull out of this routine and call COB$$FORMAT_FOUR
3       !  which uses a different Terminator Mask and does not need all the
3       !  enhancements in COB$ACC_SCR.
3       !-
3
3               IF NOT NULLPARAMETER (KEY)
3               THEN
4                   BEGIN
4                       LOCAL
4                           KEY_LEN ;
4
4                   KEY_LEN = .KEY [DSC$W_LENGTH] ;
4                   STR$DUPL_CHAR ( .KEY, KEY_LEN, BLANKS ) ;
4
4                   !+
4                   !  If these parameters are not present then we are dealing with
4                   !  a Format Four ACCEPT rather than a Format Three ACCEPT.
4                   !-
4
5                   IF (NULLPARAMETER (LENGTH)  AND
5                       NULLPARAMETER (SIZE)    AND
5                       NULLPARAMETER (DEFAULT) AND
5                       NULLPARAMETER (STRING_DEST) )
```

```
4              THEN
5                  IF NOT ( COB$$FORMAT_FOUR ( .UNIT, .FLAGS, .KEY ))
4                  THEN RETURN 0
4                  ELSE RETURN 1 ;
3              END ;
3
3  !+
3  !  Determine FUNC_VAL - QIO Function Modifiers used by RMS $GET Service.
3  !  Check FLAGS parameter to see if NO-ECHO was requested (bit 9), if so
3  !  set TRM$M_TM_NOECHO to suppress echoing of input characters to the terminal.
3  !  Set TRM$M_TM_ESCAPE to allow Escape Sequences to act as terminators (Arrow
3  !  keys, PF keys, and the Professional editing and top row function keys).
3  !  Set TRM$M_TM_NOFILTR to have the DELETE KEY handled by COB$$DELETE_KEY.
3  !  Set TRM$M_TM_TRMNOECHO to suppress echoing of the termination character
3  !  (COB$$AB_PREV handles advancing / no advancing).
3  !-
3
3      IF ( .FLAGS AND V_NO_ECHO ) NEQ 0
3      THEN
4          BEGIN
4          FUNC_VAL = TRM$M_TM_ESCAPE + TRM$M_TM_NOFILTR + TRM$M_TM_TRMNOECHO
4                                                        + TRM$M_TM_NOECHO ;
4          YES_NO_ECHO = 1 ;
4          END
3      ELSE
3          FUNC_VAL = TRM$M_TM_ESCAPE + TRM$M_TM_NOFILTR + TRM$M_TM_TRMNOECHO ;
3
3  !+
3  !  Main Loop of routine.
3  !  PROT_OK = 1 -> there was no Protection error  ''plus''
3  !  CONV_OK = 1 -> there was no Conversion error  ''equal''  SUCCESS -> pull out
3  !  of loop.  Otherwise continue accepting data until there are no errors.
3  !  If error, reprompt user for more input via macro $ERROR_REPROMPT.
3  !-
3
3      WHILE (.PROT_OK EQL 0)  OR  (.CONV_OK EQL 0) DO
4          BEGIN                                             ! Begin loop
4              LOCAL
4                  TERM_SEEN  :  INITIAL (0) ;               ! Flag for PROTECT check
4
4          IF .REPROMPT_DONE EQL 0
4          THEN
5              BEGIN                                         ! Begin no reprompt
5
5              !+
5              !  If PROTECTION requested, put a Protected Field on the screen.
5              !  $PUT ACC_SIZE blanks to screen with attributes requested
5              !  by user turned on.  (Escape sequences geared to VT100
5              !  terminals)  Can only set a one line field as a max, therefore
5              !  FIELD_BUF holds up to 300 characters.
5              !-
5
5              IF .COB$ACC_TERM_TYPE EQL VT100
5              THEN
6                  BEGIN                                     ! Begin VT100
6                  IF .YES_PROTECT
6                  THEN
```

```
; 1180   2693 7                 BEGIN                                    ! Begin Field
; 1181   2694 7                 LOCAL
; 1182   2695 7                     FIELD_BUF       :  VECTOR [300, BYTE],
; 1183   2696 7                     FIELD_LEN ;                          ! size of FIELD_BUF
; 1184   2697 7
; 1185   2698 7                 !+
; 1186   2699 7                 !  Buffer FIELD_BUF to write Protected Field contains
; 1187   2700 7                 !            - escape sequence to turn attributes on,
; 1188   2701 7                 !            - number of blanks to write to screen and
; 1189   2702 7                 !            - backspaces (same # as blanks) to put cursor
; 1190   2703 7                 !              back to original position.
; 1191   2704 7                 !-
; 1192   2705 7
; 1193   2706 7                 CH$MOVE ( .ON_LEN, ON_BUF [0], FIELD_BUF [0] ) ;
; 1194   2707 7                 FIELD_LEN = .ON_LEN ;
; 1195   2708 7                 CH$FILL ( BLANK, .ACC_SIZE, FIELD_BUF [.FIELD_LEN] ) ;
; 1196   2709 7                 FIELD_LEN = .FIELD_LEN + .ACC_SIZE ;
; 1197   2710 7                 CH$FILL ( BS, .ACC_SIZE, FIELD_BUF [.FIELD_LEN] ) ;
; 1198   2711 7                 FIELD_LEN = .FIELD_LEN + .ACC_SIZE ;
; 1199   2712 7
; 1200   2713 7                 !+
; 1201   2714 7                 !  If size of FIELD_BUF is greater than the size of the
; 1202   2715 7                 !  maximum allowed for a $PUT buffer, issue an error
; 1203   2716 7                 !  message.  Issuing multiple $PUTs at this point does
; 1204   2717 7                 !  not help as the cursor is unable to get back to the
; 1205   2718 7                 !  starting position and ends up in the wrong line.
; 1206   2719 7                 !-
; 1207   2720 7
; 1208   2721 8                 IF .FIELD_LEN GTR (COB$K_ACC_SIZE - RMS_HEADER) ! 1024 -14
; 1209   2722 7                 THEN
; 1210   2723 7                     LIB$STOP ( COB$_ERRDURACC ) ;
; 1211   2724 7
; 1212   2725 7                 !+
; 1213   2726 7                 !  $PUT to write Protected Field to terminal
; 1214   2727 7                 !-
; 1215   2728 7
; 1216   2729 7                 COB$$RMS_PUT_BUFFER ( FIELD_BUF [0], .FIELD_LEN, .FLAGS ) ;
; 1217   2730 7
; 1218   2731 6                 END ;                                    ! End Field
; 1219   2732 5             END ;                                        ! End VT100
; 1220   2733 5
; 1221   2734 5  !*****
; 1222   2735 5  !*****  RMS $GET Service
; 1223   2736 5  !*****
; 1224   2737 5
; 1225   2738 5          !+
; 1226   2739 5          !  RMS $PUT to turn on terminal attributes (blink,bold,underline,reverse).
; 1227   2740 5          !  RMS $GET to accept input.  Do not perform the $PUT if PROTECTED
; 1228   2741 5          !  is requested as the FIELD_BUF $PUT has already turned attributes
; 1229   2742 5          !  on.
; 1230   2743 5          !  Note :  TRM$_PROMPT not used because of buffer size limitations.
; 1231   2744 5          !    TRM$_MODIFIERS uses all of specified buffer for accepting input,
; 1232   2745 5          !    TRM$_PROMPT uses same buffer for both the prompt string and the
; 1233   2746 5          !    accepted data, therefore some space for accepting data is lost.
; 1234   2747 5          !-
; 1235   2748 5
; 1236   2749 5          IF .ON_LEN NEQ 0 AND .YES_PROTECT NEQ 1      ! If requested, turn
```

```
5              THEN                                           ! attributes on
5                  COB$$RMS_PUT_BUFFER ( ON_BUF [0], .ON_LEN, .FLAGS ) ;
5
5              !+
5              !  RMS $GET to accept input from terminal.
5              !-
5
5              RAB = .COB$$AL_WRITE_RAB [ .UNIT[0] ] ;
5              COB$$RMS_GET ( .RAB, .FUNC_VAL, .ACC_SIZE,
5                                               .PUT_HERE [DSC$A_POINTER] ) ;
5
5              END                                            !  End of no reprompt
4          ELSE
4              REPROMPT_DONE = 0 ;                            ! re-set flag
4
4          !+
4          !  Get number of characters read and terminator size from the fields
4          !  of the RAB.  Pass this info along to other routines.
4          !  RAB fields -
4          !       rab [rab$l_sts]          = status
4          !       rab [rab$l_rsz]          = x        no. of chars read
4          !       rab [cob$$b_stv0_term] = d        <cr> terminator seen
4          !       rab [cob$$b_stv2_len]  = 1        size of terminator
4          !  Save this information before COB$$PARTIAL_SEQ does any more $GETs.
4          !-
4
4          CHARS_READ = .RAB [RAB$W_RSZ] ;                    ! Number of chars read
4          TERM_SIZE  = .RAB [COB$$B_STV2_LEN] ;              ! Size and location of
4          TERM_LOC   = .RAB [COB$$B_STV0_TERM] ;             ! terminator - other
4                                                             ! routines may update
4                                                             ! these
4          !+
4          !  Check for partial sequence error - not enough room in input buffer
4          !  to hold entire escape sequence when a Protected ACCEPT is performed.
4          !  If necessary, call COB$$PARTIAL_SEQ to read remainder of sequence.
4          !-
4
4          IF .RAB [RAB$L_STS] EQL RMS$_PES
4          THEN
5              BEGIN
5              TERM_SEEN = 1 ;                                ! Set flag here as
5              COB$$PARTIAL_SEQ ( PARAMETERS, .UNIT ) ;       ! COB$$PARTIAL_SEQ may
4              END ;                                          ! change status value
4
4          !+
4          !  If terminator was the DELETE KEY call COB$$DELETE_KEY.
4          !-
4
4          IF .RAB [COB$$B_STV0_TERM] EQL DEL_KEY
4          THEN
4              COB$$DELETE_KEY ( PARAMETERS, .UNIT, .FLAGS ) ;
4
4 !*******
4 !******* PROTECTED
4 !*******
4
4          !+
4                   !   Was terminator seen on PROTECTED READ?
4                   !
4                   !   Looking for terminator to make sure that user hasn't tried to go
4                   !   beyond the bounds of a PROTECTED READ.
4                   !
4                   !   Two ways for a protected read to complete -
4                   !         1. terminator typed before buffer filled ( no further check necessary)
4                   !         2. buffer fill ( no terminator seen)
4                   !                 - do a one character read to make sure terminator is
4                   !                         typed, not another character.
4                   !
4                   !   The following RAB fields look like this if buffer filled
4                   !                 rab [rab$l_sts]         = status
4                   !                 rab [rab$l_rsz]         = x         no. of chars read (acc_size)
4                   !                 rab [cob$$b_stv0_term] = 0          no terminator seen
4                   !                 rab [cob$$b_stv2_len]  = 0          size of terminator
4                   !-
5               IF (.YES_PROTECT )                                    ! Was PROTECTION requested?
5                       AND  ( .CHARS_READ NEQ 0 )                    ! AND is it needed
4               THEN
4               IF .RAB [RAB$L_STS] EQL RMS$_TNS                      ! RMS$_TNS = Terminator
4                                   AND .TERM_SEEN EQL 0              ! Not Seen
4                   THEN
5                       BEGIN                                         ! Begin protect check $GET
5                       !+
5                       !   After initial $GET is performed it is necessary to perform a
5                       !   $GET of length 1 to make sure that there are no characters
5                       !   typed by the user that exceed the maximum allowed.
5                       !   (Do not echo character to terminal.)
5                       !   If the $GET of one character results in a terminator, there
5                       !   is no problem.
5                       !   If the $GET of one character results in an attempt to type
5                       !   extra characters, there is an error.
5                       !
5                       !   If VAX RPG is the caller, always return control to the
5                       !   calling program on an error.
5                       !-
5
5                       LOCAL
5                           NO_CHAR    :  INITIAL (0),                ! =1 no Protection error
5                           HAVE_TERM  :  INITIAL (0) ;               ! =1 terminator seen
5
5                       WHILE .HAVE_TERM NEQ 1 DO
6                           BEGIN                                     ! Begin HAVE_TERM loop
6
6                           NO_CHAR = 0 ;
6                           FUNC_VAL_2 = TRM$M_TM_ESCAPE + TRM$M_TM_NOFILTR
6                                               + TRM$M_TM_TRMNOECHO + TRM$M_TM_NOECHO ;
6
6                           RAB = .COB$$AL_WRITE_RAB [ .UNIT[0] ] ;
6                           COB$$RMS_GET ( .RAB, .FUNC_VAL_2, 1, NEXT_CHAR ) ;
6
6                               !+
6                               !   If user did not attempt to enter more data, set TERM_SIZE
6                               !   and TERM_IN_NEXT before possible call to COB$$PARTIAL_SEQ.
6                               !   If not enough room in $GET buffer to hold entire escape
```

```
6                     !  sequence then call COB$$PARTIAL_SEQ to read remainder
6                     !  of sequence.
6                     !-
6
6                     IF .RAB [RAB$W_RSZ] EQL 0              ! No more data entered.
6                     THEN
7                         BEGIN
7                         NO_CHAR = 1 ;                      ! Move terminator into
7                         NEXT_CHAR [0] = .RAB [COB$$B_STVO_TERM] ; ! NEXT_CHAR
6                         END ;
6                     TERM_SIZE = .RAB [COB$$B_STV2_LEN] ; ! Terminator size.
6                     TERM_LOC  = .RAB [COB$$B_STVO_TERM]; ! Terminator location.
6                     TERM_IN_NEXT = 1 ;                   ! Terminator on NEXT_CHAR
6                     IF .RAB [RAB$L_STS] EQL RMS$_PES
6                     THEN
6                         COB$$PARTIAL_SEQ ( PARAMETERS, .UNIT ) ;
6
6                     !+
6                     !  Terminators are the only acceptable input at this point.
6                     !  If NO_CHAR = 1 then there is no Protection error.
6                     !-
6
6                     IF .NO_CHAR
6                     THEN
7                         BEGIN                             ! Begin TERM accepted
7                         PROT_OK       = 1 ;               ! $GET successful
7                         HAVE_TERM     = 1 ;
7
7 !******
7 !****** DELETE KEY
7 !******
7                         !+
7                         !  Was termintor the DELETE KEY ?  If so, call
7                         !  COB$$DELETE_KEY to erase the last character
7                         !  read and to continue reading for input.
7                         !-
7
7                         IF .RAB [COB$$B_STVO_TERM] EQL DEL_KEY
7                         THEN
8                             BEGIN
8                             COB$$DELETE_KEY ( PARAMETERS, .UNIT, .FLAGS ) ;
8                             !+
8                             !  Check to see if we fell out of COB$$DELETE_KEY
8                             !  without a valid terminator.  If so, keep
8                             !  looking for it.
8                             !-
8                             IF .TERM_SIZE EQL 0
8                             THEN
9                                 BEGIN
9                                 HAVE_TERM = 0 ;           ! Loop again
9                                 PROT_OK = 0 ;
9                                 END
8                             ELSE
9                                 BEGIN
9                                 HAVE_TERM = 1 ;
9                                 TERM_IN_NEXT = 0 ;        ! Note - COB$$DELETE_KEY put
8                                 END ;                     ! the terminator in
```

```
8                                                            ! PUT_HERE.
7                             END ;
7
7                       END                                  ! End TERM accepted
6                   ELSE
6
6  !*******
6  !******* PROTECTION ERROR
6  !*******
6
6                   !+
6                   !  PROTECTION error :
6                   !  User tried to input too many characters,
6                   !     - sound terminal bell,
6                   !     - leave cursor where it is (No reprompt or backspace).
6                   !-
7
7                       BEGIN
7                       COB$$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ;
7                       PROT_OK = 0 ;                        ! Signal Protection error
7                       HAVE_TERM = 0 ;
7                       END
7
5                   END ;                                    ! End HAVE_TERM loop
5               END                                          ! End protect check $GET
4           ELSE
4               !+
4               ! Protection requested but terminator already seen,
4               ! no need for 1 character Read.
4               !-
4               PROT_OK = 1
4       ELSE
4           !+
4           ! Protection not requested, no need for 1 character Read.
4           !-
4           PROT_OK = 1 ;
4
4  !*******
4  !******* CONTROL KEY
4  !*******
4
4   IF .PROT_OK
4   THEN
4       !+
4       ! No sense going thru Control Key code if there was a protection error.
4       !-
5       BEGIN                                                ! Begin Control Key
5
5           IF .TERM_IN_NEXT                                 ! Locate terminator,
5           THEN                                             ! which buffer is it in.
5               TERM_PTR = NEXT_CHAR[0]
5           ELSE
5               TERM_PTR = .PUT_HERE[DSC$A_POINTER] + .CHARS_READ ;
5
5           !+
5           !  If parameter KEY not sent (.KEY = 0) then CR, TAB, CONTROL Z,
5           !  and DELETE KEY are the only legal terminators.
```

```
; 1465      2978  5      !
; 1466      2979  5      !   If parameter KEY not 0 then CR, TAB, CONTROL Z, DELETE KEY, PF,
; 1467      2980  5      !   ARROW and SPECIAL FUNCTION PROFESSIONAL Keys are legal
; 1468      2981  5      !   terminators.  Copy terminator to KEY parameter.
; 1469      2982  5      !
; 1470      2983  5      !   Special treatment needed for CONTROL Z under RMS.  There is a
; 1471      2984  5      !   difference between ^Z being typed alone and with data.
; 1472      2985  5      !   When ^Z is typed with data the ^Z is stored in
; 1473      2986  5      !   RAB[RAB$_STVO_TERM], but when ^Z is typed alone the status
; 1474      2987  5      !   RMS$_EOF is returned from the $Get Service.
; 1475      2988  5      !-
; 1476      2989  5
; 1477      2990  5      IF .TERM_SIZE EQL 1                          ! One byte terminator
; 1478      2991  5      THEN
; 1479      2992  6          BEGIN
; 1480      2993  6          TERM_PTR = RAB [COB$$B_STVO_TERM] ;
; 1481      2994  6          SELECTONE .RAB [COB$$B_STVO_TERM] OF
; 1482      2995  6              SET
; 1483      2996  6                  [ CR,                        ! Carriage Return
; 1484      2997  6                    TAB ] :                    ! Tab
; 1485      2998  6                      !+
; 1486      2999  6                      !  These keys are legal, do nothing if KEY = 0.
; 1487      3000  6                      !-
; 1488      3001  6                      IF NOT NULLPARAMETER (KEY)
; 1489      3002  6                      THEN
; 1490      3003  6                          CH$MOVE ( 1, .TERM_PTR, .KEY [DSC$A_POINTER] ) ;
; 1491      3004  6
; 1492      3005  6                  [ CZ ] :                     ! Control z
; 1493      3006  6
; 1494      3007  6                      !+
; 1495      3008  6                      !  CONTROL Z hit along with data  ( terminator in
; 1496      3009  6                      !  RAB [COB$$B_STVO_TERM] )
; 1497      3010  6                      !-
; 1498      3011  6
; 1499      3012  7                      BEGIN
; 1500      3013  7                      IF (.FLAGS AND V_COB_RPG) NEQ 0
; 1501      3014  7                      THEN
; 1502      3015  8                          BEGIN
; 1503      3016  8                          !+
; 1504      3017  8                          ! VAX RPG - Control Z is an illegal terminator.
; 1505      3018  8                          !-
; 1506      3019  8                          LEGAL = 0 ;
; 1507      3020  8                          COB$$ILLEGAL_TERM ( PARAMETERS, .UNIT, .FLAGS,
; 1508      3021  8                                                                   .KEY ) ;
; 1509      3022  8                          END
; 1510      3023  7                      ELSE
; 1511      3024  8                          BEGIN
; 1512      3025  8                          !+
; 1513      3026  8                          ! VAX COBOL - Control Z has special meaning.
; 1514      3027  8                          !-
; 1515      3028  8                          COB$$CLEAN_UP ( PARAMETERS, .FLAGS ) ;
; 1516      3029  8                          COB$$CONTROL_Z ( .UNIT, .KEY ) ;
; 1517      3030  8                          RETURN 0 ;
; 1518      3031  7                          END ;
; 1519      3032  6                      END ;
; 1520      3033  6
; 1521      3034  6                  [ DEL_KEY ] :                ! Delete key
; 1522          3035   6
; 1523          3036   7                             BEGIN
; 1524          3037   7                             COB$$DELETE_KEY ( PARAMETERS, .UNIT, .FLAGS ) ;
; 1525          3038   6                             END ;
; 1526          3039   6
; 1527          3040   6                         [OTHERWISE] :                     ! Error - key not a
; 1528          3041   6                                                           ! terminator
; 1529          3042   7                                 BEGIN
; 1530          3043   7                                 LEGAL = 0 ;
; 1531          3044   7                                 COB$$ILLEGAL_TERM ( PARAMETERS, .UNIT, .FLAGS,
; 1532          3045   7                                                                           .KEY ) ;
; 1533          3046   6                                 END ;
; 1534          3047   6                     TES ;
; 1535          3048   6                 END
; 1536          3049   6
; 1537          3050   5             ELSE
; 1538          3051   5                 IF .CHARS_READ EQL 0  AND  .RAB [RAB$L_STS] EQL RMS$_EOF
; 1539          3052   5                 THEN
; 1540          3053   6                     BEGIN
; 1541          3054   6                     !+
; 1542          3055   6                     !  CONTROL Z hit alone - terminator not placed in
; 1543          3056   6                     !  RAB [COB$$B_STVO_TERM], but signaled via RAB [RAB$L_STS].
; 1544          3057   6                     !-
; 1545          3058   6                     IF (.FLAGS AND V_COB_RPG) NEQ 0
; 1546          3059   6                     THEN
; 1547          3060   7                         BEGIN
; 1548          3061   7                         LEGAL = 0 ;                       ! VAX RPG - Control Z
; 1549          3062   7                         COB$$ILLEGAL_TERM ( PARAMETERS, .UNIT, .FLAGS, .KEY ) ;
; 1550          3063   7                         END
; 1551          3064   6                     ELSE
; 1552          3065   7                         BEGIN                             ! VAX COBOL
; 1553          3066   7                         COB$$CLEAN_UP ( PARAMETERS, .FLAGS ) ;
; 1554          3067   7                         COB$$CONTROL_Z ( .UNIT, .KEY ) ;
; 1555          3068   7                         RETURN 0 ;
; 1556          3069   6                         END ;
; 1557          3070   6                     END
; 1558          3071   5                 ELSE
; 1559          3072   6                     BEGIN
; 1560          3073   6
; 1561          3074   6                     !+
; 1562          3075   6                     !  Escape Sequence as Terminator.  .TERM_SIZE greater
; 1563          3076   6                     !  than 1 and RMS_$EOF not signaled.
; 1564          3077   6                     !-
; 1565          3078   6
; 1566          3079   6                     IF NOT NULLPARAMETER (KEY)
; 1567          3080   6                     THEN
; 1568          3081   6                         !+
; 1569          3082   6                         !  COB$$CONTROL_KEY converts terminator sequences to
; 1570          3083   6                         !  COBOL defined sequences and fills in KEY parameter
; 1571          3084   6                         !  if terminator is legal.
; 1572          3085   6                         !-
; 1573          3086   7                         BEGIN
; 1574          3087   8                         IF NOT ( COB$$CONTROL_KEY (TERM_PTR, .TERM_SIZE, .KEY) )
; 1575          3088   7                         THEN
; 1576          3089   8                             BEGIN
; 1577          3090   8                             LEGAL = 0 ;                   ! Illegal escape sequence
; 1578          3091   8                             COB$$ILLEGAL_TERM ( PARAMETERS, .UNIT, .FLAGS,
8                                                                 .KEY ) ;
7                               END ;
7                           END
6                       ELSE
6                           !+
6                           !   Terminator of size greater than 1 is illegal when KEY is
6                           !   not used.
6                           !-
7                           BEGIN
7                           LEGAL = 0 ;
7                           COB$$ILLEGAL_TERM ( PARAMETERS, .UNIT, .FLAGS, .KEY ) ;
6                           END ;
5                       END ;
4               END ;                                           ! End Control Key
4
4   !*******
4   !******* NULL INPUT
4   !*******
4
4               !+
4               !   Null input
4               !   RAB fields look like this for null input
4               !               rab [rab$l_sts]         = 1     status
4               !               rab [rab$l_rsz]         = 0     no. of chars read
4               !               rab [cob$$b_stv0_term] = d      <cr> terminator seen
4               !               rab [cob$$b_stv2_len]   = 1     size of terminator
4               !
4               !   Check for DEFAULT parameter - if present prepare to put it through
4               !   Conversion routines by placing DEFAULT in PUT_HERE.
4               !-
4
5               IF ( .CHARS_READ EQL 0 ) AND (( .FLAGS AND V_COB_RPG ) NEQ 0 )
4               THEN
4                   !+
4                   !   In case of null input for RPG, simply return (no DEFAULT).
4                   !   But perform any necessary clean up first.
4                   !-
5                   BEGIN
5                   COB$$RPG_CLEAN_UP ( .FLAGS ) ;
5                   RETURN 1 ;
4                   END ;
4
5               IF ( .CHARS_READ EQL 0 )
4               THEN
5                   IF NOT NULLPARAMETER (DEFAULT)  AND  (.YES_DEFAULT EQL 0)
4                   THEN
5                       BEGIN                                   ! Begin DEFAULT
5
5                       CHARS_READ = .DEFAULT [DSC$W_LENGTH];
5                       YES_DEFAULT = 1 ;
5
5                       !+
5                       !   Protection check for DEFAULT excluding the Floating
5                       !   Point data types ( these will be handled in routine
5                       !   COB$$VERIFY_FL_RANGE ).
5                       !-
5
```

```
6                   IF (.YES_PROTECT AND
7                               ( .STRING_DEST [DSC$B_DTYPE] NEQ DSC$K_DTYPE_F AND
6                                 .STRING_DEST [DSC$B_DTYPE] NEQ DSC$K_DTYPE_D ))
5                   THEN
5                       !+
5                       !  If the length of DEFAULT is greater than the expected
5                       !  input size ACC_SIZE, then there is a Protection error.
5                       !  This should be caught at compile time by VAX COBOL and
5                       !  a fatal error message issued, however there is one case
5                       !  that the compiler cannot catch, therefore issue a fatal
5                       !  run time error here.
5                       !-
6                       IF (.DEFAULT [DSC$W_LENGTH] GTR .ACC_SIZE)
5                       THEN
5                           LIB$STOP ( COB$_INVDEFVAL )
5                       ELSE
5                           PROT_OK = 1                             ! No PROTECT error
5                   ELSE
5                       PROT_OK = 1 ;                               ! No PROTECT error
5
4                   END ;                                           ! End DEFAULT
4
4   !*******
4   !******* CONVERSION
4   !******* ALL RMS $GETs COMPLETED EXCEPT POSSIBLE REPROMPT ON A CONVERSION ERROR.
4   !*******
4
4   !+
4   !  If conversion requested, call routine COB$$ACC_CONVERT
4   !-
4
5           IF ( .PROT_OK )                                         ! If protection error,
4           THEN                                                    ! don't go thru conversion
5               IF ( .YES_CONV )
4               THEN
4                   CONV_OK = COB$$ACC_CONVERT ( .STRING_DEST, .FLAGS,
4                                                   .DEFAULT, PUT_HERE, .CHARS_READ,
4                                                   .YES_DEFAULT, .YES_SIGN )
4               ELSE
5                   BEGIN
5                       LOCAL
5                           COPY_NUM ;
5                   !+
5                   !  No conversion requested - copy input data to STRING_DEST.
5                   !  Use STR$COPY_R because it BLANK fills.
5                   !-
5
5                       IF .CHARS_READ LSS .STRING_DEST[DSC$W_LENGTH]
5                       THEN
5                           COPY_NUM = .CHARS_READ
5                       ELSE
5                           COPY_NUM = .STRING_DEST[DSC$W_LENGTH] ;
5
5                       STR$COPY_R ( .STRING_DEST, COPY_NUM,
6                                       (IF .YES_DEFAULT
6                                       THEN .DEFAULT [DSC$A_POINTER]
5                                       ELSE .PUT_HERE [DSC$A_POINTER] )) ;
```

```
5
5
5               CONV_OK = 1 ;                                   ! set CONV_OK to success
4           END;
4
4       !+
4       !  Conversion completed - was it successful ?
4       !-
4
4       IF .CONV_OK EQL 0
4       THEN
4       !+
4       !  CONVERSION error.  Read UNIT parameter to determine what
4       !  to do.
4       !
4       !     Byte 2 of                            Conversion
4       !       UNIT                                 error
4       !
4       !        0                                   reprompt
4       !        1 ( at end )                        reprompt
4       !        2 ( on exception )                  return
4       !-
5       BEGIN                                           ! Begin conversion error
5           IF ( .FLAGS AND V_COB_RPG ) NEQ 0
5           THEN
5               !+
5               !  VAX RPG - return on a Conversion Error, ring bell
5               !  and clean up before exiting.
5               !-
6               BEGIN
6               COB$$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ;
6               COB$$RPG_CLEAN_UP ( .FLAGS ) ;
6               RETURN 0 ;
5               END ;
5
5           IF .UNIT [1] EQL 2
5           THEN
6               BEGIN
6               !+
6               !  Clean up before returning control to VAX COBOL.
6               !-
6               COB$$CLEAN_UP ( PARAMETERS, .FLAGS ) ;
6               RETURN 0 ;
6               END
5           ELSE
5               !+
5               !  Reprompt
5               !      - sound terminal bell,
5               !      - clear screen of all typed characters,
5               !      - reset cursor to original line/column position
5               !-
6               BEGIN
7               IF ((.FLAGS AND V_NO_ECHO) NEQ 0 )  OR  (.YES_DEFAULT )
6               THEN
7                   BEGIN                               ! No re-positioning
7                   COB$$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ;  ! necessary
7                   END
```

```
; 1750          3263  6                     ELSE
; 1751          3264  7                         BEGIN
; 1752          3265  7                         $ERROR_REPROMPT ;               ! Re-position
; 1753          3266  6                         END ;
; 1754          3267  6                     CONV_OK = 0 ;                       ! Signal Conversion error
; 1755          3268  6                     YES_DEFAULT = 0 ;
; 1756          3269  5                     END ;
; 1757          3270  5             END                                         ! End conversion error
; 1758          3271  5
; 1759          3272  4         ELSE
; 1760          3273  4             !+
; 1761          3274  4             !  Conversion not done either because Protection failed (PROT_OK=0)
; 1762          3275  4             !  or there was no data to convert
; 1763          3276  4             !-
; 1764          3277  4             CONV_OK = 1 ;
; 1765          3278  4
; 1766          3279  3     END ;                                               ! End loop
; 1767          3280  3
; 1768          3281  3         !+
; 1769          3282  3         !  RMS $GET complete - fill in optional LENGTH parameter with the
; 1770          3283  3         !  number of characters read.
; 1771          3284  3         !-
; 1772          3285  3
; 1773          3286  3         IF NOT NULLPARAMETER (LENGTH)
; 1774          3287  3         THEN
; 1775          3288  3             .LENGTH = .CHARS_READ ;
; 1776          3289  3
; 1777          3290  3 !*******
; 1778          3291  3 !******* CLEAN UP
; 1779          3292  3 !*******
; 1780          3293  3
; 1781          3294  3 !+
; 1782          3295  3 !  Call COB$$CLEAN_UP to perform (if needed) cursor positioning,
; 1783          3296  3 !  turn off terminal attributes, and advancing.
; 1784          3297  3 !-
; 1785          3298  3
; 1786          3299  3     COB$$CLEAN_UP ( PARAMETERS, .FLAGS ) ;
; 1787          3300  3
; 1788          3301  2     END;                                                !  End $GET
; 1789          3302  2
; 1790          3303  2 !+
; 1791          3304  2 !  Free local strings PUT_HERE
; 1792          3305  2 !-
; 1793          3306  2
; 1794          3307  3     IF NOT ( STR$FREE1_DX ( PUT_HERE ))
; 1795          3308  2     THEN LIB$STOP ( COB$_ERRDURACC ) ;
; 1796          3309  2
; 1797          3310  2
; 1798          3311  2     RETURN 1;
; 1799          3312  1     END;                                        ! end of routine COB$ACC_SCR


                                                            001DE         .BLKB   2
                                                  00000000# 001E0 P.AAR:  .LONG   0[22]                                             ;
```

```
                                 0FFC 00000        .ENTRY  COB$ACC_SCR, Save R2,R3,R4,R5,R6,R7,R8,R9,-   ; 2181
                                                           R10,R11
                 5E      EBF0   CE 9E 00002        MOVAB   -5136(SP), SP                                 ;
                           0C   AE D4 00007        CLRL    ON_LEN                                        ; 2271
                                5A D4 0000A        CLRL    PROT_OK                                       ;
                                7E 7C 0000C        CLRQ    REPROMPT_DONE                                 ;
                                7E 7C 0000E        CLRQ    YES_NO_ECHO                                   ;
                                01 DD 00010        PUSHL   #1                                            ;
                                56 D4 00012        CLRL    P_DATA_TYPE                                   ;
  94   AD   8E   AF      0058   8F 28 00014        MOVC3   #88, P.AAR, PARAMETERS                        ; 2297
                                50 D4 0001C        CLRL    ZEROES                                        ; 2312
            24   AE             20 D0 0001E        MOVL    #32, BLANKS                                   ; 2313
            94   AD  02100000   8F D0 00022        MOVL    #34603008, PUT_HERE                           ; 2319
                           98   AD D4 0002A        CLRL    PUT_HERE+4                                    ; 2322
                 59        0C   AC D0 0002D        MOVL    FLAGS, R9                                     ; 2337
       04        59             05 E1 00031        BBC     #5, R9, 1$                                    ;
            08   AE             01 D0 00035        MOVL    #1, YES_CONV                                  ;
                                59 95 00039 1$:    TSTB    R9                                            ; 2338
                                02 18 0003B        BGEQ    2$                                            ;
                                6E D4 0003D        CLRL    YES_SIGN                                      ;
       03        59             08 E0 0003F 2$:    BBS     #8, R9, 3$                                    ; 2340
                              00FC 31 00043        BRW     20$                                           ;
            D0   AD             01 D0 00046 3$:    MOVL    #1, YES_PROTECT                               ; 2343
                           14   AC D5 0004A        TSTL    SIZE                                          ; 2344
                                08 13 0004D        BEQL    5$                                            ;
            AC   AD        14   AC B0 0004F        MOVW    SIZE, ACC_SIZE                                ; 2345
                              00F1 31 00054 4$:    BRW     21$                                           ;
                                52 D4 00057 5$:    CLRL    PP99                                          ; 2347
                 50        08   AC D0 00059        MOVL    STRING_DEST, R0                               ; 2352
                 52        09   A0 9A 0005D        MOVZBL  9(R0), PP99                                   ;
                 51        08   A0 98 00061        CVTBL   8(R0), R1                                     ;
                 52             51 C0 00065        ADDL2   R1, PP99                                      ;
            AC   AD             60 B0 00068        MOVW    (R0), ACC_SIZE                                ; 2353
                                54 D4 0006C        CLRL    R4                                            ; 2361
                 09        03   A0 91 0006E        CMPB    3(R0), #9                                     ;
                                6B 12 00072        BNEQ    13$                                           ;
                                54 D6 00074        INCL    R4                                            ;
                                52 D5 00076        TSTL    PP99                                          ; 2362
                                05 19 00078        BLSS    6$                                            ;
                           08   A0 95 0007A        TSTB    8(R0)                                         ; 2363
                                60 15 0007D        BLEQ    13$                                           ;
                 56             01 D0 0007F 6$:    MOVL    #1, P_DATA_TYPE                               ; 2366
                                53 D4 00082        CLRL    R3                                            ; 2367
                                52 D5 00084        TSTL    PP99                                          ;
                                11 18 00086        BGEQ    8$                                            ;
                                53 D6 00088        INCL    R3                                            ;
                 51        08   A0 98 0008A        CVTBL   8(R0), R1                                     ; 2369
                                03 18 0008E        BGEQ    7$                                            ;
                 51             51 CE 00090        MNEGL   R1, R1                                        ;
            AC   AD             51 B0 00093 7$:    MOVW    R1, ACC_SIZE                                  ;
                                04 11 00097        BRB     9$                                            ;
            AC   AD             52 B0 00099 8$:    MOVW    PP99, ACC_SIZE                                ; 2371
                 B3        08   AE E9 0009D 9$:    BLBC    YES_CONV, 4$                                  ; 2373
                 03             53 E9 000A1        BLBC    R3, 10$                                       ; 2379
                           AC   AD B6 000A4        INCW    ACC_SIZE                                      ; 2381
                 51        02   A0 9A 000A7 10$:   MOVZBL  2(R0), R1                                     ; 2389
```

```
                  0F            51  91 000AB         CMPB    R1, #15                          ;
                                03  13 000AE         BEQL    11$                              ;
                         AC     AD  B6 000B0         INCW    ACC_SIZE                         ; 2391
                  07            51  91 000B3  11$:   CMPB    R1, #7                           ; 2397
                                1E  13 000B6         BEQL    12$                              ;
                  03            51  91 000B8         CMPB    R1, #3                           ; 2398
                                19  13 000BB         BEQL    12$                              ;
                  08            51  91 000BD         CMPB    R1, #8                           ; 2399
                                14  13 000C0         BEQL    12$                              ;
                  04            51  91 000C2         CMPB    R1, #4                           ; 2400
                                0F  13 000C5         BEQL    12$                              ;
                  09            51  91 000C7         CMPB    R1, #9                           ; 2401
                                0A  13 000CA         BEQL    12$                              ;
                  05            51  91 000CC         CMPB    R1, #5                           ; 2402
                                05  13 000CF         BEQL    12$                              ;
                  15            51  91 000D1         CMPB    R1, #21                          ; 2403
                                72  12 000D4         BNEQ    21$                              ;
                                6E  D5 000D6  12$:   TSTL    YES_SIGN                         ; 2404
                                6E  12 000D8         BNEQ    21$                              ;
                         AC     AD  B7 000DA         DECW    ACC_SIZE                         ; 2406
                                69  11 000DD         BRB     21$                              ; 2361
                  65     08     AE  E9 000DF  13$:   BLBC    YES_CONV, 21$                    ; 2415
                  51     02     A0  9A 000E3         MOVZBL  2(R0), R1                        ; 2423
                  13            51  91 000E7         CMPB    R1, #19                          ;
                                0D  13 000EA         BEQL    14$                              ;
                  11            51  91 000EC         CMPB    R1, #17                          ; 2424
                                08  13 000EF         BEQL    14$                              ;
                  15            51  91 000F1         CMPB    R1, #21                          ; 2425
                                06  12 000F4         BNEQ    15$                              ;
                  03            6E  E9 000F6         BLBC    YES_SIGN, 15$                    ; 2426
                         AC     AD  B6 000F9  14$:   INCW    ACC_SIZE                         ; 2428
                  29            54  E9 000FC  15$:   BLBC    R4, 17$                          ; 2438
                  07            51  91 000FF         CMPB    R1, #7                           ; 2440
                                19  13 00102         BEQL    16$                              ;
                  03            51  91 00104         CMPB    R1, #3                           ; 2441
                                14  13 00107         BEQL    16$                              ;
                  08            51  91 00109         CMPB    R1, #8                           ; 2442
                                0F  13 0010C         BEQL    16$                              ;
                  04            51  91 0010E         CMPB    R1, #4                           ; 2443
                                0A  13 00111         BEQL    16$                              ;
                  09            51  91 00113         CMPB    R1, #9                           ; 2444
                                05  13 00116         BEQL    16$                              ;
                  05            51  91 00118         CMPB    R1, #5                           ; 2445
                                0B  12 0011B         BNEQ    17$                              ;
                  08            6E  E9 0011D  16$:   BLBC    YES_SIGN, 17$                    ; 2446
           AC     AD     09     A0  9B 00120         MOVZBW  9(R0), ACC_SIZE                  ; 2448
                         AC     AD  B6 00125         INCW    ACC_SIZE                         ;
                  0A            51  91 00128  17$:   CMPB    R1, #10                          ; 2454
                                04  12 0012B         BNEQ    18$                              ;
           AC     AD            0D  B0 0012D         MOVW    #13, ACC_SIZE                    ; 2456
                  0B            51  91 00131  18$:   CMPB    R1, #11                          ; 2457
                                04  12 00134         BNEQ    19$                              ;
           AC     AD            16  B0 00136         MOVW    #22, ACC_SIZE                    ; 2459
                  0B            54  E9 0013A  19$:   BLBC    R4, 21$                          ; 2465
                         AC     AD  B6 0013D         INCW    ACC_SIZE                         ; 2467
                                06  11 00140         BRB     21$                              ; 2340
           AC     AD   03F2     8F  B0 00142  20$:   MOVW    #1010, ACC_SIZE                  ; 2473
```

```
               50        AC AD B0 00148 21$:  MOVW    ACC_SIZE, PUT_SIZE                      ; 2481
          0398 8F        AC AD B1 0014C       CMPW    ACC_SIZE, #920                          ; 2482
                            05 1E 00152       BGEQU   22$                                     ;
  50        AC AD           05 A1 00154       ADDW3   #5, ACC_SIZE, PUT_SIZE                  ;
                         94 AD 9F 00159 22$:  PUSHAB  PUT_HERE                                ; 2484
            20 AE           50 3C 0015C       MOVZWL  PUT_SIZE, 32(SP)                        ;
                         20 AE 9F 00160       PUSHAB  32(SP)                                  ;
     00000000G 00           02 FB 00163       CALLS   #2, STR$GET1_DX                         ;
               0D           50 E8 0016A       BLBS    R0, 23$                                 ;
                  00000000G 8F DD 0016D       PUSHL   #COB$_ERRDURACC                         ; 2485
     00000000G 00           01 FB 00173       CALLS   #1, LIB$STOP                            ;
               52        04 AC 9A 0017A 23$:  MOVZBL  UNIT, R2                                ; 2493
               06           52 91 0017E       CMPB    R2, #6                                  ;
                            0D 1B 00181       BLEQU   24$                                     ;
                  00000000G 8F DD 00183       PUSHL   #COB$_INVARG                            ; 2495
     00000000G 00           01 FB 00189       CALLS   #1, LIB$STOP                            ;
            14 AE 00000000G0042 DE 00190 24$: MOVAL   COB$$AL_WRITE_RAB[R2], 20(SP)           ; 2497
                         14 BE D5 00199       TSTL    @20(SP)                                 ;
                            11 12 0019C       BNEQ    27$                                     ;
  04           59           0B E1 0019E       BBC     #11, R9, 25$                            ; 2504
                            01 DD 001A2       PUSHL   #1                                      ;
                            02 11 001A4       BRB     26$                                     ;
                            7E D4 001A6 25$:  CLRL    -(SP)                                   ;
                            52 DD 001A8 26$:  PUSHL   R2                                      ; 2503
         0000V CF           02 FB 001AA       CALLS   #2, COB$$OPEN_IN                        ;
               57        14 BE D0 001AF 27$:  MOVL    @20(SP), RAB                            ; 2508
               50        44 A7 9E 001B3       MOVAB   68(R7), NAM_DSC                         ; 2519
                  00000000G 00 D5 001B7       TSTL    COB$ACC_TERM_TYPE                       ; 2521
                            23 12 001BD       BNEQ    28$                                     ;
                  00000000G 00 9F 001BF       PUSHAB  COB$ACC_TERM_TYPE                       ; 2523
               7E           60 3C 001C5       MOVZWL  (NAM_DSC), -(SP)                        ; 2524
                         04 A0 DD 001C8       PUSHL   4(NAM_DSC)                              ; 2523
     00000000G 00           03 FB 001CB       CALLS   #3, COB$$SETUP_TERM_TYPE                ;
               0D           50 E8 001D2       BLBS    R0, 28$                                 ;
                  00000000G 8F DD 001D5       PUSHL   #COB$_ERRDURACC                         ; 2526
     00000000G 00           01 FB 001DB       CALLS   #1, LIB$STOP                            ;
                  00000000G 00 D5 001E2 28$:  TSTL    COB$ACC_TERM_TYPE                       ; 2528
                            46 12 001E8       BNEQ    31$                                     ;
                            6E DD 001EA       PUSHL   YES_SIGN                                ; 2539
                         D0 AD DD 001EC       PUSHL   YES_PROTECT                             ;
                         10 AE DD 001EF       PUSHL   YES_CONV                                ; 2538
                         94 AD 9F 001F2       PUSHAB  PUT_HERE                                ; 2537
               7E        AC AD 3C 001F5       MOVZWL  ACC_SIZE, -(SP)                         ; 2538
                         1C AC DD 001F9       PUSHL   LENGTH                                  ;
                         10 AC DD 001FC       PUSHL   DEFAULT                                 ; 2537
                            59 DD 001FF       PUSHL   R9                                      ;
               7E        04 AC 7D 00201       MOVQ    UNIT, -(SP)                             ;
         0000V CF           0A FB 00205       CALLS   #10, COB$$ACC_SCR_FILE                  ;
               52           50 D0 0020A       MOVL    R0, STATUS                              ;
                         94 AD 9F 0020D       PUSHAB  PUT_HERE                                ; 2543
     00000000G 00           01 FB 00210       CALLS   #1, STR$FREE1_DX                        ;
               0D           50 E8 00217       BLBS    R0, 29$                                 ;
                  00000000G 8F DD 0021A       PUSHL   #COB$_ERRDURACC                         ; 2544
     00000000G 00           01 FB 00220       CALLS   #1, LIB$STOP                            ;
               03           52 E8 00227 29$:  BLBS    STATUS, 30$                             ; 2546
                          0542 31 0022A       BRW     106$                                    ;
                          053B 31 0022D 30$:  BRW     105$                                    ;
                          00000000'   EF                01  D0 00230 31$:   MOVL    #1, ACC_SCR                             ; 2560
                                      50  00000000G     00  9A 00237        MOVZBL  COB$$AB_PREV, R0                        ; 2575
                                                        0A  13 0023E        BEQL    32$                                     ;
                                      02                50  91 00240        CMPB    R0, #2                                  ; 2576
                                                        05  13 00243        BEQL    32$                                     ;
                                      04                50  91 00245        CMPB    R0, #4                                  ; 2577
                                                        09  12 00248        BNEQ    33$                                     ;
                                                        59  DD 0024A 32$:   PUSHL   R9                                      ; 2582
                                                        01  DD 0024C        PUSHL   #1                                      ;
                              0000V   CF                02  FB 0024E        CALLS   #2, COB$$RMS_PUT_BYTE                   ;
    D8    AD          59              04                00  EF 00253 33$:   EXTZV   #0, #4, R9, PUT_FLAG                    ; 2591
                                                        2C  13 00259        BEQL    34$                                     ; 2593
                                                 E8     AD  9F 0025B        PUSHAB  OFF_LEN                                 ; 2597
                                                 DC     AD  9F 0025E        PUSHAB  OFF_BUF                                 ;
                                                 28     AE  9F 00261        PUSHAB  ON_LEN                                  ; 2596
                                                 EC     AD  9F 00264        PUSHAB  ON_BUF                                  ;
                                                 D8     AD  DD 00267        PUSHL   PUT_FLAG                                ; 2595
                                          00000000G     00  DD 0026A        PUSHL   COB$ACC_TERM_TYPE                       ;
                          00000000G   00                06  FB 00270        CALLS   #6, COB$$SET_ATTRIBUTES_ONLY            ;
                                      0D                50  E8 00277        BLBS    R0, 34$                                 ;
                                          00000000G     8F  DD 0027A        PUSHL   #COB$_ERRDURACC                         ; 2598
                          00000000G   00                01  FB 00280        CALLS   #1, LIB$STOP                            ;
                      0C              59                04  E1 00287 34$:   BBC     #4, R9, 35$                             ; 2604
                                      50         EC     AD  9E 0028B        MOVAB   ON_BUF, R0                              ; 2607
                                 20 BE40                07  90 0028F        MOVB    #7, @ON_LEN[R0]                         ;
                                                 20     AE  D6 00294        INCL    ON_LEN                                  ; 2608
                                      06                6C  91 00297 35$:   CMPB    (AP), #6                                ; 2618
                                                        58  1F 0029A        BLSSU   41$                                     ;
                                                 18     AC  D5 0029C        TSTL    24(AP)                                  ;
                                                        53  13 0029F        BEQL    41$                                     ;
                                 28   AE         18     BC  3C 002A1        MOVZWL  @KEY, KEY_LEN                           ; 2624
                                                 24     AE  9F 002A6        PUSHAB  BLANKS                                  ; 2625
                                                 2C     AE  9F 002A9        PUSHAB  KEY_LEN                                 ;
                                                 18     AC  DD 002AC        PUSHL   KEY                                     ;
                          00000000G   00                03  FB 002AF        CALLS   #3, STR$DUPL_CHAR                       ;
                                      07                6C  91 002B6        CMPB    (AP), #7                                ; 2632
                                                        05  1F 002B9        BLSSU   36$                                     ;
                                                 1C     AC  D5 002BB        TSTL    28(AP)                                  ;
                                                        34  12 002BE        BNEQ    41$                                     ;
                                      05                6C  91 002C0 36$:   CMPB    (AP), #5                                ; 2633
                                                        05  1F 002C3        BLSSU   37$                                     ;
                                                 14     AC  D5 002C5        TSTL    20(AP)                                  ;
                                                        2A  12 002C8        BNEQ    41$                                     ;
                                      04                6C  91 002CA 37$:   CMPB    (AP), #4                                ; 2634
                                                        05  1F 002CD        BLSSU   38$                                     ;
                                                 10     AC  D5 002CF        TSTL    16(AP)                                  ;
                                                        20  12 002D2        BNEQ    41$                                     ;
                                      02                6C  91 002D4 38$:   CMPB    (AP), #2                                ; 2635
                                                        05  1F 002D7        BLSSU   39$                                     ;
                                                 08     AC  D5 002D9        TSTL    8(AP)                                   ;
                                                        16  12 002DC        BNEQ    41$                                     ;
                                                 18     AC  DD 002DE 39$:   PUSHL   KEY                                     ; 2637
                                                        59  DD 002E1        PUSHL   R9                                      ;
                                                 04     AC  DD 002E3        PUSHL   UNIT                                    ;
                              0000V   CF                03  FB 002E6        CALLS   #3, COB$$FORMAT_FOUR                    ;
                                      03                50  E8 002EB        BLBS    R0, 40$                                 ;
                                                      047E  31 002EE        BRW     106$                                    ;
```

```
                                                0477  31 002F1 40$:   BRW     105$                                  ;
                                            18    AE  D4 002F4 41$:   CLRL    24(SP)                                ; 2653
                  0F              59              09  E1 002F7        BBC     #9, R9, 42$                           ;
                                            18    AE  D6 002FB        INCL    24(SP)                                ;
                            B4    AD      5240    8F  3C 002FE        MOVZWL  #21056, FUNC_VAL                      ; 2657
                            04    AE              01  D0 00304        MOVL    #1, YES_NO_ECHO                       ; 2658
                                                  06  11 00308        BRB     43$                                   ; 2653
                            B4    AD      5200    8F  3C 0030A 42$:   MOVZWL  #20992, FUNC_VAL                      ; 2661
                                                  5A  D5 00310 43$:   TSTL    PROT_OK                               ; 2671
                                                  08  13 00312        BEQL    44$                                   ;
                                            10    AE  D5 00314        TSTL    CONV_OK                               ;
                                                  03  13 00317        BEQL    44$                                   ;
                                                041C  31 00319        BRW     103$                                  ;
                                                  58  D4 0031C 44$:   CLRL    TERM_SEEN                             ; 2672
                                            0C    AE  D5 0031E        TSTL    REPROMPT_DONE                         ; 2676
                                                  03  13 00321        BEQL    45$                                   ;
                                                008D  31 00323        BRW     49$                                   ;
                                  03 00000000G    00  D1 00326 45$:   CMPL    COB$ACC_TERM_TYPE, #3                 ; 2688
                                                  55  12 0032D        BNEQ    47$                                   ;
                                  51        D0    AD  E9 0032F        BLBC    YES_PROTECT, 47$                      ; 2691
            FE68  CD        EC    AD        20    AE  28 00333        MOVC3   ON_LEN, ON_BUF, FIELD_BUF             ; 2706
                                  56        20    AE  D0 0033B        MOVL    ON_LEN, FIELD_LEN                     ; 2707
  AC    AD        20              6E              00  2C 0033F        MOVC5   #0, (SP), #32, ACC_SIZE, FIELD_BUF-   ; 2708
                                          FE68  CD46     00345                [FIELD_LEN]                           ;
                                  50        AC    AD  3C 00349        MOVZWL  ACC_SIZE, R0                          ; 2709
                                  56              50  C0 0034D        ADDL2   R0, FIELD_LEN                         ;
  AC    AD        08              6E              00  2C 00350        MOVC5   #0, (SP), #8, ACC_SIZE, FIELD_BUF-    ; 2710
                                          FE68  CD46     00356                [FIELD_LEN]                           ;
                                  50        AC    AD  3C 0035A        MOVZWL  ACC_SIZE, R0                          ; 2711
                                  56              50  C0 0035E        ADDL2   R0, FIELD_LEN                         ;
                      000003F2    8F              56  D1 00361        CMPL    FIELD_LEN, #1010                      ; 2721
                                                  0D  15 00368        BLEQ    46$                                   ;
                                     00000000G    8F  DD 0036A        PUSHL   #COB$_ERRDURACC                       ; 2723
                     00000000G    00              01  FB 00370        CALLS   #1, LIB$STOP                          ;
                                          0240    8F  BB 00377 46$:   PUSHR   #^M<R6,R9>                            ; 2729
                                          FE68    CD  9F 0037B        PUSHAB  FIELD_BUF                             ;
                         0000V    CF              03  FB 0037F        CALLS   #3, COB$$RMS_PUT_BUFFER               ;
                                            20    AE  D5 00384 47$:   TSTL    ON_LEN                                ; 2749
                                                  13  13 00387        BEQL    48$                                   ;
                                  01        D0    AD  D1 00389        CMPL    YES_PROTECT, #1                       ;
                                                  0D  13 0038D        BEQL    48$                                   ;
                                                  59  DD 0038F        PUSHL   R9                                    ; 2751
                                            24    AE  DD 00391        PUSHL   ON_LEN                                ;
                                            EC    AD  9F 00394        PUSHAB  ON_BUF                                ;
                         0000V    CF              03  FB 00397        CALLS   #3, COB$$RMS_PUT_BUFFER               ;
                                  57        14    BE  D0 0039C 48$:   MOVL    @20(SP), RAB                          ; 2757
                                            98    AD  DD 003A0        PUSHL   PUT_HERE+4                            ; 2759
                                  7E        AC    AD  3C 003A3        MOVZWL  ACC_SIZE, -(SP)                       ; 2758
                                            B4    AD  DD 003A7        PUSHL   FUNC_VAL                              ;
                                                  57  DD 003AA        PUSHL   RAB                                   ;
                         0000V    CF              04  FB 003AC        CALLS   #4, COB$$RMS_GET                      ;
                                                  03  11 003B1        BRB     50$                                   ; 2676
                                            0C    AE  D4 003B3 49$:   CLRL    REPROMPT_DONE                         ; 2763
                            B0    AD        22    A7  3C 003B6 50$:   MOVZWL  34(RAB), CHARS_READ                   ; 2776
                            B8    AD        0E    A7  9A 003BB        MOVZBL  14(RAB), TERM_SIZE                    ; 2777
                            BC    AD        0C    A7  9A 003C0        MOVZBL  12(RAB), TERM_LOC                     ; 2778
                      000181C8    8F        08    A7  D1 003C5        CMPL    8(RAB), #98760                        ; 2787
```

```
                            0E 12 003CD         BNEQ    51$                                     ;
               58           01 D0 003CF         MOVL    #1, TERM_SEEN                           ; 2790
                       04   AC DD 003D2         PUSHL   UNIT                                    ; 2791
                       94   AD 9F 003D5         PUSHAB  PARAMETERS                              ;
       0000V   CF           02 FB 003D8         CALLS   #2, COB$$PARTIAL_SEQ                    ;
          7F   8F      0C   A7 91 003DD 51$:    CMPB    12(RAB), #127                           ; 2798
                            0D 12 003E2         BNEQ    52$                                     ;
                            59 DD 003E4         PUSHL   R9                                      ; 2800
                       04   AC DD 003E6         PUSHL   UNIT                                    ;
                       94   AD 9F 003E9         PUSHAB  PARAMETERS                              ;
       0000V   CF           03 FB 003EC         CALLS   #3, COB$$DELETE_KEY                     ;
               03      D0   AD E8 003F1 52$:    BLBS    YES_PROTECT, 54$                        ; 2825
                          00A8 31 003F5 53$:    BRW     63$                                     ;
                       B0   AD D5 003F8 54$:    TSTL    CHARS_READ                              ; 2826
                            F8 13 003FB         BEQL    53$                                     ;
    000181B8   8F      08   A7 D1 003FD         CMPL    8(RAB), #98744                          ; 2828
                            EE 12 00405         BNEQ    53$                                     ;
                            58 D5 00407         TSTL    TERM_SEEN                               ; 2829
                            EA 12 00409         BNEQ    53$                                     ;
                            53 D4 0040B         CLRL    NO_CHAR                                 ; 2831
                            52 D4 0040D 55$:    CLRL    HAVE_TERM                               ;
               01           52 D1 0040F 56$:    CMPL    HAVE_TERM, #1                           ; 2850
                            03 12 00412         BNEQ    57$                                     ;
                          008C 31 00414         BRW     64$                                     ;
                            53 D4 00417 57$:    CLRL    NO_CHAR                                 ; 2853
          1C   AE    5240   8F 3C 00419         MOVZWL  #2T056, FUNC_VAL_2                      ; 2855
               57      14   BE D0 0041F         MOVL    @20(SP), RAB                            ; 2857
                       A0   AD 9F 00423         PUSHAB  NEXT_CHAR                               ; 2858
                            01 DD 00426         PUSHL   #1                                      ;
                       24   AE DD 00428         PUSHL   FUNC_VAL_2                              ;
                            57 DD 0042B         PUSHL   RAB                                     ;
       0000V   CF           04 FB 0042D         CALLS   #4, COB$$RMS_GET                        ;
                       22   A7 B5 00432         TSTW    34(RAB)                                 ; 2868
                            08 12 00435         BNEQ    58$                                     ;
               53           01 D0 00437         MOVL    #1, NO_CHAR                             ; 2871
          A0   AD      0C   A7 90 0043A         MOVB    12(RAB), NEXT_CHAR                      ; 2872
          B8   AD      0E   A7 9A 0043F 58$:    MOVZBL  14(RAB), TERM_SIZE                      ; 2874
          BC   AD      0C   A7 9A 00444         MOVZBL  12(RAB), TERM_LOC                       ; 2875
          C4   AD           01 D0 00449         MOVL    #1, TERM_IN_NEXT                        ; 2876
    000181C8   8F      08   A7 D1 0044D         CMPL    8(RAB), #98760                          ; 2877
                            0B 12 00455         BNEQ    59$                                     ;
                       04   AC DD 00457         PUSHL   UNIT                                    ; 2879
                       94   AD 9F 0045A         PUSHAB  PARAMETERS                              ;
       0000V   CF           02 FB 0045D         CALLS   #2, COB$$PARTIAL_SEQ                    ;
               2D           53 E9 00462 59$:    BLBC    NO_CHAR, 62$                            ; 2886
               5A           01 D0 00465         MOVL    #1, PROT_OK                             ; 2889
               52           01 D0 00468         MOVL    #1, HAVE_TERM                           ; 2890
          7F   8F      0C   A7 91 0046B         CMPB    12(RAB), #127                           ; 2901
                            9D 12 00470         BNEQ    56$                                     ;
                            59 DD 00472         PUSHL   R9                                      ; 2904
                       04   AC DD 00474         PUSHL   UNIT                                    ;
                       94   AD 9F 00477         PUSHAB  PARAMETERS                              ;
       0000V   CF           03 FB 0047A         CALLS   #3, COB$$DELETE_KEY                     ;
                       B8   AD D5 0047F         TSTL    TERM_SIZE                               ; 2910
                            06 12 00482         BNEQ    61$                                     ;
                            52 D4 00484         CLRL    HAVE_TERM                               ; 2913
                            5A D4 00486         CLRL    PROT_OK                                 ; 2914
```

```
                                                       85  11 00488 60$:    BRB     56$                                     ; 2910
                                          52           01  D0 0048A 61$:    MOVL    #1, HAVE_TERM                           ; 2918
                                                  C4   AD  D4 0048D         CLRL    TERM_IN_NEXT                            ; 2919
                                                       F6  11 00490         BRB     60$                                     ; 2901
                                                       59  DD 00492 62$:    PUSHL   R9                                      ; 2939
                                                       02  DD 00494         PUSHL   #2                                      ;
                                  0000V   CF           02  FB 00496         CALLS   #2, COB$$RMS_PUT_BYTE                   ;
                                                       5A  D4 0049B         CLRL    PROT_OK                                 ; 2940
                                                     FF6D  31 0049D         BRW     55$                                     ; 2941
                                          5A           01  D0 004A0 63$:    MOVL    #1, PROT_OK                             ; 2956
                                          6B           5A  E9 004A3 64$:    BLBC    PROT_OK, 70$                            ; 2962
                                          07      C4   AD  E9 004A6         BLBC    TERM_IN_NEXT, 65$                       ; 2969
                                     C0   AD      A0   AD  9E 004AA         MOVAB   NEXT_CHAR, TERM_PTR                     ; 2971
                                                       07  11 004AF         BRB     66$                                     ;
                      C0    AD       98   AD      B0   AD  C1 004B1 65$:    ADDL3   CHARS_READ, PUT_HERE+4, TERM_PTR        ; 2973
                                          01      B8   AD  D1 004B8 66$:    CMPL    TERM_SIZE, #1                           ; 2990
                                                       5D  12 004BC         BNEQ    72$                                     ;
                                     C0   AD      0C   A7  9E 004BE         MOVAB   12(RAB), TERM_PTR                       ; 2993
                                          50      0C   A7  9A 004C3         MOVZBL  12(RAB), R0                             ; 2994
                                          09           50  91 004C7         CMPB    R0, #9                                  ; 2996
                                                       05  13 004CA         BEQL    67$                                     ;
                                          0D           50  91 004CC         CMPB    R0, #13                                 ;
                                                       15  12 004CF         BNEQ    68$                                     ;
                                          06           6C  91 004D1 67$:    CMPB    (AP), #6                                ; 3001
                                                       3B  1F 004D4         BLSSU   70$                                     ;
                                                  18   AC  D5 004D6         TSTL    24(AP)                                  ;
                                                       36  13 004D9         BEQL    70$                                     ;
                                          50      18   AC  D0 004DB         MOVL    KEY, R0                                 ; 3003
                                     04   B0      C0   BD  90 004DF         MOVB    @TERM_PTR, @4(R0)                       ;
                                                       2B  11 004E4         BRB     70$                                     ; 3001
                                          1A           50  91 004E6 68$:    CMPB    R0, #26                                 ; 3005
                                                       13  12 004E9         BNEQ    69$                                     ;
                            24            59           0B  E0 004EB         BBS     #11, R9, 71$                            ; 3013
                                                       59  DD 004EF         PUSHL   R9                                      ; 3028
                                                  94   AD  9F 004F1         PUSHAB  PARAMETERS                              ;
                                  0000V   CF           02  FB 004F4         CALLS   #2, COB$$CLEAN_UP                       ;
                                                  18   AC  DD 004F9         PUSHL   KEY                                     ; 3029
                                                       40  11 004FC         BRB     73$                                     ;
                                     7F   8F           50  91 004FE 69$:    CMPB    R0, #127                                ; 3034
                                                       0F  12 00502         BNEQ    71$                                     ;
                                                       59  DD 00504         PUSHL   R9                                      ; 3037
                                                  04   AC  DD 00506         PUSHL   UNIT                                    ;
                                                  94   AD  9F 00509         PUSHAB  PARAMETERS                              ;
                                  0000V   CF           03  FB 0050C         CALLS   #3, COB$$DELETE_KEY                     ;
                                                       64  11 00511 70$:    BRB     77$                                     ; 2994
                                                  CC   AD  D4 00513 71$:    CLRL    LEGAL                                   ; 3043
                                                  18   AC  DD 00516         PUSHL   KEY                                     ; 3045
                                                       4F  11 00519         BRB     76$                                     ; 3044
                                          52      18   AC  D0 0051B 72$:    MOVL    KEY, R2                                 ; 3062
                                                  B0   AD  D5 0051F         TSTL    CHARS_READ                              ; 3051
                                                       25  12 00522         BNEQ    74$                                     ;
                               0001827A   8F      08   A7  D1 00524         CMPL    8(RAB), #98938                          ;
                                                       1B  12 0052C         BNEQ    74$                                     ;
                            33            59           0B  E0 0052E         BBS     #11, R9, 75$                            ; 3058
                                                       59  DD 00532         PUSHL   R9                                      ; 3066
                                                  94   AD  9F 00534         PUSHAB  PARAMETERS                              ;
                                  0000V   CF           02  FB 00537         CALLS   #2, COB$$CLEAN_UP                       ;
```

```
                           52  DD 0053C        PUSHL    R2                                ; 3067
                      04   AC  DD 0053E 73$:   PUSHL    UNIT                              ;
    0000V  CF              02  FB 00541        CALLS    #2, COB$$CONTROL_Z                ;
                         0226  31 00546        BRW      106$                              ; 3068
           06              6C  91 00549 74$:   CMPB     (AP), #6                          ; 3079
                           17  1F 0054C        BLSSU    75$                               ;
                      18   AC  D5 0054E        TSTL     24(AP)                            ;
                           12  13 00551        BEQL     75$                               ;
                           52  DD 00553        PUSHL    R2                                ; 3087
                      B8   AD  DD 00555        PUSHL    TERM_SIZE                         ;
                      C0   AD  9F 00558        PUSHAB   TERM_PTR                          ;
00000000G  00              03  FB 0055B        CALLS    #3, COB$$CONTROL_KEY              ;
           12              50  E8 00562        BLBS     R0, 77$                           ;
                      CC   AD  D4 00565 75$:   CLRL     LEGAL                             ; 3101
                           52  DD 00568        PUSHL    R2                                ; 3102
                           59  DD 0056A 76$:   PUSHL    R9                                ;
                      04   AC  DD 0056C        PUSHL    UNIT                              ;
                      94   AD  9F 0056F        PUSHAB   PARAMETERS                        ;
    0000V  CF              04  FB 00572        CALLS    #4, COB$$ILLEGAL_TERM             ;
                           52  D4 00577 77$:   CLRL     R2                                ; 3123
                      B0   AD  D5 00579        TSTL     CHARS_READ                        ;
                           10  12 0057C        BNEQ     78$                               ;
                           52  D6 0057E        INCL     R2                                ;
       0A  59              0B  E1 00580        BBC      #11, R9, 78$                      ;
                           59  DD 00584        PUSHL    R9                                ; 3130
    0000V  CF              01  FB 00586        CALLS    #1, COB$$RPG_CLEAN_UP             ;
                         01DD  31 0058B        BRW      105$                              ; 3131
           45              52  E9 0058E 78$:   BLBC     R2, 80$                           ; 3134
           04              6C  91 00591        CMPB     (AP), #4                          ; 3136
                           40  1F 00594        BLSSU    80$                               ;
                      10   AC  D5 00596        TSTL     16(AP)                            ;
                           3B  13 00599        BEQL     80$                               ;
                      D4   AD  D5 0059B        TSTL     YES_DEFAULT                       ;
                           36  12 0059E        BNEQ     80$                               ;
       B0  AD         10   BC  3C 005A0        MOVZWL   @DEFAULT, CHARS_READ              ; 3140
       D4  AD              01  D0 005A5        MOVL     #1, YES_DEFAULT                   ; 3141
           26         D0   AD  E9 005A9        BLBC     YES_PROTECT, 79$                  ; 3149
           50         08   AC  D0 005AD        MOVL     STRING_DEST, R0                   ; 3150
           0A         02   A0  91 005B1        CMPB     2(R0), #10                        ;
                           1C  13 005B5        BEQL     79$                               ;
           0B         02   A0  91 005B7        CMPB     2(R0), #11                        ; 3151
                           16  13 005BB        BEQL     79$                               ;
       AC  AD         10   BC  B1 005BD        CMPW     @DEFAULT, ACC_SIZE                ; 3161
                           0F  1B 005C2        BLEQU    79$                               ;
               00000000G   8F  DD 005C4        PUSHL    #COB$_INVDEFVAL                   ; 3163
00000000G  00              01  FB 005CA        CALLS    #1, LIB$STOP                      ;
                           03  11 005D1        BRB      80$                               ;
           5A              01  D0 005D3 79$:   MOVL     #1, PROT_OK                       ; 3167
           5A              5A  E9 005D6 80$:   BLBC     PROT_OK, 86$                      ; 3180
           20         08   AE  E9 005D9        BLBC     YES_CONV, 81$                     ; 3182
                           6E  DD 005DD        PUSHL    YES_SIGN                          ; 3186
                      D4   AD  DD 005DF        PUSHL    YES_DEFAULT                       ;
                      B0   AD  DD 005E2        PUSHL    CHARS_READ                        ; 3185
                      94   AD  9F 005E5        PUSHAB   PUT_HERE                          ; 3184
                      10   AC  DD 005E8        PUSHL    DEFAULT                           ; 3185
                           59  DD 005EB        PUSHL    R9                                ; 3184
                      08   AC  DD 005ED        PUSHL    STRING_DEST                       ;
```

```
                         00000000G   00          07 FB 005F0       CALLS   #7, COB$$ACC_CONVERT                     ;
                                10   AE          50 D0 005F7       MOVL    R0, CONV_OK                              ;
                                                 36 11 005FB       BRB     86$                                      ;
    B0   AD     08   BC              10          00 ED 005FD 81$:  CMPZV   #0, #16, @STRING_DEST, CHARS_READ        ; 3196
                                                 07 15 00604       BLEQ    82$                                      ;
                                2C   AE     B0   AD D0 00606       MOVL    CHARS_READ, COPY_NUM                     ; 3198
                                                 05 11 0060B       BRB     83$                                      ;
                                2C   AE     08   BC 3C 0060D 82$:  MOVZWL  @STRING_DEST, COPY_NUM                   ; 3200
                                     09     D4   AD E9 00612 83$:  BLBC    YES_DEFAULT, 84$                         ; 3203
                                     50     10   AC D0 00616       MOVL    DEFAULT, R0                              ; 3204
                                            04   A0 DD 0061A       PUSHL   4(R0)                                    ;
                                                 03 11 0061D       BRB     85$                                      ;
                                            98   AD DD 0061F 84$:  PUSHL   PUT_HERE+4                               ; 3205
                                            30   AE 9F 00622 85$:  PUSHAB  COPY_NUM                                 ; 3202
                                            08   AC DD 00625       PUSHL   STRING_DEST                              ;
                         00000000G   00          03 FB 00628       CALLS   #3, STR$COPY_R                           ;
                                10   AE          01 D0 0062F       MOVL    #1, CONV_OK                              ; 3208
                                            10   AE D5 00633 86$:  TSTL    CONV_OK                                  ; 3215
                                                 03 13 00636       BEQL    87$                                      ;
                                               00F6 31 00638       BRW     101$                                     ;
                     12              59          0B E1 0063B 87$:  BBC     #11, R9, 88$                             ; 3229
                                                 59 DD 0063F       PUSHL   R9                                       ; 3236
                                                 02 DD 00641       PUSHL   #2                                       ;
                             0000V   CF          02 FB 00643       CALLS   #2, COB$$RMS_PUT_BYTE                    ;
                                                 59 DD 00648       PUSHL   R9                                       ; 3237
                             0000V   CF          01 FB 0064A       CALLS   #1, COB$$RPG_CLEAN_UP                    ;
                                                 10 11 0064F       BRB     89$                                      ; 3238
                                     02     05   AC 91 00651 88$:  CMPB    UNIT+1, #2                               ; 3241
                                                 0D 12 00655       BNEQ    90$                                      ;
                                                 59 DD 00657       PUSHL   R9                                       ; 3247
                                            94   AD 9F 00659       PUSHAB  PARAMETERS                               ;
                             0000V   CF          02 FB 0065C       CALLS   #2, COB$$CLEAN_UP                        ;
                                               010B 31 00661 89$:  BRW     106$                                     ; 3248
                                     04     18   AE E8 00664 90$:  BLBS    24(SP), 91$                              ; 3258
                                     0C     D4   AD E9 00668       BLBC    YES_DEFAULT, 92$                         ;
                                                 59 DD 0066C 91$:  PUSHL   R9                                       ; 3261
                                                 02 DD 0066E       PUSHL   #2                                       ;
                             0000V   CF          02 FB 00670       CALLS   #2, COB$$RMS_PUT_BYTE                    ;
                                               00B1 31 00675       BRW     100$                                     ; 3258
                                                 56 D4 00678 92$:  CLRL    PUT_TOTAL                                ; 3264
                                                 58 D4 0067A       CLRL    INDEX                                    ;
                                                 59 DD 0067C       PUSHL   R9                                       ;
                                                 02 DD 0067E       PUSHL   #2                                       ;
                             0000V   CF          02 FB 00680       CALLS   #2, COB$$RMS_PUT_BYTE                    ;
                                            04   AE D5 00685       TSTL    YES_NO_ECHO                              ;
                                                 59 12 00688       BNEQ    96$                                      ;
                                                 5B D4 0068A       CLRL    R11                                      ;
                                            D8   AD D5 0068C       TSTL    PUT_FLAG                                 ;
                                                 12 13 0068F       BEQL    93$                                      ;
                                                 5B D6 00691       INCL    R11                                      ;
                                            D0   AD D5 00693       TSTL    YES_PROTECT                              ;
                                                 0B 12 00696       BNEQ    93$                                      ;
                30   AE         DC   AD     E8   AD 28 00698       MOVC3   OFF_LEN, OFF_BUF, RESTORE_CURSOR         ;
                                     56     E8   AD D0 0069F       MOVL    OFF_LEN, PUT_TOTAL                       ;
                                     58          56 D0 006A3 93$:  MOVL    PUT_TOTAL, INDEX                         ;
                     51              56     B0   AD C1 006A6       ADDL3   CHARS_READ, PUT_TOTAL, R1                ;
                                     50     FF   A6 9E 006AB       MOVAB   -1(R6), P                                ;
```

```
                                      12 11 006AF         BRB     95$                                           ;
                  30  AE48            08 90 006B1 94$:    MOVB    #8, RESTORE_CURSOR[INDEX]                     ;
                  31  AE48            20 90 006B6         MOVB    #32, RESTORE_CURSOR+1[INDEX]                  ;
                  32  AE48            08 90 006BB         MOVB    #8, RESTORE_CURSOR+2[INDEX]                   ;
                        58            03 C0 006C0         ADDL2   #3, INDEX                                     ;
     EA                 50            51 F2 006C3 95$:    AOBLSS  R1, P, 94$                                    ;
     50           B0    AD            03 C5 006C7         MULL3   #3, CHARS_READ, R0                            ;
                        56            50 C0 006CC         ADDL2   R0, PUT_TOTAL                                 ;
                        11            5B E9 006CF         BLBC    R11, 96$                                      ;
                                 D0   AD D5 006D2         TSTL    YES_PROTECT                                   ;
                                      0C 12 006D5         BNEQ    96$                                           ;
30 AE46           EC    AD       20   AE 28 006D7         MOVC3   ON_LEN, ON_BUF, RESTORE_CURSOR[PUT_TOTAL]     ;
                        56       20   AE C0 006DF         ADDL2   ON_LEN, PUT_TOTAL                             ;
                                      51 D4 006E3 96$:    CLRL    LAST_WRITE                                    ;
                                      51 D5 006E5 97$:    TSTL    LAST_WRITE                                    ;
                                      1B 12 006E7         BNEQ    99$                                           ;
            000003F2    8F            56 D1 006E9         CMPL    PUT_TOTAL, #1010                              ;
                                      0A 15 006F0         BLEQ    98$                                           ;
                        50     03F2   8F 3C 006F2         MOVZWL  #1010, P_TOT                                  ;
                        56            50 C2 006F7         SUBL2   P_TOT, PUT_TOTAL                              ;
                                      E9 11 006FA         BRB     97$                                           ;
                        50            56 D0 006FC 98$:    MOVL    PUT_TOTAL, P_TOT                              ;
                        51            01 D0 006FF         MOVL    #1, LAST_WRITE                                ;
                                      E1 11 00702         BRB     97$                                           ;
                               0201   8F BB 00704 99$:    PUSHR   #^M<R0,R9>                                    ;
                                 38   AE 9F 00708         PUSHAB  RESTORE_CURSOR                                ;
               0000V    CF            03 FB 0070B         CALLS   #3, COB$$RMS_PUT_BUFFER                       ;
                        57       14   BE D0 00710         MOVL    @20(SP), RAB                                  ;
                                 98   AD DD 00714         PUSHL   PUT_HERE+4                                    ;
                        7E       AC   AD 3C 00717         MOVZWL  ACC_SIZE, -(SP)                               ;
                                 B4   AD DD 0071B         PUSHL   FUNC_VAL                                      ;
                                      57 DD 0071E         PUSHL   RAB                                           ;
               0000V    CF            04 FB 00720         CALLS   #4, COB$$RMS_GET                              ;
                  0C    AE            01 D0 00725         MOVL    #1, REPROMPT_DONE                             ;
                                 10   AE D4 00729 100$:   CLRL    CONV_OK                                       ; 3267
                                 D4   AD D4 0072C         CLRL    YES_DEFAULT                                   ; 3268
                                      04 11 0072F         BRB     102$                                          ; 3215
                  10    AE            01 D0 00731 101$:   MOVL    #1, CONV_OK                                   ; 3277
                                    FBD8 31 00735 102$:   BRW     43$                                           ; 2671
                        07            6C 91 00738 103$:   CMPB    (AP), #7                                      ; 3286
                                      0A 1F 0073B         BLSSU   104$                                          ;
                                 1C   AC D5 0073D         TSTL    28(AP)                                        ;
                                      05 13 00740         BEQL    104$                                          ;
                  1C    BC       B0   AD D0 00742         MOVL    CHARS_READ, @LENGTH                           ; 3288
                                      59 DD 00747 104$:   PUSHL   R9                                            ; 3299
                                 94   AD 9F 00749         PUSHAB  PARAMETERS                                    ;
               0000V    CF            02 FB 0074C         CALLS   #2, COB$$CLEAN_UP                             ;
                                 94   AD 9F 00751         PUSHAB  PUT_HERE                                      ; 3307
           00000000G    00            01 FB 00754         CALLS   #1, STR$FREE1_DX                              ;
                        0D            50 E8 0075B         BLBS    R0, 105$                                      ;
                          00000000G   8F DD 0075E         PUSHL   #COB$_ERRDURACC                               ; 3308
           00000000G    00            01 FB 00764         CALLS   #1, LIB$STOP                                  ;
                        50            01 D0 0076B 105$:   MOVL    #1, R0                                        ; 3311
                                         04 0076E         RET                                                   ;
                                      50 D4 0076F 106$:   CLRL    R0                                            ; 3312
                                         04 00771         RET                                                   ;
```

```
; Routine Size:  1906 bytes,    Routine Base:  _COB$CODE + 0238
```

```
1 %SBTTL 'COB$ACC_SCR_FILE - Screen enhancements for files'
1 ROUTINE COB$$ACC_SCR_FILE ( UNIT          :  VECTOR [2,BYTE],
1                            STRING_DEST   :  REF $STR$DESCRIPTOR,
1                            FLAGS,
1                            DEFAULT       :  REF $STR$DESCRIPTOR,
1                            LENGTH,
1                            ACC_SIZE,
1                            PUT_HERE      :  REF BLOCK [8, BYTE],
1                                             ! Contains input characters
1                            YES_CONV,        ! =1 if conversion requested
1                            YES_PROTECT,     ! =1 if protection requested
1                            YES_SIGN         ! =1 if sign should be included
1                           ) =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine handles the VAX COBOL Version 3 ACCEPT statement
1 !       with Screen Enhancements when a file (not a terminal) is used
1 !       for input.  A non terminal $GET service does not contain all the
1 !       features of a terminal $GET service, so this routine is a scaled
1 !       down version of COB$ACC_SCR.  Note that the fields RAB [RAB$V_ETO]
1 !       and RAB [RAB$L_XAB] are not set.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       UNIT.rbu.va     Array of two unsigned byte integers.
1 !                       The first byte is the unit number designating the
1 !                       device from which the string is to be read.
1 !                       The second byte indicates whether the routine should
1 !                       abort or return to the calling program.
1 !                         Byte 2 = 0  -  routine will abort on control z
1 !                                        and reprompt on conversion errors.
1 !                                = 1  -  ( AT END )
1 !                                        routine will return to calling program
1 !                                        on control z and reprompt on conversion
1 !                                        errors.
1 !                                = 2  -  ( ON EXCEPTION )
1 !                                        routine will return to calling program
1 !                                        on control z and conversion errors.
1 !
1 !       STRING_DEST.mt.ds   Address of descriptor to receive the read input.
1 !
1 !       FLAGS.rlu.v     Screen enhancement flag;
1 !
1 !                               bit 0  -  bold
1 !                               bit 1  -  reverse
1 !                               bit 2  -  blink
1 !                               bit 3  -  underline
1 !                               bit 4  -  bell
1 !                               bit 5  -  conversion
1 !                               bit 6  -  decimal point is comma
1 !                               bit 7  -  0 to allow space for sign in PROTECTED
1 !                                         ACCEPT, 1 no allowance for sign
1 !                               bit 8  -  protect
1 !                               bit 9  -  no-echo
1 !                               bit 10 -  0 advancing, 1 no advancing
```

```
1 !                                 bit 11 -  0 for VAX COBOL, 1 for VAX RPG
1 !
1 !         DEFAULT.rt.dx    Default source moved to destination descriptor
1 !                          (STRING_DEST) in the event of null input.
1 !
1 !         LENGTH.wlu.r     Destination of the number of characters read.
1 !
1 !         ACC_SIZE.rlu.v   # of characters to RMS $GET.
1 !
1 !         PUT_HERE.rt.dx   Buffer to hold input characters.
1 !
1 !         YES_CONV.rlu.v   Flag = 1 if Conversion requested by user.
1 !
1 !         YES_PROTECT.rlu.v  Flag = 1 if Protection requested by user.
1 !
1 !         YES_SIGN.rlu.v   Flag = 1 if sign should be included in COMP or COMP3
1 !                          data type.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !         Status of whether the input file is currently open.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !         Updated status of file
1 !
1 ! ROUTINE VALUE:
1 !
1 !         If .UNIT[1] is false :  Unspecified.
1 !         If .UNIT[1] is true  :  Either true or false, indicating success or
1 !                                 EOF, respectively.
1 !
1 ! SIDE EFFECTS:
1 !
1 !         Reads a record from a designated uint.
1 !
1 !--
1
2      BEGIN
2
2      LOCAL
2          RAB              :  REF $RAB_DECL,
2          CR_BUF           :  VECTOR [T,BYTE],
2          CHARS_READ       :  INITIAL (0),          ! Number of characters read
2          CONV_OK          :  INITIAL (0),          ! = 1 if no conversion errors
2          YES_DEFAULT      :  INITIAL (0) ;         ! = 1 if DEFAULT was used as input
2
2      BUILTIN
2          NULLPARAMETER ;
2
2      !+
2      ! RMS $PUT - If previous call requires advancing, $PUT a linefeed to
2      ! SYS$OUTPUT.  Open SYS$OUTPUT if necessary.
2      !-
2
3      IF (.COB$$AB_PREV[0] EQL DISP
3          OR .COB$$AB_PREV[0] EQL POS
3           OR .COB$$AB_PREV[0] EQL ACC_ADV )
2       THEN
2           COB$$RMS_PUT_BYTE ( LINE_FD, .FLAGS ) ;
2
2       !+
2       !  RMS $GET to accept input from a file.
2       !-
2
2       RAB = .COB$$AL_WRITE_RAB [.UNIT[0]] ;
2       RAB [RAB$W_USZ] = .ACC_SIZE ;
2       RAB [RAB$L_UBF] = .PUT_HERE [DSC$A_POINTER] ;
2       !+
2       !  Turn off RAB [RAB$V_ETO] just in case a 'screen enhancement ACCEPT'
2       !  was performed before this one.
2       !-
2       RAB [RAB$V_ETO] = 0 ;
2
2       WHILE $GET (RAB = .RAB) EQL RMS$_RSA DO $WAIT (RAB = .RAB) ;
2
2       IF NOT .RAB [RAB$L_STS] AND NULLPARAMETER (DEFAULT)
2       THEN
3           LIB$STOP (( IF .RAB[RAB$L_STS] EQL RMS$_EOF
3                       THEN
3                           !+
3                           !  If ON EXCEPTION or AT END, return to user program.
3                           !-
3                           IF .UNIT [1] EQL 1 OR .UNIT [1] EQL 2
3                           THEN
3                               RETURN 0
3                           ELSE
3                               COB$_EOFON_ACC
3                       ELSE
2                              COB$_ERRDURACC),
2                    1, .RAB + RAB$C_BLN, .RAB [RAB$L_STS], .RAB [RAB$L_STV] ) ;
2
2       !+
2       !  Put number of characters read from $GET in CHARS_READ.
2       !  Pass this info along to COB$$ACC_CONVERT.
2       !-
2
2       CHARS_READ = .RAB [RAB$W_RSZ] ;                         ! Number of chars read
2
2    !*******
2    !******* NULL INPUT
2    !*******
2
2       !+
2       !  Null input.
2       !  Check for DEFAULT parameter - if present prepare to put it through
2       !  Conversion routines by placing DEFAULT in PUT_HERE.
2       !-
2
3       IF ( .CHARS_READ EQL 0 ) AND (( .FLAGS AND V_COB_RPG ) NEQ 0 )
2       THEN
2           !+
2           !  In case of null input for RPG, simply return (no DEFAULT),
2           !  after setting advancing flag.
```

```
2              !-
3              BEGIN
3              IF (.FLAGS AND V_ADV) NEQ 0
3              THEN
3                  COB$$AB_PREV[0] = ACC_DNA
3              ELSE
3                  COB$$AB_PREV[0] = ACC_ADV ;
3              RETURN 1 ;
2              END ;
2
2          !+
2          !  There can be no PROTECTION check on input when dealing with files as
2          !  RMS will only read ACC_SIZE characters or less.  If .ACC_SIZE were 4
2          !  but the record contained "abcdef", only "abcd" will be pulled from the
2          !  record.  RMS ignores the remaining characters "ef" and goes on to the
2          !  next record.  However it is possible to perform a PROTECTION check
2          !  when the DEFAULT value is used.
2          !-
2
3          IF ( .CHARS_READ EQL 0 )
2          THEN
3              BEGIN
4              IF (.DEFAULT NEQ 0) AND (.YES_DEFAULT EQL 0)
3              THEN
4                  BEGIN                                   ! Begin YES Default
4
4                  CHARS_READ = .DEFAULT [DSC$W_LENGTH];
4                  YES_DEFAULT = 1 ;
4
4                  !+
4                  !  Protection check for DEFAULT excluding the Floating
4                  !  Point data types ( these will be handled in
4                  !  COB$$VERIFY_FL_RANGE.
4                  !-
4
5                  IF (.YES_PROTECT AND
6                            ( .STRING_DEST [DSC$B_DTYPE] NEQ DSC$K_DTYPE_F AND
5                              .STRING_DEST [DSC$B_DTYPE] NEQ DSC$K_DTYPE_D ))
4                  THEN                                     ! Check protection
5                      IF (.DEFAULT [DSC$W_LENGTH] GTR .ACC_SIZE)
4                      THEN
4                          !+
4                          !  If the length of DEFAULT is greater than the
4                          !  expected input size ACC_SIZE, then there is a
4                          !  Protection error.
4                          !-
4                          LIB$STOP ( COB$_INVDEFVAL ) ;
4
3                  END ;                                   ! End YES Default
2              END ;
2
2      !*******
2      !******* CONVERSION
2      !*******
2
2         !+
2         !  If conversion requested, call routine COB$$ACC_CONVERT
```

```
2       !-
2
3         IF ( .YES_CONV )
2         THEN
2             CONV_OK = COB$$ACC_CONVERT ( .STRING_DEST, .FLAGS,
2                                          .DEFAULT, .PUT_HERE, .CHARS_READ,
2                                                    .YES_DEFAULT, .YES_SIGN )
2         ELSE
3             BEGIN
3                 LOCAL
3                     COPY_NUM ;
3
3             !+
3             !   No conversion requested - copy input data to STRING_DEST.
3             !   Use STR$COPY_R because it BLANK fills.
3             !-
3
3             IF .CHARS_READ LSS .STRING_DEST[DSC$W_LENGTH]
3             THEN
3                 COPY_NUM = .CHARS_READ
3             ELSE
3                 COPY_NUM = .STRING_DEST[DSC$W_LENGTH] ;
3
3             STR$COPY_R ( .STRING_DEST, COPY_NUM,
4                                         (IF .YES_DEFAULT
4                                         THEN .DEFAULT [DSC$A_POINTER]
3                                         ELSE .PUT_HERE [DSC$A_POINTER] )) ;
3
3             CONV_OK = 1 ;                           ! set CONV_OK to success
2             END;
2
2         !+
2         !  Conversion completed - was it successful ?
2         !-
2
2         IF .CONV_OK  EQL 0
2         THEN
2             !+
2             !  CONVERSION error.  Read UNIT parameter to determine what
2             !  to do.  There is no Reprompting done with Files as input.
2             !
2             !          Byte 2 of                        Conversion
2             !            UNIT                             error
2             !
2             !            0                                COB$_ERRDURACC
2             !            1 ( at end )                     COB$_ERRDURACC
2             !            2 ( on exception )               Return
2             !-
2
3             BEGIN                                   ! Begin conversion error
3             IF ( .FLAGS AND V_COB_RPG ) NEQ 0
3             THEN
3                 !+
3                 !  VAX RPG - return on a Conversion Error, ring bell
3                 !  and clean up first.
3                 !-
4                 BEGIN
```

```
4               COB$$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ;
4               COB$$RPG_CLEAN_UP ( .FLAGS );
4               RETURN 0 ;
3               END ;
3
3           IF .UNIT [1] EQL 2
3           THEN
3               RETURN 0
3           ELSE
3
3               !+
3               !  When dealing with a file, it was decided to return a fatal
3               !  error message rather then REPROMPT.  This lets the user know
3               !  where the problem in the file is so that the input file can
3               !  be corrected before running the program again.  Otherwise, the
3               !  the user might run out of data by the end of the program or the
3               !  reprompting process may lead to further conversion errors.
3               !-
3
3               LIB$STOP ( COB$_ERRDURACC ) ;
2           END ;                                       ! End conversion error
2
2       !+
2       !  Fill in optional LENGTH parameter with the number of
2       !  characters read if no error.
2       !-
2
2       IF .LENGTH NEQ 0
2       THEN
2           .LENGTH = .CHARS_READ ;
2
2   !*******
2   !******* CLEAN UP
2   !*******
2
2       !+
2       !  Determine if ADVANCING is requested.
2       !  If bit 10 = 0 advancing.  If bit 10 = 1 no advancing.
2       !  Set COB$$AB_PREV[0] - also depending on bit 10, to flag to next COBOL
2       !  statement that advancing/no advancing is required following this
2       !  ACCEPT statement.  Echo carriage return to screen if advancing is
2       !  called for.
2       !-
2
2       IF (.FLAGS AND V_ADV) NEQ 0
2       THEN
2           COB$$AB_PREV[0] = ACC_DNA                           ! No Advancing
2       ELSE
2           COB$$RMS_PUT_BYTE ( CARR_RET, .FLAGS ) ;            ! Advance via a carriage
2                                                               ! return
2       RETURN 1;
1       END;                                            ! End of COB$$ACC_SCR_FILE
```

```
                                                       07FC 00000 COB$$ACC_SCR_FILE:
                                                                          .WORD    Save R2,R3,R4,R5,R6,R7,R8,R9,R10             ; 3314
                                       5A     0000V  CF  9E 00002         MOVAB    COB$$RMS_PUT_BYTE, R10                       ;
                                       59 00000000G  8F  D0 00007         MOVL     #COB$_ERRDURACC, R9                          ;
                                       58 00000000G  00  9E 0000E         MOVAB    COB$$AB_PREV, R8                             ;
                                       57 00000000G  00  9E 00015         MOVAB    LIB$STOP, R7                                 ;
                                       5E            04  C2 0001C         SUBL2    #4, SP                                       ;
                                                     55  7C 0001F         CLRQ     CHARS_READ                                   ; 3408
                                                     54  D4 00021         CLRL     YES_DEFAULT                                  ;
                                       50            68  9A 00023         MOVZBL   COB$$AB_PREV, R0                             ; 3425
                                                     0A  13 00026         BEQL     1$                                           ;
                                       02            50  91 00028         CMPB     R0, #2                                       ; 3426
                                                     05  13 0002B         BEQL     1$                                           ;
                                       04            50  91 0002D         CMPB     R0, #4                                       ; 3427
                                                     08  12 00030         BNEQ     2$                                           ;
                                                0C   AC  DD 00032 1$:     PUSHL    FLAGS                                        ; 3429
                                                     01  DD 00035         PUSHL    #1                                           ;
                                       6A            02  FB 00037         CALLS    #2, COB$$RMS_PUT_BYTE                        ;
                                       50       04   AC  9A 0003A 2$:     MOVZBL   UNIT, R0                                     ; 3435
                                       52 00000000G0040  D0 0003E         MOVL     COB$$AL_WRITE_RAB[R0], RAB                   ;
                                  20   A2       18   AC  B0 00046         MOVW     ACC_SIZE, 32(RAB)                            ; 3436
                                       53       1C   AC  D0 0004B         MOVL     PUT_HERE, R3                                 ; 3437
                                  24   A2       04   A3  D0 0004F         MOVL     4(R3), 36(RAB)                               ;
                                  07   A2            10  8A 00054         BICB2    #16, 7(RAB)                                  ; 3442
                                                     52  DD 00058 3$:     PUSHL    RAB                                          ; 3444
                            00000000G  00            01  FB 0005A         CALLS    #1, SYS$GET                                  ;
                            000182DA   8F            50  D1 00061         CMPL     R0, #99034                                   ;
                                                     0B  12 00068         BNEQ     4$                                           ;
                                                     52  DD 0006A         PUSHL    RAB                                          ;
                            00000000G  00            01  FB 0006C         CALLS    #1, SYS$WAIT                                 ;
                                                     E3  11 00073         BRB      3$                                           ;
                                       3C       08   A2  E8 00075 4$:     BLBS     8(RAB), 10$                                  ; 3446
                                       04            6C  91 00079         CMPB     (AP), #4                                     ;
                                                     05  1F 0007C         BLSSU    5$                                           ;
                                                10   AC  D5 0007E         TSTL     16(AP)                                       ;
                                                     32  12 00081         BNEQ     10$                                          ;
                                       7E       08   A2  7D 00083 5$:     MOVQ     8(RAB), -(SP)                                ; 3460
                                                44   A2  9F 00087         PUSHAB   68(RAB)                                      ;
                                                     01  DD 0008A         PUSHL    #1                                           ; 3448
                            0001827A   8F       08   A2  D1 0008C         CMPL     8(RAB), #98938                               ;
                                                     1A  12 00094         BNEQ     8$                                           ;
                                       01       05   AC  91 00096         CMPB     UNIT+1, #1                                   ; 3453
                                                     04  13 0009A         BEQL     6$                                           ;
                                       02       05   AC  91 0009C         CMPB     UNIT+1, #2                                   ;
                                                     03  12 000A0 6$:     BNEQ     7$                                           ;
                                                   00F9  31 000A2         BRW      26$                                          ;
                                       50 00000000G  8F  D0 000A5 7$:     MOVL     #COB$_EOFON_ACC, R0                          ;
                                                     50  DD 000AC         PUSHL    R0                                           ;
                                                     02  11 000AE         BRB      9$                                           ;
                                                     59  DD 000B0 8$:     PUSHL    R9                                           ; 3448
                                       67            05  FB 000B2 9$:     CALLS    #5, LIB$STOP                                 ;
                                       55       22   A2  3C 000B5 10$:    MOVZWL   34(RAB), CHARS_READ                          ; 3467
                                                     50  D4 000B9         CLRL     R0                                           ; 3479
                                                     55  D5 000BB         TSTL     CHARS_READ                                   ;
                                                     15  12 000BD         BNEQ     12$                                          ;
                                                     50  D6 000BF         INCL     R0                                           ;
                         0E       0C   AC            0B  E1 000C1         BBC      #11, FLAGS, 12$                              ;
```

```
                   03         0C   AC               0A E1 000C6       BBC     #10, FLAGS, 11$                     ; 3486
                                                  00BF 31 000CB       BRW     23$                                 ;
                                   68               04 90 000CE 11$:  MOVB    #4, COB$$AB_PREV                    ; 3490
                                                  00C6 31 000D1       BRW     25$                                 ; 3491
                                   36               50 E9 000D4 12$:  BLBC    R0, 13$                             ; 3503
                                         10         AC D5 000D7       TSTL    DEFAULT                             ; 3506
                                                    31 13 000DA       BEQL    13$                                 ;
                                                    54 D5 000DC       TSTL    YES_DEFAULT                         ;
                                                    2D 12 000DE       BNEQ    13$                                 ;
                                   55    10         BC 3C 000E0       MOVZWL  @DEFAULT, CHARS_READ                ; 3510
                                   54               01 D0 000E4       MOVL    #1, YES_DEFAULT                     ; 3511
                                   22    24         AC E9 000E7       BLBC    YES_PROTECT, 13$                    ; 3519
                                   50    08         AC D0 000EB       MOVL    STRING_DEST, R0                     ; 3520
                                   0A    02         A0 91 000EF       CMPB    2(R0), #10                          ;
                                                    18 13 000F3       BEQL    13$                                 ;
                                   0B    02         A0 91 000F5       CMPB    2(R0), #11                          ; 3521
                                                    12 13 000F9       BEQL    13$                                 ;
  18   AC     10   BC              10               00 ED 000FB       CMPZV   #0, #16, @DEFAULT, ACC_SIZE         ; 3523
                                                    09 15 00102       BLEQ    13$                                 ;
                                  00000000G         8F DD 00104       PUSHL   #COB$_INVDEFVAL                     ; 3530
                                   67               01 FB 0010A       CALLS   #1, LIB$STOP                        ;
                                   1A    20         AC E9 0010D 13$:  BLBC    YES_CONV, 14$                       ; 3543
                                         28         AC DD 00111       PUSHL   YES_SIGN                            ; 3547
                                                    54 DD 00114       PUSHL   YES_DEFAULT                         ;
                                                    28 3B 00116       PUSHR   #^M<R3,R5>                          ; 3546
                                   7E    0C         AC 7D 00118       MOVQ    FLAGS, -(SP)                        ; 3545
                                         08         AC DD 0011C       PUSHL   STRING_DEST                         ;
                       00000000G   00               07 FB 0011F       CALLS   #7, COB$$ACC_CONVERT                ;
                                   56               50 D0 00126       MOVL    R0, CONV_OK                         ;
                                                    30 11 00129       BRB     19$                                 ;
       55     08   BC              10               00 ED 0012B 14$:  CMPZV   #0, #16, @STRING_DEST, CHARS_READ   ; 3558
                                                    05 15 00131       BLEQ    15$                                 ;
                                   6E               55 D0 00133       MOVL    CHARS_READ, COPY_NUM                ; 3560
                                                    04 11 00136       BRB     16$                                 ;
                                   6E    08         BC 3C 00138 15$:  MOVZWL  @STRING_DEST, COPY_NUM              ; 3562
                                   09               54 E9 0013C 16$:  BLBC    YES_DEFAULT, 17$                    ; 3565
                                   50    10         AC D0 0013F       MOVL    DEFAULT, R0                         ; 3566
                                         04         A0 DD 00143       PUSHL   4(R0)                               ;
                                                    03 11 00146       BRB     18$                                 ;
                                         04         A3 DD 00148 17$:  PUSHL   4(R3)                               ; 3567
                                         04         AE 9F 0014B 18$:  PUSHAB  COPY_NUM                            ; 3564
                                         08         AC DD 0014E       PUSHL   STRING_DEST                         ;
                       00000000G   00               03 FB 00151       CALLS   #3, STR$COPY_R                      ;
                                   56               01 D0 00158       MOVL    #1, CONV_OK                         ; 3569
                                                    22 12 0015B 19$:  BNEQ    21$                                 ; 3576
                   12         0C   AC               0B E1 0015D       BBC     #11, FLAGS, 20$                     ; 3591
                                         0C         AC DD 00162       PUSHL   FLAGS                               ; 3598
                                                    02 DD 00165       PUSHL   #2                                  ;
                                   6A               02 FB 00167       CALLS   #2, COB$$RMS_PUT_BYTE               ;
                                         0C         AC DD 0016A       PUSHL   FLAGS                               ; 3599
                           0000V   CF               01 FB 0016D       CALLS   #1, COB$$RPG_CLEAN_UP               ;
                                                    2A 11 00172       BRB     26$                                 ; 3600
                                   02    05         AC 91 00174 20$:  CMPB    UNIT+1, #2                          ; 3603
                                                    24 13 00178       BEQL    26$                                 ;
                                                    59 DD 0017A       PUSHL   R9                                  ; 3617
                                   67               01 FB 0017C       CALLS   #1, LIB$STOP                        ;
                                         14         AC D5 0017F 21$:  TSTL    LENGTH                              ; 3625
```

```
                                             04  13 00182          BEQL    22$                                              ;
                          14    BC           55  D0 00184          MOVL    CHARS_READ, @LENGTH                              ; 3627
                 05       0C    AC           0A  E1 00188 22$:     BBC     #10, FLAGS, 24$                                  ; 3642
                                68           05  90 0018D 23$:     MOVB    #5, COB$$AB_PREV                                 ; 3644
                                             08  11 00190          BRB     25$                                              ;
                                       0C    AC  DD 00192 24$:     PUSHL   FLAGS                                            ; 3646
                                             7E  D4 00195          CLRL    -(SP)                                            ;
                                6A           02  FB 00197          CALLS   #2, COB$$RMS_PUT_BYTE                            ;
                                50           01  D0 0019A 25$:     MOVL    #1, R0                                           ; 3648
                                                 04 0019D          RET                                                      ;
                                             50  D4 0019E 26$:     CLRL    R0                                               ; 3649
                                                 04 001A0          RET                                                      ;

; Routine Size:  417 bytes,    Routine Base:  _COB$CODE + 09AA
```

```
1 %SBTTL 'COB$$OPEN_IN - Open for INPUT'
1 GLOBAL ROUTINE COB$$OPEN_IN (UNIT, RPG): NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Open a file for reading, given its unit number.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       UNIT.rl.v           integer unit number designating the device
1 !                           from which the string is to be read.
1 !
1 !       RPG.rl.v            = 1 if RPG is calling this routine
1 !                           = 0 if COBOL is calling this routine
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
1
2     BEGIN
2     LITERAL
2         MAX_BUF =           MAX(LNM$C_NAMLENGTH, NAM$C_MAXRSS);
2     LOCAL
2         FAB:                $FAB_DECL,
2         NAM:                $NAM_DECL,
2         RAB:                REF $RAB_DECL,
2         FILE_NAME:          BLOCK[8, BYTE],         ! Descriptor for the file name
2         TRANSLATE:          BLOCK[8, BYTE],
2         P:                  REF VECTOR[,BYTE],
2         RSLBUF:             VECTOR[MAX_BUF,BYTE],
2         STATUS;
2
2
2     ! Determine whether the COB$xxx name is defined.
2     ! If so, use it.  If not, use the corresponding SYS$xxx name.
2     !
2     TRANSLATE[DSC$B_DTYPE]   = DSC$K_DTYPE_T;
2     TRANSLATE[DSC$B_CLASS]   = DSC$K_CLASS_S;
2     TRANSLATE[DSC$W_LENGTH]  = MAX_BUF;
2     TRANSLATE[DSC$A_POINTER] = RSLBUF;
2
2     !+
   2        !  If VAX RPG is calling this routine, bypass COB_TABLE.
   2        !-
   2
   2        IF .RPG
   2        THEN
   3            BEGIN                                           ! Use the SYS$xxx logical
   3            P = .SYS_TABLE[.UNIT] + BASE;
   3            FILE_NAME[DSC$W_LENGTH] = .P[0];
   3            FILE_NAME[DSC$A_POINTER] = P[1];
   3            END
   2        ELSE
   3            BEGIN                                           ! Use the COB$xxx logical
   3            P = .COB_TABLE[.UNIT] + BASE;
   3            FILE_NAME[DSC$B_DTYPE]   = DSC$K_DTYPE_T;
   3            FILE_NAME[DSC$B_CLASS]   = DSC$K_CLASS_S;
   3            FILE_NAME[DSC$W_LENGTH]  = .P[0];
   3            FILE_NAME[DSC$A_POINTER] = P[1];
   3            IF $TRNLOG(LOGNAM = FILE_NAME, RSLBUF = TRANSLATE) NEQ SS$_NORMAL
   3            THEN
   4                BEGIN                                       ! Use the SYS$xxx logical
   4                P = .SYS_TABLE[.UNIT] + BASE;
   4                FILE_NAME[DSC$W_LENGTH] = .P[0];
   4                FILE_NAME[DSC$A_POINTER] = P[1];
   3                END;
   2            END ;
   2
P  2        $FAB_INIT(
P  2            FAB = FAB,
P  2            NAM = NAM,
P  2            FAC = <GET,PUT>,
P  2            FNA = .FILE_NAME[DSC$A_POINTER],
P  2            FNS = .FILE_NAME[DSC$W_LENGTH],
   2            FOP = SQO);
   2
P  2        $NAM_INIT(
P  2            NAM = NAM,
P  2            ESA = RSLBUF,
P  2            ESS = NAM$C_MAXRSS,
P  2            RSA = RSLBUF,
   2            RSS = NAM$C_MAXRSS);
   2
   2        STATUS = $OPEN(FAB = FAB);
   2        IF (TRANSLATE[DSC$W_LENGTH] = .NAM[NAM$B_RSL]) EQL 0 THEN
   2        IF (TRANSLATE[DSC$W_LENGTH] = .NAM[NAM$B_ESL]) EQL 0
   2        THEN
   3            BEGIN
   3            TRANSLATE[DSC$W_LENGTH] = .FAB[FAB$B_FNS];
   3            TRANSLATE[DSC$A_POINTER]= .FAB[FAB$L_FNA];
   2            END;
   2
   2
   2        IF NOT .STATUS
   2        THEN
   2            LIB$STOP(COB$_ERRDURACC, 1, TRANSLATE, .FAB[FAB$L_STS], .FAB[FAB$L_STV]);
   2
   2
   3        IF NOT (STATUS = LIB$GET_VM(%REF(RAB$C_BLN + 8 + .NAM[NAM$B_RSL]), RAB))
```

```
; 2253          3764  2     THEN
; 2254          3765  2         LIB$STOP(COB$_FAIGET_VM, 0, .STATUS);
; 2255          3766  2
; 2256          3767  2
; 2257          3768  2     ! Save a descriptor for the resultant file name string,
; 2258          3769  2     ! and the string itself, after the RAB
; 2259          3770  2     !
; 2260          3771  3     BEGIN
; 2261          3772  3     LOCAL
; 2262          3773  3         Q: REF BLOCK[,BYTE];
; 2263          3774  3     Q = .RAB + RAB$C_BLN;
; 2264          3775  3     Q[DSC$B_DTYPE]   = DSC$K_DTYPE_T;
; 2265          3776  3     Q[DSC$B_CLASS]   = DSC$K_CLASS_S;
; 2266          3777  3     Q[DSC$W_LENGTH]  = .TRANSLATE[DSC$W_LENGTH];
; 2267          3778  3     Q[DSC$A_POINTER] = .RAB+RAB$C_BLN+8;
; 2268          3779  3     CH$MOVE( .Q[DSC$W_LENGTH], .TRANSLATE[DSC$A_POINTER], .RAB+RAB$C_BLN+8 );
; 2269          3780  2     END;
; 2270          3781  2
; 2271          3782  2     !+
; 2272          3783  2     ! Initiate terminal XABTRM and include it in the RAB.
; 2273          3784  2     !-
; 2274          3785  2
; 2275        P 3786  2     $XABTRM_INIT ( XAB         = XABTRM,
; 2276        P 3787  2                    ITMLST      = XAB_ITMLST,
; 2277          3788  2                    ITMLST_LEN = %ALLOCATION (XAB_ITMLST)-4 ) ;   ! $ITMLST_DECL
; 2278          3789  2                                                                  ! adds extra 4
; 2279        P 3790  2     $RAB_INIT(
; 2280        P 3791  2         RAB = .RAB,
; 2281        P 3792  2         FAB = FAB,
; 2282          3793  2         XAB = XABTRM);
; 2283          3794  2
; 2284          3795  3     IF NOT $CONNECT(RAB = .RAB)
; 2285          3796  2     THEN
; 2286          3797  2         LIB$STOP(COB$_ERRDURACC, 1, .RAB+RAB$C_BLN, .RAB[RAB$L_STS], .RAB[RAB$L_STV]);
; 2287          3798  2
; 2288          3799  2     COB$$AL_WRITE_RAB[.UNIT] = .RAB;
; 2289          3800  2     COB$$AW_WRITE_IFI[.UNIT] = .FAB[FAB$W_IFI];
; 2290          3801  2
; 2291          3802  1     END;                                        ! End of COB$$OPEN_IN



                                                          $RMS_PTR=           XABTRM
                                                                  .EXTRN   SYS$TRNLOG, SYS$OPEN
                                                                  .EXTRN   SYS$CONNECT

                                             0FFC 00000           .ENTRY   COB$$OPEN_IN, Save R2,R3,R4,R5,R6,R7,R8,R9,-; 3651
                                                                           R10,R11                                     ;
                      5B 00000000G   00   9E 00002           MOVAB    LIB$STOP, R11                               ;
                      5A 00000000'   EF   9E 00009           MOVAB    $RMS_PTR, R10                               ;
                      5E       FE38   CE   9E 00010           MOVAB    -456(SP), SP                                ;
              FF40    CD 010E00FF   8F   D0 00015           MOVL     #17694975, TRANSLATE                        ; 3703
              FF44    CD         08   AE   9E 0001E           MOVAB    RSLBUF, TRANSLATE+4                         ; 3704
                      58         04   AC   D0 00024           MOVL     UNIT, R8                                    ; 3713
          57          58              02   78 00028           ASHL     #2, R8, R7                                  ;
                      3A         08   AC   E8 0002C           BLBS     RPG, 1$                                     ; 3710
                      50       F481   CF   9E 00030           MOVAB    BASE, R0                                    ; 3719
```

```
                                         F4E3 CF47  9F 00035       PUSHAB  COB_TABLE[R7]                           ;
                 52             50                    9E  C1 0003A       ADDL3   @(SP)+, R0, P                           ;
                          FF4A  CD       010E   8F  B0 0003E       MOVW    #270, FILE_NAME+2                       ; 3720
                          FF48  CD              62  9B 00045       MOVZBW  (P), FILE_NAME                          ; 3722
                          FF4C  CD         01   A2  9E 0004A       MOVAB   1(R2), FILE_NAME+4                      ; 3723
                                                7E  7C 00050       CLRQ    -(SP)                                   ; 3724
                                                7E  D4 00052       CLRL    -(SP)                                   ;
                                       FF40     CD  9F 00054       PUSHAB  TRANSLATE                               ;
                                                7E  D4 00058       CLRL    -(SP)                                   ;
                                       FF48     CD  9F 0005A       PUSHAB  FILE_NAME                               ;
                     00000000G  00              06  FB 0005E       CALLS   #6, SYS$TRNLOG                          ;
                                01              50  D1 00065       CMPL    R0, #1                                  ;
                                                19  13 00068       BEQL    2$                                      ;
                                50     F447     CF  9E 0006A 1$:   MOVAB   BASE, R0                                ; 3727
                                       F511   CF47  9F 0006F       PUSHAB  SYS_TABLE[R7]                           ;
                 52             50              9E  C1 00074       ADDL3   @(SP)+, R0, P                           ;
                          FF48  CD              62  9B 00078       MOVZBW  (P), FILE_NAME                          ; 3728
                          FF4C  CD         01   A2  9E 0007D       MOVAB   1(R2), FILE_NAME+4                      ; 3729
  0050    8F     00             6E              00  2C 00083 2$:   MOVC5   #0, (SP), #0, #80, $RMS_PTR             ; 3739
                                           B0   AD     0008A                                                   ;
                            B0  AD       5003   8F  B0 0008C       MOVW    #20483, $RMS_PTR                        ;
                            B4  AD         40   8F  9A 00092       MOVZBL  #64, $RMS_PTR+4                         ;
                            C6  AD              03  90 00097       MOVB    #3, $RMS_PTR+22                         ;
                            CF  AD              02  90 0009B       MOVB    #2, $RMS_PTR+31                         ;
                            D8  AD       FF50   CD  9E 0009F       MOVAB   NAM, $RMS_PTR+40                        ;
                            DC  AD       FF4C   CD  D0 000A5       MOVL    FILE_NAME+4, $RMS_PTR+44                ;
                            E4  AD       FF48   CD  90 000AB       MOVB    FILE_NAME, $RMS_PTR+52                  ;
  0060    8F     00             6E              00  2C 000B1       MOVC5   #0, (SP), #0, #96, $RMS_PTR             ; 3746
                                         FF50   CD     000B8                                                   ;
                          FF50  CD       6002   8F  B0 000BB       MOVW    #24578, $RMS_PTR                        ;
                          FF52  CD              01  8E 000C2       MNEGB   #1, $RMS_PTR+2                          ;
                          FF54  CD         08   AE  9E 000C7       MOVAB   RSLBUF, $RMS_PTR+4                      ;
                          FF5A  CD              01  8E 000CD       MNEGB   #1, $RMS_PTR+10                         ;
                          FF5C  CD         08   AE  9E 000D2       MOVAB   RSLBUF, $RMS_PTR+12                     ;
                                           B0   AD  9F 000D8       PUSHAB  FAB                                     ; 3748
                     00000000G  00              01  FB 000DB       CALLS   #1, SYS$OPEN                            ;
                                52              50  D0 000E2       MOVL    R0, STATUS                              ;
                          FF40  CD       FF53   CD  9B 000E5       MOVZBW  NAM+3, TRANSLATE                        ; 3749
                                                15  12 000EC       BNEQ    3$                                      ;
                          FF40  CD       FF5B   CD  9B 000EE       MOVZBW  NAM+11, TRANSLATE                       ; 3750
                                                0C  12 000F5       BNEQ    3$                                      ;
                          FF40  CD         E4   AD  9B 000F7       MOVZBW  FAB+52, TRANSLATE                       ; 3753
                          FF44  CD         DC   AD  D0 000FD       MOVL    FAB+44, TRANSLATE+4                     ; 3754
                                13              52  E8 00103 3$:   BLBS    STATUS, 4$                              ; 3758
                                7E         B8   AD  7D 00106       MOVQ    FAB+8, -(SP)                            ; 3760
                                         FF40   CD  9F 0010A       PUSHAB  TRANSLATE                               ;
                                                01  DD 0010E       PUSHL   #1                                      ;
                                    00000000G   8F  DD 00110       PUSHL   #COB$_ERRDURACC                         ;
                                6B              05  FB 00116       CALLS   #5, LIB$STOP                            ;
                                           04   AE  9F 00119 4$:   PUSHAB  RAB                                     ; 3763
                            04  AE       FF53   CD  9A 0011C       MOVZBL  NAM+3, 4(SP)                            ;
                            04  AE   0000004C   8F  C0 00122       ADDL2   #76, 4(SP)                              ;
                                           04   AE  9F 0012A       PUSHAB  4(SP)                                   ;
                     00000000G  00              02  FB 0012D       CALLS   #2, LIB$GET_VM                          ;
                                52              50  D0 00134       MOVL    R0, STATUS                              ;
                                0D              52  E8 00137       BLBS    STATUS, 5$                              ;
                                                52  DD 0013A       PUSHL   STATUS                                  ; 3765
```

```
                                                   7E  D4 0013C         CLRL   -(SP)                                      ;
                                        00000000G  8F  DD 0013E         PUSHL  #COB$_FAIGET_VM                            ;
                                      6B           03  FB 00144         CALLS  #3, LIB$STOP                               ;
                                      56       04  AE  D0 00147 5$:     MOVL   RAB, R6                                    ; 3774
                                      59       44  A6  9E 0014B         MOVAB  68(R6), R9                                 ;
                                      50           59  D0 0014F         MOVL   R9, Q                                      ;
                                  02  A0     010E  8F  B0 00152         MOVW   #270, 2(Q)                                 ; 3775
                                      60     FF40  CD  B0 00158         MOVW   TRANSLATE, (Q)                             ; 3777
                                  04  A0       4C  A6  9E 0015D         MOVAB  76(R6), 4(Q)                               ; 3778
                    4C  A6      FF44  DD           60  28 00162         MOVC3  (Q), @TRANSLATE+4, 76(R6)                  ; 3779
           24           00            6E           00  2C 00169         MOVC5  #0, (SP), #0, #36, $RMS_PTR                ; 3788
                                                   6A     0016E                                                           ;
                                      6A     241F  8F  B0 0016F         MOVW   #9247, $RMS_PTR                            ;
                                  08  AA       24  AA  9E 00174         MOVAB  XAB_ITMLST, $RMS_PTR+8                     ;
                                  0C  AA           18  B0 00179         MOVW   #24, $RMS_PTR+12                           ;
    0044   8F           00            6E           00  2C 0017D         MOVC5  #0, (SP), #0, #68, (R6)                    ; 3793
                                                   66     00184                                                           ;
                                      66     4401  8F  B0 00185         MOVW   #17409, (R6)                               ;
                                  3C  A6       B0  AD  9E 0018A         MOVAB  FAB, 60(R6)                                ;
                                  40  A6           6A  9E 0018F         MOVAB  XABTRM, 64(R6)                             ;
                                                   56  DD 00193         PUSHL  R6                                         ; 3795
                           00000000G  00           01  FB 00195         CALLS  #1, SYS$CONNECT                            ;
                                      11           50  E8 0019C         BLBS   R0, 6$                                     ;
                                      7E       08  A6  7D 0019F         MOVQ   8(R6), -(SP)                               ; 3797
                                                   59  DD 001A3         PUSHL  R9                                         ;
                                                   01  DD 001A5         PUSHL  #1                                         ;
                                        00000000G  8F  DD 001A7         PUSHL  #COB$_ERRDURACC                            ;
                                      6B           05  FB 001AD         CALLS  #5, LIB$STOP                               ;
                                        00000000G0047  9F 001B0 6$:     PUSHAB COB$$AL_WRITE_RAB[R7]                      ; 3799
                                      9E           56  D0 001B7         MOVL   R6, @(SP)+                                 ;
                           00000000G0048       B2  AD  B0 001BA         MOVW   FAB+2, COB$$AW_WRITE_IFI[R8]               ; 3800
                                                       04 001C3         RET                                               ; 3802

; Routine Size:  452 bytes,    Routine Base:  _COB$CODE + 0B4B
; 2293           3803  1 %SBTTL 'COB$$RMS_GET - Perform an RMS $GET Service'
; 2294           3804  1 ROUTINE COB$$RMS_GET ( RAB      :  REF $RAB_DECL,
; 2295           3805  1                        FUNC_VAL,
; 2296           3806  1                        LENGTH,
; 2297           3807  1                        BUFFER
; 2298           3808  1                     )  :  NOVALUE  =
; 2299           3809  1
; 2300           3810  1 !++
; 2301           3811  1 ! FUNCTIONAL DESCRIPTION:
; 2302           3812  1 !
; 2303           3813  1 !
; 2304           3814  1 ! FORMAL PARAMETERS:
; 2305           3815  1 !
; 2306           3816  1 !
; 2307           3817  1 ! IMPLICIT INPUTS:
; 2308           3818  1 !
; 2309           3819  1 !       NONE
; 2310           3820  1 !
; 2311           3821  1 ! IMPLICIT OUTPUTS:
; 2312           3822  1 !
; 2313           3823  1 !       NONE
; 2314           3824  1 !
; 2315           3825  1 ! ROUTINE VALUE:
; 2316           3826  1 ! COMPLETION CODES:
; 2317           3827  1 !
; 2318           3828  1 !       NONE
; 2319           3829  1 !
; 2320           3830  1 ! SIDE EFFECTS:
; 2321           3831  1 !
; 2322           3832  1 !--
; 2323           3833  1
; 2324           3834  2     BEGIN
; 2325           3835  2
; 2326         P 3836  2     $ITMLST_INIT (ITMLST = XAB_ITMLST,                          ! Item list for $GET
; 2327         P 3837  2                   (ITMCOD = TRM$_MODIFIERS,
; 2328         P 3838  2                    BUFSIZ = 0,
; 2329         P 3839  2                    BUFADR = .FUNC_VAL),
; 2330         P 3840  2                   (ITMCOD = TRM$_TERM,
; 2331         P 3841  2                    BUFSIZ = 20,
; 2332           3842  2                    BUFADR = MASK_VECTOR) ) ;
; 2333           3843  2
; 2334           3844  2     RAB [RAB$W_USZ] = .LENGTH ;
; 2335           3845  2     RAB [RAB$L_UBF] = .BUFFER ;
; 2336           3846  2     RAB [RAB$V_ETO] = 1 ;                                      ! Extended Terminal $GET
; 2337           3847  2     RAB [RAB$L_XAB] = XABTRM ;
; 2338           3848  2     WHILE $GET (RAB = .RAB) EQL RMS$_RSA DO $WAIT (RAB = .RAB) ;
; 2339           3849  2
; 2340           3850  2     IF NOT .RAB [RAB$L_STS]
; 2341           3851  2     THEN
; 2342           3852  2         !+
; 2343           3853  2         !  These are special case status that will be handled later.
; 2344           3854  2         !-
; 2345           3855  3         IF (.RAB [RAB$L_STS] NEQ RMS$_BES AND                  ! Bad Escape Sequence
; 2346           3856  3             .RAB [RAB$L_STS] NEQ RMS$_EOF AND                  ! End Of File
; 2347           3857  3             .RAB [RAB$L_STS] NEQ RMS$_PES AND                  ! Partial Escape Seq
; 2348           3858  3             .RAB [RAB$L_STS] NEQ RMS$_RTB AND                  ! Record Too Big
; 2349           3859  3             .RAB [RAB$L_STS] NEQ RMS$_TNS )                    ! Terminator Not Seen
; 2350       3860  2          THEN
; 2351       3861  2              LIB$STOP (COB$_ERRDURACC, 1, .RAB + RAB$C_BLN, .RAB [RAB$L_STS],
; 2352       3862  2                                                          .RAB [RAB$L_STV] ) ;
; 2353       3863  1      END ;




                                       000C 00000 COB$$RMS_GET:
                                                       .WORD    Save R2,R3                              ; 3804
          53 00000000'  EF  9E 00002                   MOVAB    XAB_ITMLST, R3                          ;
          50            63  9E 00009                   MOVAB    XAB_ITMLST, $$ITMBLKPTR                 ; 3842
                        80  D4 0000C                   CLRL     ($$ITMBLKPTR)+                          ;
          80        08  AC  D0 0000E                   MOVL     FUNC_VAL, ($$ITMBLKPTR)+                ;
                        80  D4 00012                   CLRL     ($$ITMBLKPTR)+                          ;
          80  00030014  8F  D0 00014                   MOVL     #196628, ($$ITMBLKPTR)+                 ;
          80        1C  A3  9E 0001B                   MOVAB    MASK_VECTOR, ($$ITMBLKPTR)+             ;
                        80  7C 0001F                   CLRQ     ($$ITMBLKPTR)+                          ;
          52        04  AC  D0 00021                   MOVL     RAB, R2                                 ; 3844
       20 A2        0C  AC  B0 00025                   MOVW     LENGTH, 32(R2)                          ;
       24 A2        10  AC  D0 0002A                   MOVL     BUFFER, 36(R2)                          ; 3845
       07 A2            10  88 0002F                   BISB2    #16, 7(R2)                              ; 3846
       40 A2        DC  A3  9E 00033                   MOVAB    XABTRM, 64(R2)                          ; 3847
                        52  DD 00038 1$:               PUSHL    R2                                      ; 3848
00000000G 00            01  FB 0003A                   CALLS    #1, SYS$GET                             ;
000182DA  8F            50  D1 00041                   CMPL     R0, #99034                              ;
                        0B  12 00048                   BNEQ     2$                                      ;
                        52  DD 0004A                   PUSHL    R2                                      ;
00000000G 00            01  FB 0004C                   CALLS    #1, SYS$WAIT                            ;
                        E3  11 00053                   BRB      1$                                      ;
          50        08  A2  D0 00055 2$:               MOVL     8(R2), R0                               ; 3850
          44            50  E8 00059                   BLBS     R0, 3$                                  ;
000181C0  8F            50  D1 0005C                   CMPL     R0, #98752                              ; 3855
                        3B  13 00063                   BEQL     3$                                      ;
0001827A  8F            50  D1 00065                   CMPL     R0, #98938                              ; 3856
                        32  13 0006C                   BEQL     3$                                      ;
000181C8  8F            50  D1 0006E                   CMPL     R0, #98760                              ; 3857
                        29  13 00075                   BEQL     3$                                      ;
000181A8  8F            50  D1 00077                   CMPL     R0, #98728                              ; 3858
                        20  13 0007E                   BEQL     3$                                      ;
000181B8  8F            50  D1 00080                   CMPL     R0, #98744                              ; 3859
                        17  13 00087                   BEQL     3$                                      ;
                    0C  A2  DD 00089                   PUSHL    12(R2)                                  ; 3862
                        50  DD 0008C                   PUSHL    R0                                      ; 3861
                    44  A2  9F 0008E                   PUSHAB   68(R2)                                  ;
                        01  DD 00091                   PUSHL    #1                                      ;
             00000000G  8F  DD 00093                   PUSHL    #COB$_ERRDURACC                         ;
00000000G 00            05  FB 00099                   CALLS    #5, LIB$STOP                            ;
                            04 000A0 3$:               RET                                              ; 3863

; Routine Size:  161 bytes,    Routine Base:  _COB$CODE + 0D0F
```

```
1 %SBTTL 'COB$$RMS_PUT_BYTE - Perform an RMS $PUT Service'
1 ROUTINE COB$$RMS_PUT_BYTE ( WHICH, FLAGS )  :  NOVALUE  =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine writes a one byte buffer to the terminal. Either a
1 !       Carriage Return, Linefeed or Ring the Terminal Bell, depending
1 !       on the value of WHICH.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       WHICH.rl.v      if 0, write Linefeed to terminal
1 !                       if 1, write Carriage Return to terminal
1 !                       if 2, ring terminal bell
1 !
1 !       FLAGS.rlu.v     Screen enhancement flag
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !--
1
2     BEGIN
2     LOCAL
2         RAB    :  REF $RAB_DECL,
2         CR_BUF :  VECTOR [1,BYTE] ;
2
2     IF .COB$ACC_TERM_TYPE NEQ UNKNOWN
2     THEN
3         BEGIN
3
3         SELECTONE .WHICH OF
3             SET
3                 [0] :   CR_BUF [0] = CR ;                 ! Carriage Return
3
3                 [1] :   CR_BUF [0] = LF ;                 ! Linefeed
3
3                 [2] :   CR_BUF [0] = BELL ;               ! Bell
3             TES ;
3
3         COB$$AB_USPCODE [0] = 0 ;
3         COB$$AB_USPCODE [1] = 0 ;
3
```

```
; 2412          3921  3         IF .COB$$AL_WRITE_RAB [1] EQL 0
; 2413          3922  3         THEN
; 2414          3923  4             BEGIN
; 2415          3924  4             !+
; 2416          3925  4             !  Open SYS$OUTPUT.  Second parameter tells COB$$OPEN_OUT whether
; 2417          3926  4             !  VAX COBOL (0) or VAX RPG (1) is the caller.
; 2418          3927  4             !  VMS V4 defines SYS$INPUT as read only, therefore any $PUTs must
; 2419          3928  4             !  be made through SYS$OUTPUT.  When a terminal is the input device
; 2420          3929  4             !  for an ACCEPT it is also the OUTPUT device, and must be OPENed
; 2421          3930  4             !  for both.
; 2422          3931  4             !-
; 2423          3932  4             COB$$OPEN_OUT ( 1,
; 2424          3933  4                             IF ( .FLAGS AND V_COB_RPG ) NEQ 0
; 2425          3934  4                             THEN 1
; 2426          3935  4                             ELSE 0 ) ;
; 2427          3936  3             END ;
; 2428          3937  3         RAB = .COB$$AL_WRITE_RAB [1] ;
; 2429          3938  3         RAB [RAB$L_RBF] = CR_BUF [0] ;
; 2430          3939  3         RAB [RAB$W_RSZ] = 1 ;
; 2431          3940  3         WHILE $PUT (RAB = .RAB) EQL RMS$_RSA DO $WAIT (RAB = .RAB) ;
; 2432          3941  3
; 2433          3942  3         IF NOT .RAB [RAB$L_STS]
; 2434          3943  3         THEN
; 2435          3944  3             LIB$STOP ( COB$_ERRDURACC, 1, .RAB + RAB$C_BLN,
; 2436          3945  3                         .RAB [RAB$L_STS], .RAB [RAB$L_STV] ) ;
; 2437          3946  2         END ;
; 2438          3947  1     END ;                                   ! End of COB$$RMS_PUT_BYTE


                                                             .EXTRN  SYS$PUT

                                     000C 00000 COB$$RMS_PUT_BYTE:
                                                             .WORD   Save R2,R3                          ; 3865
                53 00000000G    00   9E 00002                MOVAB   COB$$AL_WRITE_RAB+4, R3             ;
                5E              04   C2 00009                SUBL2   #4, SP                              ;
                   00000000G    00   D5 0000C                TSTL    COB$ACC_TERM_TYPE                   ; 3905
                                7D   13 00012                BEQL    9$                                  ;
                50        04    AC   D0 00014                MOVL    WHICH, R0                           ; 3909
                                05   12 00018                BNEQ    1$                                  ; 3911
                6E              0D   90 0001A                MOVB    #13, CR_BUF                         ;
                                12   11 0001D                BRB     3$                                  ;
                01              50   D1 0001F 1$:            CMPL    R0, #1                              ; 3913
                                05   12 00022                BNEQ    2$                                  ;
                6E              0A   90 00024                MOVB    #10, CR_BUF                         ;
                                08   11 00027                BRB     3$                                  ;
                02              50   D1 00029 2$:            CMPL    R0, #2                              ; 3915
                                03   12 0002C                BNEQ    3$                                  ;
                6E              07   90 0002E                MOVB    #7, CR_BUF                          ;
                   00000000G    00   B4 00031 3$:            CLRW    COB$$AB_USPCODE                     ; 3918
                                63   D5 00037                TSTL    COB$$AL_WRITE_RAB+4                 ; 3921
                                14   12 00039                BNEQ    6$                                  ;
    04        08    AC          0B   E1 0003B                BBC     #11, FLAGS, 4$                      ; 3933
                                01   DD 00040                PUSHL   #1                                  ;
                                02   11 00042                BRB     5$                                  ;
                                7E   D4 00044 4$:            CLRL    -(SP)                               ;
                                01   DD 00046 5$:            PUSHL   #1                                  ; 3932
```

```
                 0000000G  00            02  FB 00048       CALLS   #2, COB$$OPEN_OUT                          ;
                           52            63  D0 0004F 6$:   MOVL    COB$$AL_WRITE_RAB+4, RAB                   ; 3937
                       28  A2            6E  9E 00052       MOVAB   CR_BUF, 40(RAB)                            ; 3938
                       22  A2            01  B0 00056       MOVW    #1, 34(RAB)                                ; 3939
                                         52  DD 0005A 7$:   PUSHL   RAB                                        ; 3940
                 0000000G  00            01  FB 0005C       CALLS   #1, SYS$PUT                                ;
                 000182DA  8F            50  D1 00063       CMPL    R0, #99034                                 ;
                                         0B  12 0006A       BNEQ    8$                                         ;
                                         52  DD 0006C       PUSHL   RAB                                        ;
                 0000000G  00            01  FB 0006E       CALLS   #1, SYS$WAIT                               ;
                                         E3  11 00075       BRB     7$                                         ;
                           16        08  A2  E8 00077 8$:   BLBS    8(RAB), 9$                                 ; 3942
                           7E        08  A2  7D 0007B       MOVQ    8(RAB), -(SP)                              ; 3945
                                     44  A2  9F 0007F       PUSHAB  68(RAB)                                    ; 3944
                                         01  DD 00082       PUSHL   #1                                         ;
                               0000000G  8F  DD 00084       PUSHL   #COB$_ERRDURACC                            ;
                 0000000G  00            05  FB 0008A       CALLS   #5, LIB$STOP                               ;
                                             04 00091 9$:   RET                                                ; 3947

; Routine Size:  146 bytes,    Routine Base:  _COB$CODE + 0DB0
```

```
1 %SBTTL 'COB$$RMS_PUT_BUFFER - Perform RMS $PUT Service'
1 ROUTINE COB$$RMS_PUT_BUFFER ( BUFFER,
1                               LENGTH,
1                               FLAGS )  :  NOVALUE  =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine writes buffer of more than one byte to the terminal.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       BUFFER.rt.r     Holds sequence to write to screen
1 !
1 !       LENGHT.rlu.v    Length of BUFFER
1 !
1 !       FLAGS.rlu.v     Screen enhancement flag
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 ! COMPLETION CODES:
1 !
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !--
1
2     BEGIN
2     LOCAL
2         RAB    :  REF $RAB_DECL ;
2
2     IF .COB$ACC_TERM_TYPE NEQ UNKNOWN
2     THEN
3         BEGIN
3
3         COB$$AB_USPCODE [0] = 0 ;
3         COB$$AB_USPCODE [1] = 0 ;
3
3         IF .COB$$AL_WRITE_RAB [1] EQL 0
3         THEN
4             BEGIN
4             !+
4             !  Open SYS$OUTPUT.  Second parameter tells COB$$OPEN_OUT whether
4             !  VAX COBOL (0) or VAX RPG (1) is the caller.
4             !-
4             COB$$OPEN_OUT ( 1,
4                             IF ( .FLAGS AND V_COB_RPG ) NEQ 0
4                             THEN 1
; 2497          4005   4                              ELSE 0 ) ;
; 2498          4006   3              END ;
; 2499          4007   3          RAB = .COB$$AL_WRITE_RAB [1] ;
; 2500          4008   3          RAB [RAB$L_RBF] = .BUFFER ;
; 2501          4009   3          RAB [RAB$W_RSZ] = .LENGTH ;
; 2502          4010   3          WHILE $PUT (RAB = .RAB) EQL RMS$_RSA DO $WAIT (RAB = .RAB) ;
; 2503          4011   3
; 2504          4012   3          IF NOT .RAB [RAB$L_STS]
; 2505          4013   3          THEN
; 2506          4014   3              LIB$STOP ( COB$_ERRDURACC, 1, .RAB + RAB$C_BLN,
; 2507          4015   3                          .RAB [RAB$L_STS], .RAB [RAB$L_STV] ) ;
; 2508          4016   2          END ;
; 2509          4017   1      END ;                                           ! End of COB$$RMS_PUT_BUFFER



                                                       000C 00000 COB$$RMS_PUT_BUFFER:
                                                                            .WORD    Save R2,R3                                    ; 3949
                                        53 00000000G  00 9E 00002           MOVAB    COB$$AL_WRITE_RAB+4, R3                       :
                                           00000000G  00 D5 00009           TSTL     COB$ACC_TERM_TYPE                             ; 3988
                                                      62 13 0000F           BEQL     6$                                            :
                                           00000000G  00 B4 00011           CLRW     COB$$AB_USPCODE                               ; 3992
                                                      63 D5 00017           TSTL     COB$$AL_WRITE_RAB+4                           ; 3995
                                                      14 12 00019           BNEQ     3$                                            :
                          04       0C   AC            0B E1 0001B           BBC      #11, FLAGS, 1$                                ; 4003
                                                      01 DD 00020           PUSHL    #1                                            :
                                                      02 11 00022           BRB      2$                                            :
                                                      7E D4 00024 1$:       CLRL     -(SP)                                         :
                                                      01 DD 00026 2$:       PUSHL    #1                                            ; 4002
                             00000000G  00            02 FB 00028           CALLS    #2, COB$$OPEN_OUT                             :
                                        52            63 D0 0002F 3$:       MOVL     COB$$AL_WRITE_RAB+4, RAB                      ; 4007
                                        28  A2     04 AC D0 00032           MOVL     BUFFER, 40(RAB)                               ; 4008
                                        22  A2     08 AC B0 00037           MOVW     LENGTH, 34(RAB)                               ; 4009
                                                      52 DD 0003C 4$:       PUSHL    RAB                                           ; 4010
                             00000000G  00            01 FB 0003E           CALLS    #1, SYS$PUT                                   :
                             000182DA   8F            50 D1 00045           CMPL     R0, #99034                                    :
                                                      0B 12 0004C           BNEQ     5$                                            :
                                                      52 DD 0004E           PUSHL    RAB                                           :
                             00000000G  00            01 FB 00050           CALLS    #1, SYS$WAIT                                  :
                                                      E3 11 00057           BRB      4$                                            :
                                        16         08 A2 E8 00059 5$:       BLBS     8(RAB), 6$                                    ; 4012
                                        7E         08 A2 7D 0005D           MOVQ     8(RAB), -(SP)                                 ; 4015
                                                   44 A2 9F 00061           PUSHAB   68(RAB)                                       ; 4014
                                                      01 DD 00064           PUSHL    #1                                            :
                                           00000000G  8F DD 00066           PUSHL    #COB$_ERRDURACC                               :
                             00000000G  00            05 FB 0006C           CALLS    #5, LIB$STOP                                  :
                                                         04 00073 6$:       RET                                                    ; 4017

; Routine Size:  116 bytes,    Routine Base:  _COB$CODE + 0E42
```

```
1 %SBTTL 'COB$$CONTROL_Z - Handle ^Z'
1 ROUTINE COB$$CONTROL_Z ( UNIT            :  VECTOR [2, BYTE],
1                          KEY             :  REF $STR$DESCRIPTOR
1                         )  :  NOVALUE =
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Read UNIT parameter to determine what to do when a Control Z was typed.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       UNIT.rbu.va     Array of two unsigned byte integers.
1 !                       The first byte is the unit number designating the
1 !                       device from which the string is to be read.
1 !                       The second byte indicates whether the routine should
1 !                       abort or return to the calling program.
1 !
1 !       KEY.wt.ds       Destination of the receiving field of the control key.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !--
2       BEGIN
2           LOCAL
2               TERM_PTR,                               ! Points to terminator
2               CZ_PTR ;                                ! Needed for CH$MOVE
2
2        !+
2        !  CONTROL Z - read UNIT parameter to determine what to do.
2        !
2        !        Byte 2 of             Ctrl z
2        !          UNIT
2        !
2        !           0                  abort
2        !           1 ( at end )       return
2        !           2 ( on exception ) return
2        !-
2
2               IF .UNIT [1]  EQL  0
2               THEN
2                   LIB$STOP ( COB$_EOFON_ACC )         ! Abort
2               ELSE
3                   BEGIN
3                   IF .KEY NEQ 0
3                   THEN
```

```
; 2568          4075 4                      BEGIN
; 2569          4076 4                      !+
; 2570          4077 4                      !  Pass CONTROL Z back to user program via KEY,
; 2571          4078 4                      !  if requested
; 2572          4079 4                      !-
; 2573          4080 4                      CZ_PTR = CZ ;                       ! CZ is literal %X'1A'
; 2574          4081 4                      TERM_PTR = CZ_PTR ;
; 2575          4082 4                      CH$MOVE ( 1, .TERM_PTR, .KEY [DSC$A_POINTER] ) ;
; 2576          4083 3                      END ;
; 2577          4084 2                  END ;
; 2578          4085 1          END ;                                   ! End routine COB$$CONTROL_Z



                                          0000 00000 COB$$CONTROL_Z:
                                                              .WORD   Save nothing                          ; 4019
                     5E             04 C2 00002               SUBL2   #4, SP                                ;
                               05   AC 95 00005               TSTB    UNIT+1                                ; 4068
                                    0E 12 00008               BNEQ    1$                                    ;
                       00000000G    8F DD 0000A               PUSHL   #COB$_EOFON_ACC                       ; 4070
              00000000G  00         01 FB 00010               CALLS   #1, LIB$STOP                          ;
                                       04 00017               RET                                           ;
                     50        08   AC D0 00018 1$:           MOVL    KEY, R0                               ; 4073
                                    0A 13 0001C               BEQL    2$                                    ;
                     6E             1A D0 0001E               MOVL    #26, CZ_PTR                           ; 4080
                     51             6E 9E 00021               MOVAB   CZ_PTR, TERM_PTR                      ; 4081
                 04  B0             61 90 00024               MOVB    (TERM_PTR), @4(R0)                    ; 4082
                                       04 00028 2$:           RET                                           ; 4085

; Routine Size:  41 bytes,    Routine Base:  _COB$CODE + 0EB6
```

```
1 %SBTTL 'COB$$PARTIAL_SEQ - Partial Escape Sequence'
1 ROUTINE COB$$PARTIAL_SEQ ( PARAMETERS : REF VECTOR,
1                            UNIT       : VECTOR [2,BYTE] )  :  NOVALUE =
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       The entire Escape sequence did not fit in the initial $GET's buffer.
1 !       Perform 1 character Reads until the full sequence is in PUT_HERE or
1 !       NEXT_CHAR.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       PARAMETERS.mlu.ra  Contains data for this routine.
1 !
1 !       UNIT.rbu.va        Array of two unsigned byte integers.
1 !                          The first byte is the unit number designating the
1 !                          device from which the string is to be read.
1 !                          The second byte indicates whether the routine should
1 !                          abort or return to the calling program.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
2     BEGIN
2
2     LOCAL
2         RAB           :  REF $RAB_DECL,
2         FUNC_VAL_2,                            ! QIO Function Modifiers for
2                                                ! item list of RMS $GET.
2         TERM_CHAR     :  BYTE,                 ! $GET input buffer.
2         PH            :  REF VECTOR [1100,BYTE],  ! Address of PUT_HERE
2         PH_PTR,                                ! Pointer to PUT_HERE.
2         NC_PTR,                                ! Pointer to NEXT_CHAR.
2         END_OF_TERM   :  INITIAL (0) ;         ! =1 whole Seq in buffer.
2
2         !+
2         !  Bind PARAMETERS to other names.
2         !-
2
2         $BIND_PARAMETERS ;
2         PH = .PUT_HERE [DSC$A_POINTER] ;
2
2         !+
2         !  PH_PTR and NC_PTR point to next free space in buffer
```

```
2           !  PUT_HERE or NEXT_CHAR.
2           !-
2
2           PH_PTR = .CHARS_READ + .TERM_SIZE ;
2           NC_PTR = 1 ;
2
2           !+
2           !  Read one character at a time until the entire escape sequence has
2           !  been read.
2           !-
2
2           WHILE .END_OF_TERM EQL 0 DO
3               BEGIN                                           ! Begin loop
3
3               FUNC_VAL_2 = TRM$M_TM_ESCAPE + TRM$M_TM_NOFILTR + TRM$M_TM_TRMNOECHO
3                                                               + TRM$M_TM_NOECHO ;
3
3               RAB = .COB$$AL_WRITE_RAB [ .UNIT[0] ] ;
3               COB$$RMS_GET ( .RAB, .FUNC_VAL_2, 1, TERM_CHAR ) ;
3
3               !+
3               !  Deposit sequence character in appropriate buffer.
3               !-
3
3               IF .TERM_IN_NEXT EQL 0
3               THEN
4                   BEGIN
4                   !+
4                   !  This is a workaround for an RMS bug that did not
4                   !  make it into the final code freeze for V4.0.
4                   !  The next three lines can be pulled when the RMS fix
4                   !  is made.  (see NEXT_CHAR below)
4                   !-
4                   IF .TERM_SIZE EQL 1                         ! Put first character
4                   THEN                                        ! of terminator seq
4                       PH [.PH_PTR - 1] = %X'1B' ;             ! into PUT_HERE
4
4                   PH [.PH_PTR] = .TERM_CHAR ;                 ! Put character just
4                   PH_PTR = .PH_PTR + 1 ;                      ! read in PUT_HERE
4                   TERM_IN_NEXT = 0 ;
4                   END
3               ELSE
4                   BEGIN
4                   !+
4                   !  This is a workaround for an RMS bug that did not
4                   !  make it into the final code freeze for V4.0.
4                   !  The next three lines can be pulled when the RMS fix
4                   !  is made.
4                   !-
4                   IF .TERM_SIZE EQL 1                         ! Put first character
4                   THEN                                        ! of terminator seq
4                       NEXT_CHAR [0] = %X'1B' ;                ! into NEXT_CHAR.
4
4                   NEXT_CHAR [.NC_PTR] = .TERM_CHAR ;          ! Put character just
4                   NC_PTR = .NC_PTR + 1 ;                      ! read in NEXT_CHAR
3                   END ;
3
```

```
; 2694          4200  3                 TERM_SIZE = .TERM_SIZE + 1 ;                        ! Total Terminator size
; 2695          4201  3
; 2696          4202  3                 !+
; 2697          4203  3                 !   Ugly - but it's the only way to check for the end of an
; 2698          4204  3                 !   escape sequence.  All known KEY escape sequences end in
; 2699          4205  3                 !   one of these characters and none of these characters fall
; 2700          4206  3                 !   in the middle of an escape sequence.  This will have to be
; 2701          4207  3                 !   updated if new escape sequences surface.
; 2702          4208  3                 !-
; 2703          4209  3
; 2704          4210  4                 IF ((.TERM_CHAR GEQ %C'A'  AND  .TERM_CHAR LEQ  %C'M')  OR
; 2705          4211  4                     (.TERM_CHAR GEQ %C'P'  AND  .TERM_CHAR LEQ  %C'S')  OR
; 2706          4212  4                     (.TERM_CHAR GEQ %C'l'  AND  .TERM_CHAR LEQ  %C'y')  OR
; 2707          4213  4                     (.TERM_CHAR EQL %X'7E'))
; 2708          4214  3                 THEN
; 2709          4215  4                     BEGIN
; 2710          4216  4                     END_OF_TERM = 1 ;                               ! Signal completion
; 2711          4217  4                     !+
; 2712          4218  4                     !   Have to get rid of a possible status RMS$_TNS, Terminator
; 2713          4219  4                     !   Not Seen.  Assume success if we have reached this point.
; 2714          4220  4                     !   It is not advisable to overwrite data in the RAB but there
; 2715          4221  4                     !   is not way to avoid it in this case.
; 2716          4222  4                     !-
; 2717          4223  4                     RAB [RAB$L_STS] = RMS$_SUC ;
; 2718          4224  3                     END ;
; 2719          4225  3
; 2720          4226  2               END ;                                               ! End loop
; 2721          4227  1           END ;                                                   ! End COB$$PARTIAL_SEQ




                                               07FC 00000 COB$$PARTIAL_SEQ:
                                                                   .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10                    ; 4087
                   5E               04   C2 00002                  SUBL2   #4, SP                                              ;
                                    5A   D4 00005                  CLRL    END_OF_TERM                                         ; 4122
                   52       04      AC   D0 00007                  MOVL    PARAMETERS, R2                                      ; 4132
                   57       24      A2   9E 0000B                  MOVAB   36(R2), R7                                          ;
                   53       04      A2   D0 0000F                  MOVL    4(R2), PH                                           ; 4139
 58         1C     A2               67   C1 00013                  ADDL3   (R7), 28(R2), PH_PTR                                ; 4146
                   56               01   D0 00018                  MOVL    #1, NC_PTR                                          ; 4147
                   54       08      AC   9A 0001B                  MOVZBL  UNIT, R4                                            ; 4160
                                    5A   D5 0001F 1$:              TSTL    END_OF_TERM                                         ; 4154
                                    7F   12 00021                  BNEQ    10$                                                 ;
                   59     5240      8F   3C 00023                  MOVZWL  #21056, FUNC_VAL_2                                  ; 4158
                   55 00000000G0044      D0 00028                  MOVL    COB$$AL_WRITE_RAB[R4], RAB                          ; 4160
                                    5E   DD 00030                  PUSHL   SP                                                  ; 4161
                                    01   DD 00032                  PUSHL   #1                                                  ;
                          0220      8F   BB 00034                  PUSHR   #^M<R5,R9>                                          ;
            FDF3   CF               04   FB 00038                  CALLS   #4, COB$$RMS_GET                                    ;
                   50               6E   9A 0003D                  MOVZBL  TERM_CHAR, R0                                       ; 4180
                            30      A2   D5 00040                  TSTL    48(R2)                                              ; 4167
                                    13   12 00043                  BNEQ    3$                                                  ;
                   01               67   D1 00045                  CMPL    (R7), #1                                            ; 4176
                                    05   12 00048                  BNEQ    2$                                                  ;
              FF A843               1B   90 0004A                  MOVB    #27, -1(PH_PTR)[PH]                                 ; 4178
```

```
                    8843            50  90 0004F 2$:    MOVB    R0, (PH_PTR)+[PH]                           ; 4180
                              30    A2  D4 00053        CLRL    48(R2)                                      ; 4182
                                    10  11 00056        BRB     5$                                          ; 4167
                      01            67  D1 00058 3$:    CMPL    (R7), #1                                    ; 4192
                                    04  12 0005B        BNEQ    4$                                          ;
                 0C   A2            1B  90 0005D        MOVB    #27, 12(R2)                                 ; 4194
                 0C A246            50  90 00061 4$:    MOVB    R0, 12(R2)[NC_PTR]                          ; 4196
                                    56  D6 00066        INCL    NC_PTR                                      ; 4197
                                    67  D6 00068 5$:    INCL    (R7)                                        ; 4200
                 41   8F            50  91 0006A        CMPB    R0, #65                                     ; 4210
                                    06  1F 0006E        BLSSU   6$                                          ;
                 4D   8F            50  91 00070        CMPB    R0, #77                                     ;
                                    1E  1B 00074        BLEQU   9$                                          ;
                 50   8F            50  91 00076 6$:    CMPB    R0, #80                                     ; 4211
                                    06  1F 0007A        BLSSU   7$                                          ;
                 53   8F            50  91 0007C        CMPB    R0, #83                                     ;
                                    12  1B 00080        BLEQU   9$                                          ;
                 6C   8F            50  91 00082 7$:    CMPB    R0, #108                                    ; 4212
                                    06  1F 00086        BLSSU   8$                                          ;
                 79   8F            50  91 00088        CMPB    R0, #121                                    ;
                                    06  1B 0008C        BLEQU   9$                                          ;
                 7E   8F            50  91 0008E 8$:    CMPB    R0, #126                                    ; 4213
                                    8B  12 00092        BNEQ    1$                                          ;
                      5A            01  D0 00094 9$:    MOVL    #1, END_OF_TERM                             ; 4216
                 08   A5 00010001   8F  D0 00097        MOVL    #65537, 8(RAB)                              ; 4223
                                  FF7D  31 0009F        BRW     1$                                          ; 4154
                                        04 000A2 10$:   RET                                                 ; 4227

; Routine Size:  163 bytes,    Routine Base:  _COB$CODE + 0EDF
```

```
1 %SBTTL 'COB$$DELETE_KEY - Delete Key processing'
1 ROUTINE COB$$DELETE_KEY ( PARAMETERS : REF VECTOR,
1                           UNIT       : VECTOR [2,BYTE],
1                           FLAGS                           )  :  NOVALUE =
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Delete Key processing.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       PARAMETERS.mlu.ra  Contains data for this routine.
1 !
1 !       UNIT.rbu.va        Array of two unsigned byte integers.
1 !                          The first byte is the unit number designating the
1 !                          device from which the string is to be read.
1 !                          The second byte indicates whether the routine should
1 !                          abort or return to the calling program.
1 !
1 !       FLAGS.rlu.v        Screen enhancement flag.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
2     BEGIN
2
2       LOCAL
2           RAB       :  REF $RAB_DECL,
2           DELETE_BUF :  VECTOR [3, BYTE],
2           CHARS_OK   :  INITIAL (0),                    ! Characters not deleted
2           REST_LEN,                                     ! Length yet to be ACCEPTed
2                                                         ! after DELETE KEY was hit
2           REST_PTR ;                                    ! Where to put rest of
2                                                         ! chars on next $GET
2 !+
2 !  Delete Key processing.  Delete one character at a time.  Delete previous
2 !  character typed by backspacing, writing a space to  delete (overwrite)
2 !  character, and backspacing again to put cursor in position to continue
2 !  ACCEPTing data.
2 !  Hitting the DELETE KEY terminated the previous $GET, therefore perform
2 !  another $GET to read the rest of the data expected.
2 !
2 !  Bind PARAMETERS to other names.
2 !-
```

```
2
2       $BIND_PARAMETERS ;
2
2       IF ( .FLAGS AND V_NO_ECHO ) NEQ 0
2       THEN
2           !+
2           !  If characters were not echoed to terminal there is no need
2           !  to move cursor.
2           !-
3           BEGIN
3           DELETE_BUF [0] = NULL ;                         ! null
3           DELETE_BUF [1] = NULL ;
3           DELETE_BUF [2] = NULL ;
3           END
2       ELSE
3           BEGIN
3           DELETE_BUF [0] = BS ;                           ! Backspace
3           DELETE_BUF [1] = BLANK ;                        ! Space
3           DELETE_BUF [2] = BS ;                           ! Backspace
2           END    ;
2
2           WHILE .TERM_LOC EQL DEL_KEY DO
3               BEGIN                                       ! Begin Delete Loop
3
3               CHARS_READ = (.CHARS_READ - 1) ;    ! Decr # of valid input chars
3               CHARS_OK   = .CHARS_READ ;          ! Save for left border check
3
3               IF .CHARS_READ LSS 0
3               THEN CHARS_READ = 0 ;
3
3               !+
3               !  Calculations for $GET
3               !-
3               REST_LEN   = .ACC_SIZE - .CHARS_READ ;
3               REST_PTR   = .PUT_HERE [DSC$A_POINTER] + .CHARS_READ ;
3
3               !+
3               !  Check for too many deletes, do not delete more characters
3               !  then were input.  Ring bell to signal attempt to go beyond
3               !  left border.
3               !-
3               IF .CHARS_OK LSS 0
3               THEN
4                   BEGIN
4                   COB$$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ;
4                   END
3               ELSE
3                   !+
3                   !  $PUT to Delete one character.
3                   !-
3                   COB$$RMS_PUT_BUFFER ( DELETE_BUF [0], 3, .FLAGS ) ;
3
3               !+
3               !  Continue to Read input
3               !-
3
3               RAB = .COB$$AL_WRITE_RAB [ .UNIT[0] ] ;
```

```
; 2837           4342  3            COB$$RMS_GET ( .RAB, .FUNC_VAL, .REST_LEN, .REST_PTR ) ;
; 2838           4343  3
; 2839           4344  3            !+
; 2840           4345  3            !  Reset CHARS_READ - Update # of input chars read.
; 2841           4346  3            !  Reset TERM_SIZE and TERM_LOC - New terminator ( Note: this could
; 2842           4347  3            !                                            be the DELETE KEY again)
; 2843           4348  3            !-
; 2844           4349  3
; 2845           4350  3            CHARS_READ = .CHARS_READ + .RAB [RAB$W_RSZ] ;
; 2846           4351  3            TERM_SIZE  = .RAB [COB$$B_STV2_LEN] ;
; 2847           4352  3            TERM_LOC   = .RAB [COB$$B_STV0_TERM]; ! Terminator location.
; 2848           4353  3
; 2849           4354  3            !+
; 2850           4355  3            ! Check for partial sequence error
; 2851           4356  3            !-
; 2852           4357  3
; 2853           4358  3            IF .RAB [RAB$L_STS] EQL RMS$_PES
; 2854           4359  3            THEN
; 2855           4360  3                COB$$PARTIAL_SEQ ( .PARAMETERS, .UNIT ) ;
; 2856           4361  3
; 2857           4362  2            END ;                                    ! End Delete Loop
; 2858           4363  2
; 2859           4364  2 !+
; 2860           4365  2 !  Did the latest $GET come across a terminator ?
; 2861           4366  2 !  If so, set flag used by COB$$ILLEGAL_TERM to signal that the terminator
; 2862           4367  2 !  was encountered in this routine.
; 2863           4368  2 !-
; 2864           4369  2
; 2865           4370  2    IF .TERM_SIZE NEQ 0 OR .RAB [RAB$L_STS] EQL RMS$_EOF
; 2866           4371  2    THEN
; 2867           4372  2        TERM_FROM_DEL = 1 ;
; 2868           4373  2
; 2869           4374  1    END ;                                                ! End COB$$DELETE_KEY



                                                      01FC 00000 COB$$DELETE_KEY:
                                                                         .WORD   Save R2,R3,R4,R5,R6,R7,R8                   ; 4229
                                        5E           04 C2 00002         SUBL2   #4, SP                                      ;
                                                     58 D4 00005         CLRL    CHARS_OK                                    ; 4265
                                        53      04   AC D0 00007         MOVL    PARAMETERS, R3                              ; 4273
                                        55      1C   A3 9E 0000B         MOVAB   28(R3), R5                                  ;
                          07      0C    AC           09 E1 0000F         BBC     #9, FLAGS, 1$                               ; 4288
                                                     6E B4 00014         CLRW    DELETE_BUF                                  ; 4295
                                                02   AE 94 00016         CLRB    DELETE_BUF+2                                ; 4297
                                                     09 11 00019         BRB     2$                                          ; 4288
                                        6E    2008   8F B0 0001B 1$:     MOVW    #8200, DELETE_BUF                           ; 4301
                                  02    AE           08 90 00020         MOVB    #8, DELETE_BUF+2                            ; 4303
                                        54      08   AC 9A 00024 2$:     MOVZBL  UNIT, R4                                    ; 4341
                            0000007F    8F      28   A3 D1 00028 3$:     CMPL    40(R3), #127                                ; 4306
                                                     6F 12 00030         BNEQ    7$                                          ;
                                                     65 D7 00032         DECL    (R5)                                        ; 4309
                                        58           65 D0 00034         MOVL    (R5), CHARS_OK                              ; 4310
                                                     02 18 00037         BGEQ    4$                                          ; 4312
                                                     65 D4 00039         CLRL    (R5)                                        ; 4313
```

```
                     57        18  A3  3C 0003B 4$:      MOVZWL  24(R3), REST_LEN                          ; 4318
                     57            65  C2 0003F          SUBL2   (R5), REST_LEN                            ;
          56     04  A3            65  C1 00042          ADDL3   (R5), 4(R3), REST_PTR                     ; 4319
                                   58  D5 00047          TSTL    CHARS_OK                                  ; 4326
                                   0C  18 00049          BGEQ    5$                                        ;
                               0C  AC  DD 0004B          PUSHL   FLAGS                                     ; 4329
                                   02  DD 0004E          PUSHL   #2                                        ;
               FDD9  CF            02  FB 00050          CALLS   #2, COB$$RMS_PUT_BYTE                     ;
                                   0D  11 00055          BRB     6$                                        ; 4326
                               0C  AC  DD 00057 5$:      PUSHL   FLAGS                                     ; 4335
                                   03  DD 0005A          PUSHL   #3                                        ;
                               08  AE  9F 0005C          PUSHAB  DELETE_BUF                                ;
               FE5C  CF            03  FB 0005F          CALLS   #3, COB$$RMS_PUT_BUFFER                   ;
                     52 00000000G00044 D0 00064 6$:      MOVL    COB$$AL_WRITE_RAB[R4], RAB                ; 4341
                                   56  DD 0006C          PUSHL   REST_PTR                                  ; 4342
                                   57  DD 0006E          PUSHL   REST_LEN                                  ;
                               20  A3  DD 00070          PUSHL   32(R3)                                    ;
                                   52  DD 00073          PUSHL   RAB                                       ;
               FD13  CF            04  FB 00075          CALLS   #4, COB$$RMS_GET                          ;
                     50        22  A2  3C 0007A          MOVZWL  34(RAB), R0                               ; 4350
                     65            50  C0 0007E          ADDL2   R0, (R5)                                  ;
                 24  A3        0E  A2  9A 00081          MOVZBL  14(RAB), 36(R3)                           ; 4351
                 28  A3        0C  A2  9A 00086          MOVZBL  12(RAB), 40(R3)                           ; 4352
           000181C8  8F        08  A2  D1 0008B          CMPL    8(RAB), #98760                            ; 4358
                                   93  12 00093          BNEQ    3$                                        ;
                               08  AC  DD 00095          PUSHL   UNIT                                      ; 4360
                                   53  DD 00098          PUSHL   R3                                        ;
               FEBE  CF            02  FB 0009A          CALLS   #2, COB$$PARTIAL_SEQ                      ;
                                   87  11 0009F          BRB     3$                                        ; 4306
                               24  A3  D5 000A1 7$:      TSTL    36(R3)                                    ; 4370
                                   0A  12 000A4          BNEQ    8$                                        ;
           0001827A  8F        08  A2  D1 000A6          CMPL    8(RAB), #98938                            ;
                                   04  12 000AE          BNEQ    9$                                        ;
                 34  A3            01  D0 000B0 8$:      MOVL    #1, 52(R3)                                ; 4372
                                       04 000B4 9$:      RET                                               ; 4374

; Routine Size:  181 bytes,    Routine Base:  _COB$CODE + 0F82
```

```
1 %SBTTL 'COB$$ILLEGAL_TERM - Illegal Terminator'
1 ROUTINE COB$$ILLEGAL_TERM ( PARAMETERS : REF VECTOR,
1                             UNIT       : VECTOR [2,BYTE],
1                             FLAGS,
1                             KEY        : REF $STR$DESCRIPTOR )  :  NOVALUE =
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Terminator from previous $GET was illegal - ring terminal bell to
1 !       signal this.  Perform another $GET of length 1 to look for another
1 !       terminator.  Verify this new terminator.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       PARAMETERS.mlu.ra  Contains data for this routine.
1 !
1 !       UNIT.rbu.va        Array of two unsigned byte integers.
1 !                          The first byte is the unit number designating the
1 !                          device from which the string is to be read.
1 !                          The second byte indicates whether the routine should
1 !                          abort or return to the calling program.
1 !
1 !       FLAGS.rlu.v        Screen enhancement flag.
1 !
1 !       KEY.wt.ds          Destination of the receiving field of the control key.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
2    BEGIN
2       LOCAL
2          RAB :  REF $RAB_DECL,
2          FUNC_VAL_2,                                        ! QIO Function Modifiers
2          NO_BELL              :  INITIAL (0),               ! =0 ring bell, =1 don't
2          LOOK_FOR_TERM        :  INITIAL (0),               ! =1 buffer full, $GET
2                                                             ! only for a terminator
2          REST_LEN,                                          ! Length yet to be input
2          REST_PTR ;                                         ! Where to put rest of
2                                                             ! input data
2 !+
2 ! Bind PARAMETERS to other names.
2 !-
2
2    $BIND_PARAMETERS ;
```

```
2
2  !+
2  !  Note :  If COB$$DELETE_KEY was called before this routine some
2  !  special handling is necessary.
2  !         It is possible a previous call to COB$$DELETE_KEY would have
2  !         filled the input buffer without coming across a terminator.
2  !         When the input buffer is full - look for terminator only.
2  !
2  !         It is also possible that COB$$DELETE_KEY came across a terminator,
2  !         therefore it is only necessary to verify the terminator not
2  !         perform another $GET.  This is flagged by TERM_FROM_DEL = 1.
2  !-
2
2      WHILE .LEGAL EQL 0 DO
3          BEGIN                                                ! Begin Term Loop
3          IF .NO_BELL EQL 0 AND .TERM_FROM_DEL EQL 0           ! NO_BELL is set in
3          THEN                                                 ! $VERIFY_TERMINATOR
3              !+
3              !  Ring bell to signal illegal terminator.
3              !  Don't ring bell if processing the Delete key, or if the
3              !  terminator has come from COB$$DELETE_KEY (wait for
3              !  $VERIFY_TERMINATOR to check terminator).
3              !-
4              BEGIN
4              COB$$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ;
4              END
3          ELSE                                                 ! Reset - next time
3              NO_BELL = 0 ;                                    ! ring bell
3          !+
3          !  Is there still data yet to be input ?
3          !-
3
3          IF .TERM_FROM_DEL EQL 0
3          THEN
4              BEGIN                                            ! Begin TERM_FROM_DEL=0
4
4              IF .ACC_SIZE GTR .CHARS_READ
4              THEN
5                  BEGIN
5
5                  !+
5                  !  Calculations for $GET.
5                  !-
5                  REST_LEN   = .ACC_SIZE - .CHARS_READ ;
5                  REST_PTR   = .PUT_HERE [DSC$A_POINTER] + .CHARS_READ ;
5
5                  !+
5                  !  NEVER do a Read of 0 length, this causes an infinite loop
5                  !  of bell ringing.
5                  !-
5                  IF .REST_LEN EQL 0 THEN REST_LEN = 1 ;
5
5                  RAB = .COB$$AL_WRITE_RAB [ .UNIT[0] ] ;
5                  COB$$RMS_GET ( .RAB, .FUNC_VAL, .REST_LEN, .REST_PTR ) ;
5
5                  !+
5                  !  Update CHARS_READ, TERM_SIZE and TERM_LOC.
```

```
5               !-
5
5           CHARS_READ = .CHARS_READ + .RAB [RAB$W_RSZ] ;
5           TERM_SIZE  = .RAB [COB$$B_STV2_LEN] ;
5           TERM_LOC   = .RAB [COB$$B_STV0_TERM] ;
5
4           END ;
4
4       !+
4       !  $GET buffer filled but no terminator seen - TERM_SIZE = 0
4       !  Do 1 character reads until you hit a terminator that
4       !  you can then attempt to verify.
4       !  Also trap an End of File ^Z here and do not perform
4       !  another $GET, $VERIFY_TERMINATOR will take care of the ^Z.
4       !  .LOOK_FOR_TERM EQL 1 case -> came into this routine with
4       !  $GET buffer filled but illegal terminator, therefore
4       !  we are looking only for a terminator.
4       !-
4
4       IF .ACC_SIZE EQL .CHARS_READ
4       THEN LOOK_FOR_TERM = 1 ;
5       WHILE (.TERM_SIZE EQL 0 AND .RAB [RAB$L_STS] NEQ RMS$_EOF)
4              OR (.LOOK_FOR_TERM EQL 1 ) DO
4
5           BEGIN                                   ! Begin 1 char $GET
5           REST_PTR   = .PUT_HERE [DSC$A_POINTER] + .CHARS_READ ;
5
5           FUNC_VAL_2 = TRM$M_TM_ESCAPE + TRM$M_TM_NOFILTR
5                                    + TRM$M_TM_TRMNOECHO + TRM$M_TM_NOECHO ;
5
5           RAB = .COB$$AL_WRITE_RAB [ .UNIT[0] ] ;
5           COB$$RMS_GET ( .RAB, .FUNC_VAL_2, 1, .REST_PTR ) ;
5
5           !+
5           !  Set TERM_SIZE and TERM_IN_NEXT before possible
5           !  call to COB$$PARTIAL_SEQ.
5           !  If user attempts to input data other than a
5           !  terminator - error.
5           !-
5
5           IF .RAB [RAB$W_RSZ] NEQ 0
5           THEN
6               BEGIN
6               COB$$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ;   ! Error.
6               TERM_SIZE = 0 ;
6               END
5           ELSE                                    ! Terminator seen.
5               IF .RAB [RAB$L_STS] EQL RMS$_EOF
5               THEN
5                   !+
5                   !  NOTE:  When Control Z is typed in as the only
5                   !  input to a $GET it is not recorded in
5                   !  RAB [COB$$B_STV2_LEN] therefore, pull out of
5                   !  loop and let $VERIFY_TERMINATOR handle the ^Z,
5                   !  but first you have to load the ^Z in
5                   !  RAB [COB$$B_STV2_LEN]  as this is where
5                   !  $VERIFY_TERMINATOR looks for it.
```

```
; 3042       4546  5                            !-
; 3043       4547  6                            BEGIN
; 3044       4548  6                            TERM_SIZE = 1 ;
; 3045       4549  6                            RAB [COB$$B_STVO_TERM] = CZ ;
; 3046       4550  6                            LOOK_FOR_TERM = 0 ;             ! Set to get out of loop
; 3047       4551  6                            END
; 3048       4552  5                        ELSE
; 3049       4553  6                            BEGIN
; 3050       4554  6                            LOOK_FOR_TERM = 0 ;             ! Set to get out of loop
; 3051       4555  6                            TERM_SIZE = .RAB [COB$$B_STV2_LEN] ;
; 3052       4556  6                            TERM_LOC  = .RAB [COB$$B_STVO_TERM] ;
; 3053       4557  5                            END ;
; 3054       4558  4                    END ;                                   ! End 1 char $GET
; 3055       4559  4
; 3056       4560  4                !+
; 3057       4561  4                ! Check for partial sequence error
; 3058       4562  4                !-
; 3059       4563  4
; 3060       4564  4                IF .RAB [RAB$L_STS] EQL RMS$_PES
; 3061       4565  4                THEN
; 3062       4566  4                    COB$$PARTIAL_SEQ ( .PARAMETERS, .UNIT ) ;
; 3063       4567  4
; 3064       4568  3                END ;                                       ! End TERM_FROM_DEL=0
; 3065       4569  3
; 3066       4570  3            !+
; 3067       4571  3            ! Now have a Terminator in PUT_HERE.  Reset flags.  Call
; 3068       4572  3            ! macro to verify Terminator.
; 3069       4573  3            !-
; 3070       4574  3
; 3071       4575  3            TERM_PTR = .PUT_HERE [DSC$A_POINTER] + .CHARS_READ ;
; 3072       4576  3            TERM_FROM_DEL = 0 ;
; 3073       4577  3            TERM_IN_NEXT = 0 ;
; 3074       4578  3            $VERIFY_TERMINATOR ;
; 3075       4579  3
; 3076       4580  2            END ;                                           ! End Term Loop
; 3077       4581  2
; 3078       4582  1        END ;                                               ! End COB$$ILLEGAL_TERM



                                        0FFC 00000 COB$$ILLEGAL_TERM:
                                                             .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11            ; 4376
                  5E            04   C2 00002                SUBL2   #4, SP                                          ;
                                5A   7C 00005                CLRQ    LOOK_FOR_TERM                                   ; 4417
                  52      04    AC   D0 00007                MOVL    PARAMETERS, R2                                  ; 4425
                  56      1C    A2   9E 0000B                MOVAB   28(R2), R6                                      ;
                  55      24    A2   9E 0000F                MOVAB   36(R2), R5                                      ;
                  57      38    A2   9E 00013                MOVAB   56(R2), R7                                      ;
                                67   D5 00017 1$:            TSTL    (R7)                                            ; 4445
                                01   13 00019                BEQL    2$                                              ;
                                     04 0001B                RET                                                     ;
                                5B   D5 0001C 2$:            TSTL    NO_BELL                                         ; 4447
                                11   12 0001E                BNEQ    3$                                              ;
                          34    A2   D5 00020                TSTL    52(R2)                                          ;
                                0C   12 00023                BNEQ    3$                                              ;
```

```
                                            0C    AC DD 00025         PUSHL   FLAGS                              ; 4456
                                                  02 DD 00028         PUSHL   #2                                 ;
                          FD4A    CF              02 FB 0002A         CALLS   #2, COB$$RMS_PUT_BYTE              ;
                                                  02 11 0002F         BRB     4$                                 ; 4447
                                                  5B D4 00031 3$:     CLRL    NO_BELL                            ; 4459
                                            34    A2 D5 00033 4$:     TSTL    52(R2)                             ; 4464
                                                  03 13 00036         BEQL    5$                                 ;
                                                00CE 31 00038         BRW     15$                                ;
    66      18  A2                10              00 ED 0003B 5$:     CMPZV   #0, #16, 24(R2), (R6)              ; 4468
                                                  3D 15 00041         BLEQ    7$                                 ;
                                  59        18    A2 3C 00043         MOVZWL  24(R2), REST_LEN                   ; 4475
                                  59              66 C2 00047         SUBL2   (R6), REST_LEN                     ;
                58          04    A2              66 C1 0004A         ADDL3   (R6), 4(R2), REST_PTR              ; 4476
                                                  59 D5 0004F         TSTL    REST_LEN                           ; 4482
                                                  03 12 00051         BNEQ    6$                                 ;
                                  59              01 D0 00053         MOVL    #1, REST_LEN                       ;
                                  50        08    AC 9A 00056 6$:     MOVZBL  UNIT, R0                           ; 4484
                                  53   00000000G0040 D0 0005A         MOVL    COB$$AL_WRITE_RAB[R0], RAB         ;
                                                  58 DD 00062         PUSHL   REST_PTR                           ; 4485
                                                  59 DD 00064         PUSHL   REST_LEN                           ;
                                            20    A2 DD 00066         PUSHL   32(R2)                             ;
                                                  53 DD 00069         PUSHL   RAB                                ;
                          FC68    CF              04 FB 0006B         CALLS   #4, COB$$RMS_GET                   ;
                                  50        22    A3 3C 00070         MOVZWL  34(RAB), R0                        ; 4491
                                  66              50 C0 00074         ADDL2   R0, (R6)                           ;
                                  65        0E    A3 9A 00077         MOVZBL  14(RAB), (R5)                      ; 4492
                            28    A2        0C    A3 9A 0007B         MOVZBL  12(RAB), 40(R2)                    ; 4493
    66      18  A2                10              00 ED 00080 7$:     CMPZV   #0, #16, 24(R2), (R6)              ; 4508
                                                  03 12 00086         BNEQ    8$                                 ;
                                  5A              01 D0 00088         MOVL    #1, LOOK_FOR_TERM                  ; 4509
                                  54        08    AC 9A 0008B 8$:     MOVZBL  UNIT, R4                           ; 4519
                                                  65 D5 0008F 9$:     TSTL    (R5)                               ; 4510
                                                  0A 12 00091         BNEQ    10$                                ;
                      0001827A    8F        08    A3 D1 00093         CMPL    8(RAB), #98938                     ;
                                                  05 12 0009B         BNEQ    11$                                ;
                                  01              5A D1 0009D 10$:    CMPL    LOOK_FOR_TERM, #1                  ; 4511
                                                  53 12 000A0         BNEQ    14$                                ;
                58          04    A2              66 C1 000A2 11$:    ADDL3   (R6), 4(R2), REST_PTR              ; 4514
                                  6E      5240    8F 3C 000A7         MOVZWL  #21056, FUNC_VAL_2                 ; 4517
                                  53   00000000G0044 D0 000AC         MOVL    COB$$AL_WRITE_RAB[R4], RAB         ; 4519
                                                  58 DD 000B4         PUSHL   REST_PTR                           ; 4520
                                                  01 DD 000B6         PUSHL   #1                                 ;
                                            08    AE DD 000B8         PUSHL   FUNC_VAL_2                         ;
                                                  53 DD 000BB         PUSHL   RAB                                ;
                          FC16    CF              04 FB 000BD         CALLS   #4, COB$$RMS_GET                   ;
                                            22    A3 B5 000C2         TSTW    34(RAB)                            ; 4529
                                                  0E 13 000C5         BEQL    12$                                ;
                                            0C    AC DD 000C7         PUSHL   FLAGS                              ; 4532
                                                  02 DD 000CA         PUSHL   #2                                 ;
                          FCA8    CF              02 FB 000CC         CALLS   #2, COB$$RMS_PUT_BYTE              ;
                                                  65 D4 000D1         CLRL    (R5)                               ; 4533
                                                  BA 11 000D3         BRB     9$                                 ; 4529
                                                  5A D4 000D5 12$:    CLRL    LOOK_FOR_TERM                      ; 4550
                      0001827A    8F        08    A3 D1 000D7         CMPL    8(RAB), #98938                     ; 4536
                                                  09 12 000DF         BNEQ    13$                                ;
                                  65              01 D0 000E1         MOVL    #1, (R5)                           ; 4548
                            0C    A3              1A 90 000E4         MOVB    #26, 12(RAB)                       ; 4549
```

```
                                    A5 11 000E8         BRB     9$                              ; 4536
                        65    0E    A3 9A 000EA 13$:    MOVZBL  14(RAB), (R5)                   ; 4555
                    28  A2    0C    A3 9A 000EE         MOVZBL  12(RAB), 40(R2)                 ; 4556
                                    9A 11 000F3         BRB     9$                              ; 4510
              000181C8  8F    08    A3 D1 000F5 14$:    CMPL    8(RAB), #98760                  ; 4564
                                    0A 12 000FD         BNEQ    15$
                              08    AC DD 000FF         PUSHL   UNIT                            ; 4566
                                    52 DD 00102         PUSHL   R2
                  FD9F  CF          02 FB 00104         CALLS   #2, COB$$PARTIAL_SEQ
2C    A2            04  A2          66 C1 00109 15$:    ADDL3   (R6), 4(R2), 44(R2)             ; 4575
                              30    A2 7C 0010F         CLRQ    48(R2)                          ; 4577
                        01          65 D1 00112         CMPL    (R5), #1
                                    3B 12 00115         BNEQ    18$
                    2C  A2    0C    A3 9E 00117         MOVAB   12(RAB), 44(R2)
                        51    0C    A3 9A 0011C         MOVZBL  12(RAB), R1
                        09          51 91 00120         CMPB    R1, #9
                                    05 13 00123         BEQL    16$
                        0D          51 91 00125         CMPB    R1, #13
                                    0D 12 00128         BNEQ    17$
                        50    10    AC D0 0012A 16$:    MOVL    KEY, R0
                                    62 13 0012E         BEQL    21$
                    04  B0    2C    B2 90 00130         MOVB    @44(R2), @4(R0)
                                    5B 11 00135         BRB     21$
                        1A          51 91 00137 17$:    CMPB    R1, #26
                                    24 13 0013A         BEQL    19$
                    7F  8F          51 91 0013C         CMPB    R1, #127
                                    55 12 00140         BNEQ    22$
                        7E    08    AC 7D 00142         MOVQ    UNIT, -(SP)
                                    52 DD 00146         PUSHL   R2
                  FDFE  CF          03 FB 00148         CALLS   #3, COB$$DELETE_KEY
                        5B          01 D0 0014D         MOVL    #1, NO_BELL
                                    49 11 00150         BRB     23$
                                    66 D5 00152 18$:    TSTL    (R6)
                                    25 12 00154         BNEQ    20$
              0001827A  8F    08    A3 D1 00156         CMPL    8(RAB), #98938
                                    1B 12 0015E         BNEQ    20$
      32            0C  AC          0B E0 00160 19$:    BBS     #11, FLAGS, 22$
                              0C    AC DD 00165         PUSHL   FLAGS
                                    52 DD 00168         PUSHL   R2
                 0000V  CF          02 FB 0016A         CALLS   #2, COB$$CLEAN_UP
                              10    AC DD 0016F         PUSHL   KEY
                              08    AC DD 00172         PUSHL   UNIT
                  FD05  CF          02 FB 00175         CALLS   #2, COB$$CONTROL_Z
                                       04 0017A         RET
                              10    AC D5 0017B 20$:    TSTL    KEY
                                    17 13 0017E         BEQL    22$
                              10    AC DD 00180         PUSHL   KEY
                                    65 DD 00183         PUSHL   (R5)
                              2C    A2 9F 00185         PUSHAB  44(R2)
             00000000G  00          03 FB 00188         CALLS   #3, COB$$CONTROL_KEY
                        05          50 E9 0018F         BLBC    R0, 22$
                        67          01 D0 00192 21$:    MOVL    #1, (R7)
                                    04 11 00195         BRB     23$
                                    67 D4 00197 22$:    CLRL    (R7)
                                    65 D4 00199         CLRL    (R5)
                                  FE79 31 0019B 23$:    BRW     1$                              ; 4445
                                       04 0019E         RET                                     ; 4582
```

```
; Routine Size:  415 bytes,    Routine Base:  _COB$CODE + 1037
```

```
1 %SBTTL 'COB$$CLEAN_UP - Clean up for VAX COBOL'
1 ROUTINE COB$$CLEAN_UP ( PARAMETERS : REF VECTOR,
1                         FLAGS
1                       )  :  NOVALUE =
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Perform clean up before returning control to VAX COBOL.
1 !       Position cursor after FIELD of POTECTED ACCEPT, $PUT to turn
1 !       attributes off, and determine if ADVANCING is needed.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       PARAMETERS.mlu.ra  Contains data for this routine.
1 !
1 !       FLAGS.rlu.v        Screen enhancement flag.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !--
2     BEGIN
2       LOCAL
2           RAB  :  REF $RAB_DECL ;
2
2       !+
2       ! Bind PARAMETERS to other names.
2       !-
2
2       $BIND_PARAMETERS ;
2
2       !+
2       ! Position cursor after FIELD of POTECTED Read.
2       ! This code is necessary if the # of characters input is less than
2       ! the # of characters expected.  Move cursor the difference of the
2       ! two numbers.
2       ! If DEFAULT has been used move cursor the whole length of the expected
2       ! size.
2       !-
2
2       IF .YES_PROTECT
2       THEN
3           BEGIN
3               LOCAL
3                   MOVE_CURSOR  :  INITIAL (0),            ! Flag
3                   MOVE_NUM ;                              ! # of positions to
```

```
; 3137           4640 3                                                           !  move cursor
; 3138           4641 3         IF .YES_DEFAULT NEQ 0
; 3139           4642 3         THEN
; 3140           4643 4             BEGIN
; 3141           4644 4             MOVE_NUM = .ACC_SIZE ;                      !  DEFAULT used
; 3142           4645 4             MOVE_CURSOR = 1 ;
; 3143           4646 4             END
; 3144           4647 3         ELSE
; 3145           4648 3             IF .CHARS_READ LSS .ACC_SIZE                !  # of chars input is
; 3146           4649 3             THEN                                        !  less than expected
; 3147           4650 4                 BEGIN
; 3148           4651 4                 MOVE_NUM = .ACC_SIZE - .CHARS_READ ;
; 3149           4652 4                 MOVE_CURSOR = 1 ;
; 3150           4653 3                 END ;
; 3151           4654 3
; 3152           4655 3         IF .MOVE_CURSOR NEQ 0
; 3153           4656 3         THEN
; 3154           4657 4             BEGIN
; 3155           4658 4                 LOCAL
; 3156           4659 4                     SPACE_BUF : VECTOR [200,BYTE] ;
; 3157           4660 4
; 3158           4661 4             CH$FILL ( BLANK, .MOVE_NUM, SPACE_BUF [0] ) ; ! # of spaces to move
; 3159           4662 4             COB$$RMS_PUT_BUFFER ( SPACE_BUF [0], .MOVE_NUM, .FLAGS ) ;  ! cursor
; 3160           4663 3             END ;
; 3161           4664 2     END ;
; 3162           4665 2
; 3163           4666 2     !+
; 3164           4667 2     !  $PUT to turn attributes off.
; 3165           4668 2     !  If no attributes were turned on, there is no need to turn them off.
; 3166           4669 2     !  OFF_BUF holds escape sequence to turn attributes off.  OFF_LEN holds
; 3167           4670 2     !  the length of that sequence.
; 3168           4671 2     !-
; 3169           4672 2
; 3170           4673 2     IF .PUT_FLAG NEQ 0
; 3171           4674 2     THEN
; 3172           4675 2         COB$$RMS_PUT_BUFFER ( OFF_BUF [0], .OFF_LEN, .FLAGS ) ;
; 3173           4676 2
; 3174           4677 2     !+
; 3175           4678 2     !  Determine if ADVANCING is requested.
; 3176           4679 2     !  If bit 10 = 0 advancing.  If bit 10 = 1 no advancing.
; 3177           4680 2     !  Set COB$$AB_PREV[0] - also depending on bit 10, to flag to next COBOL
; 3178           4681 2     !  statement that advancing/no advancing is required following this
; 3179           4682 2     !  ACCEPT statement.
; 3180           4683 2     !-
; 3181           4684 2
; 3182           4685 2     IF (.FLAGS AND V_ADV) NEQ 0
; 3183           4686 2     THEN
; 3184           4687 2         COB$$AB_PREV[0] = ACC_DNA                       ! Signal Do Not Advance
; 3185           4688 2     ELSE
; 3186           4689 2         !+
; 3187           4690 2         !  Echo carriage return to screen if advancing is called for.
; 3188           4691 2         !-
; 3189           4692 3         BEGIN
; 3190           4693 3         COB$$RMS_PUT_BYTE ( CARR_RET, .FLAGS ) ;
; 3191           4694 3         COB$$AB_PREV[0] = ACC_ADV ;                     ! Signal ADVance
; 3192           4695 2         END ;
; 3193           4696 1     END ;                                               ! End of COB$$CLEAN_UP
```

```
                                                         01FC 00000 COB$$CLEAN_UP:
                                                                          .WORD   Save R2,R3,R4,R5,R6,R7,R8             ; 4584
                                        58 00000000G  00   9E 00002       MOVAB   COB$$AB_PREV, R8                      ;
                                        5E      FF38  CE   9E 00009       MOVAB   -200(SP), SP                          ;
                                        56        04  AC   D0 0000E       MOVL    PARAMETERS, R6                        ; 4617
                                        38        3C  A6   E9 00012       BLBC    60(R6), 4$                            ; 4634
                                                      50   D4 00016       CLRL    MOVE_CURSOR                           ; 4636
                                                  40  A6   D5 00018       TSTL    64(R6)                                ; 4641
                                                      06   13 0001B       BEQL    1$                                    ;
                                        57        18  A6   3C 0001D       MOVZWL  24(R6), MOVE_NUM                      ; 4644
                                                      11   11 00021       BRB     2$                                    ; 4645
    1C   A6      18   A6                10            00   ED 00023 1$:   CMPZV   #0, #16, 24(R6), 28(R6)               ; 4648
                                                      0B   15 0002A       BLEQ    3$                                    ;
                                        57        18  A6   3C 0002C       MOVZWL  24(R6), MOVE_NUM                      ; 4651
                                        57        1C  A6   C2 00030       SUBL2   28(R6), MOVE_NUM                      ;
                                        50            01   D0 00034 2$:   MOVL    #1, MOVE_CURSOR                       ; 4652
                                                      50   D5 00037 3$:   TSTL    MOVE_CURSOR                           ; 4655
                                                      13   13 00039       BEQL    4$                                    ;
         57           20                6E            00   2C 0003B       MOVC5   #0, (SP), #32, MOVE_NUM, SPACE_BUF    ; 4661
                                                      6E      00040                                                     ;
                                                  08  AC   DD 00041       PUSHL   FLAGS                                 ; 4662
                                                      57   DD 00044       PUSHL   MOVE_NUM                              ;
                                                  08  AE   9F 00046       PUSHAB  SPACE_BUF                             ;
                             FC1E  CF                 03   FB 00049       CALLS   #3, COB$$RMS_PUT_BUFFER               ;
                                                  44  A6   D5 0004E 4$:   TSTL    68(R6)                                ; 4673
                                                      0E   13 00051       BEQL    5$                                    ;
                                                  08  AC   DD 00053       PUSHL   FLAGS                                 ; 4675
                                                  54  A6   DD 00056       PUSHL   84(R6)                                ;
                                                  48  A6   9F 00059       PUSHAB  72(R6)                                ;
                             FC0B  CF                 03   FB 0005C       CALLS   #3, COB$$RMS_PUT_BUFFER               ;
                      04       08  AC                 0A   E1 00061 5$:   BBC     #10, FLAGS, 6$                        ; 4685
                                   68                 05   90 00066       MOVB    #5, COB$$AB_PREV                      ; 4687
                                                           04 00069       RET                                           ;
                                                  08  AC   DD 0006A 6$:   PUSHL   FLAGS                                 ; 4693
                                                      7E   D4 0006D       CLRL    -(SP)                                 ;
                             FB66  CF                 02   FB 0006F       CALLS   #2, COB$$RMS_PUT_BYTE                 ;
                                   68                 04   90 00074       MOVB    #4, COB$$AB_PREV                      ; 4694
                                                           04 00077       RET                                           ; 4696

; Routine Size:  120 bytes,    Routine Base:  _COB$CODE + 11D6
```

```
1 %SBTTL 'COB$$RPG_CLEAN_UP - Clean up for VAX RPG'
1 ROUTINE COB$$RPG_CLEAN_UP ( FLAGS )  :  NOVALUE =
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Perform clean up before returning control to VAX RPG.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       FLAGS.rlu.v     Screen enhancement flag.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
2     BEGIN
2
2     !+
2     !  Determine if ADVANCING is requested.
2     !  If bit 10 = 0 advancing.  If bit 10 = 1 no advancing.
2     !  Set COB$$AB_PREV[0] - also depending on bit 10, to flag to next COBOL
2     !  statement that advancing/no advancing is required following this
2     !  ACCEPT statement.
2     !-
2
2     IF (.FLAGS AND V_ADV) NEQ 0
2     THEN
2         COB$$AB_PREV[0] = ACC_DNA                          ! Signal Do Not Advance
2     ELSE
3         BEGIN
3         !+
3         !  Echo carriage return to screen if advancing is called for.
3         !-
3         COB$$RMS_PUT_BYTE ( CARR_RET, .FLAGS ) ;
3         COB$$AB_PREV[0] = ACC_ADV ;                        ! Signal ADVance
2         END;
2
1     END ;                                                ! End of COB$$RPG_CLEAN_UP



                              0004 00000 COB$$RPG_CLEAN_UP:
                                         .WORD   Save R2                                  ; 4698
                              52 00000000G  00  9E 00002          MOVAB   COB$$AB_PREV, R2                                  ;
                  04          6C                2A  E1 00009          BBC     #42, FLAGS, 1$                                    ; 4734
                              62                05  90 0000D          MOVB    #5, COB$$AB_PREV                                  ; 4736
                                                    04 00010          RET                                                       ;
                                          04    AC  DD 00011 1$:      PUSHL   FLAGS                                             ; 4742
                                                7E  D4 00014          CLRL    -(SP)                                             ;
                        FB47  CF                02  FB 00016          CALLS   #2, COB$$RMS_PUT_BYTE                             ;
                              62                04  90 0001B          MOVB    #4, COB$$AB_PREV                                  ; 4743
                                                    04 0001E          RET                                                       ; 4746

; Routine Size:  31 bytes,    Routine Base:  _COB$CODE + 124E
```

```
1 %SBTTL 'COB$$FORMAT_FOUR - Format Four'
1 ROUTINE COB$$FORMAT_FOUR  ( UNIT          :  VECTOR [2,BYTE],
1                             FLAGS,
1                             KEY           :  REF $STR$DESCRIPTOR
1                           ) =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine handles VAX COBOL ACCEPT Statement FORMAT FOUR,
1 !       Control Key.
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       UNIT.rbu.va     Array of two unsigned byte integers.
1 !                       The first byte is the unit number designating the
1 !                       device from which the string is to be read.
1 !                       The second byte indicates whether the routine should
1 !                       abort or return to the calling program.
1 !                         Byte 2 = 0  -  routine will abort on control z
1 !                                        and reprompt on conversion errors.
1 !                                = 1  -  ( AT END )
1 !                                        routine will return to calling program
1 !                                        on control z and reprompt on conversion
1 !                                        errors.
1 !                                = 2  -  ( ON EXCEPTION )
1 !                                        routine will return to calling program
1 !                                        on control z and conversion errors.
1 !
1 !       FLAGS.rlu.v     Screen enhancement flag;
1 !
1 !       KEY.wt.ds       Destination of the receiving field of the control key.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !
1 !--
1
2    BEGIN
2
2    LOCAL
2       RAB             :  REF $RAB_DECL,
2       FUNC_VAL,                               ! Read QIO Function Modifiers
2                                               ! used in item list by RMS
2       TERM_PTR,                               ! Pointer to terminator in buffer
```

```
  2             NEXT_CHAR         :  VECTOR [10,BYTE],    ! Buffer to hold terminator sequence
  2             LEGAL             :  INITIAL (0),         ! = 0 if illegal terminator hit
  2             TMASK             :  VECTOR [2] ;         ! Longform terminator mask
  2
  2     !+
  2     !   Terminator mask - EVERY key is treated as a terminator.  Each key pressed
  2     !   is checked for validity as a terminator.
  2     !   Valid terminators are Carriage Return, Tab, Control Z, Arrow keys,
  2     !   PF keys, and the PROFESSIONAL Editing and Top Row Function keys.
  2     !-
  2
  2         TMASK [0] = 32 ;
  2         TMASK [1] = UPLIT (-1, -1, -1, -1, -1, -1, -1, -1 ) ;
  2
  2     !+
  2     !   Ring the terminal bell if user requests.
  2     !-
  2
  2         IF ( .FLAGS AND V_BELL ) NEQ 0
  2         THEN
  2             COB$$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ;
  2
  2     !+
  2     !   Determine FUNC_VAL - QIO Function Modifiers used by RMS $GET Service.
  2     !   Set TRM$M_TM_NOECHO to suppress echoing of input characters to the terminal.
  2     !   Set TRM$M_TM_ESCAPE to allow Escape sequences to act as terminators (Arrow
  2     !   keys and PF keys and the Professional editing and top row function keys).
  2     !   Set TRM$M_TM_NOFILTR to allow this routine to handle the DELETE KEY. (not a
  2     !   valid terminator).
  2     !   Set TRM$M_TM_TRMNOECHO to suppress echoing of the termination character
  2     !   (COB$$AB_PREV handles advancing / no advancing).
  2     !-
  2
  2         FUNC_VAL = TRM$M_TM_ESCAPE + TRM$M_TM_NOFILTR + TRM$M_TM_TRMNOECHO
  2                                                        + TRM$M_TM_NOECHO ;
  2
P 2         $ITMLST_INIT (ITMLST = XAB_ITMLST,                         ! Item list for $GET
P 2                         (ITMCOD = TRM$_MODIFIERS,
P 2                          BUFSIZ = 0,
P 2                          BUFADR = .FUNC_VAL),
P 2                         (ITMCOD = TRM$_TERM,
P 2                          BUFSIZ = 32,                              ! 32 bytes in TMASK
  2                          BUFADR = .TMASK[1]) ) ;
  2     !+
  2     !   RMS $GET - expect only terminators.  NOTE:  This $GET call is not the
  2     !   same as the call in routine COB$$RMS_GET.
  2     !-
  2
  2         WHILE .LEGAL EQL 0 DO
  3             BEGIN                                                  ! Begin Loop
  3
  3             RAB = .COB$$AL_WRITE_RAB [.UNIT[0]] ;
  3             RAB [RAB$W_USZ] = 10 ;
  3             RAB [RAB$L_UBF] = NEXT_CHAR ;
  3             RAB [RAB$V_ETO] = 1 ;
  3             RAB [RAB$L_XAB] = XABTRM ;
  3             WHILE $GET (RAB = .RAB) EQL RMS$_RSA DO $WAIT (RAB = .RAB) ;
```

```
3
3           IF NOT .RAB [RAB$L_STS]
3           THEN
3               !+
3               !  These are special case status that will be handled later.
3               !  (See note below for explanation of missing RMS$_TNS)
3               !-
4               IF (.RAB [RAB$L_STS] NEQ RMS$_BES AND
4               .RAB [RAB$L_STS] NEQ RMS$_EOF AND
4               .RAB [RAB$L_STS] NEQ RMS$_PES AND
4               .RAB [RAB$L_STS] NEQ RMS$_RTB )
3               THEN
3                   LIB$STOP (COB$_ERRDURACC, 1, .RAB + RAB$C_BLN, .RAB [RAB$L_STS],
3                                                                 .RAB [RAB$L_STV] ) ;
3   !+
3   !  NOTE:  No need for call to COB$$PARTIAL_SEQ as buffer of 10 bytes
3   !         is more than sufficient to hold complete escape sequences.
3   !         Most key escape sequences are between 1-4 bytes long.
3   !         Status RMS$_TNS, terminator not seen, would signal a need to
3   !         call routine COB$$PARTIAL_SEQ.
3   !-
3           TERM_PTR = NEXT_CHAR[0] ;
3
3           !+
3           !  Check for legal terminator, then copy it to KEY.
3           !-
3
3               IF .RAB [COB$$B_STV2_LEN] EQL 1                 ! Terminator is one byte
3               THEN
4                   BEGIN
4                   TERM_PTR = RAB [COB$$B_STV0_TERM] ;
4                   SELECTONE .RAB [COB$$B_STV0_TERM] OF
4                       SET
4                           [ CR,                               ! Carriage Return
4                             TAB ] :                           ! Tab
4
5                               BEGIN
5                               CH$MOVE ( 1, .TERM_PTR, .KEY [DSC$A_POINTER] ) ;
5                               LEGAL = 1 ;
4                               END ;
4
4                           [OTHERWISE] :                       ! Error - key not a
4                                                               ! terminator
5                               BEGIN
5                               COB$$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ;
5                               LEGAL = 0 ;
4                               END ;
4                       TES ;
4                   END
4
3               ELSE
3                   IF .RAB [RAB$L_STS] EQL RMS$_EOF
3                   THEN
3                       !+
3                       !  CONTROL Z  - the status RMS$_EOF is returned
3                       !  from the $Get Service.  ^Z is not stored in
3                       !  RAB[RAB$_STV0_TERM].
```

```
3                   !-
3
4                   BEGIN
4                   IF .UNIT [1]  EQL  0
4                   THEN
4                       LIB$STOP ( COB$_EOFON_ACC )         ! Abort
4                   ELSE
4                       COB$$CONTROL_Z ( .UNIT, .KEY ) ;    ! Return to calling
4                       RETURN 0 ;                          ! program.
4                   END
3               ELSE
3                   !+
3                   !  Escape Sequence as Terminator.  COB$$CONTROL_KEY converts
3                   !  terminator sequences to COBOL defined sequences and fills
3                   !  in KEY parameter if terminator is legal.
3                   !-
3
4                   BEGIN
5                   IF NOT ( COB$$CONTROL_KEY (TERM_PTR, .RAB [COB$$B_STV2_LEN],
5                                                                        .KEY) )
4                   THEN
5                       BEGIN                              ! Error, illegal escape
5                       COB$$RMS_PUT_BYTE ( RING_BELL, .FLAGS ) ;    ! sequence.
5                       LEGAL = 0 ;
5                       END
4                   ELSE
4                       LEGAL = 1 ;
3                   END ;
2       END ;                                              ! End Loop
2
2   !+
2   !  VAX COBOL Version 1 / Version 3 interaction.
2   !  Determine if ADVANCING is requested.
2   !  If bit 10 = 0 advancing.  If bit 10 = 1 no advancing.
2   !  Set COB$$AB_PREV[0] - also depending on bit 10, to flag to next COBOL
2   !  statement that advancing/no advancing is required following this
2   !  ACCEPT statement.
2   !-
2
2   IF (.FLAGS AND V_ADV) NEQ 0
2   THEN
2       COB$$AB_PREV[0] = ACC_DNA                          ! Signal- Do Not Advance
2   ELSE
2       !+
2       !  Echo carriage return to screen if advancing is called for.
2       !-
3       BEGIN
3       COB$$RMS_PUT_BYTE ( CARR_RET, .FLAGS ) ;
3       COB$$AB_PREV[0] = ACC_ADV ;                        ! Signal- ADVance
2       END ;
2
2   RETURN 1 ;
1   END ;                                       ! End of routine COB$$FORMAT_FOUR


                                  0126D         .BLKB   3
```

```
FFFFFFFF  FFFFFFFF  FFFFFFFF  FFFFFFFF  FFFFFFFF  FFFFFFFF  01270 P.AAS:  .LONG   -1, -1, -1, -1, -1, -1, -1, -1                    ;
                                        FFFFFFFF  FFFFFFFF  01288                                                                     ;



                                                  01FC 00000 COB$$FORMAT_FOUR:
                                                                         .WORD    Save R2,R3,R4,R5,R6,R7,R8                         ; 4748
                     58 00000000G   00    9E 00002                       MOVAB    COB$$AB_PREV, R8                                  ;
                     57 00000000G   00    9E 00009                       MOVAB    LIB$STOP, R7                                      ;
                     56     FB0C    CF    9E 00010                       MOVAB    COB$$RMS_PUT_BYTE, R6                             ;
                     5E             18    C2 00015                       SUBL2    #24, SP                                           ;
                                    55    D4 00018                       CLRL     LEGAL                                             ; 4797
              04     AE             20    D0 0001A                       MOVL     #32, TMASK                                        ; 4815
              08     AE       BF    AF    9E 0001E                       MOVAB    P.AAS, TMASK+4                                    ; 4816
                     54       08    AC    D0 00023                       MOVL     FLAGS, R4                                         ; 4822
     07              54             04    E1 00027                       BBC      #4, R4, 1$                                        ;
                                    54    DD 0002B                       PUSHL    R4                                                ; 4824
                                    02    DD 0002D                       PUSHL    #2                                                ;
                     66             02    FB 0002F                       CALLS    #2, COB$$RMS_PUT_BYTE                             ;
                     51     5240    8F    3C 00032 1$:                   MOVZWL   #21056, FUNC_VAL                                  ; 4838
                     50 00000000'   EF    9E 00037                       MOVAB    XAB_ITMLST, $$ITMBLKPTR                           ; 4846
                                    80    D4 0003E                       CLRL     ($$ITMBLKPTR)+                                    ;
                     80             51    D0 00040                       MOVL     FUNC_VAL, ($$ITMBLKPTR)+                          ;
                                    80    D4 00043                       CLRL     ($$ITMBLKPTR)+                                    ;
                     80 00030020    8F    D0 00045                       MOVL     #196640, ($$ITMBLKPTR)+                           ;
                     80       08    AE    D0 0004C                       MOVL     TMASK+4, ($$ITMBLKPTR)+                           ;
                                    80    7C 00050                       CLRQ     ($$ITMBLKPTR)+                                    ;
                     53       04    AC    9A 00052                       MOVZBL   UNIT, R3                                          ; 4855
                                    55    D5 00056 2$:                   TSTL     LEGAL                                             ; 4852
                                    03    13 00058                       BEQL     3$                                                ;
                                  00EB    31 0005A                       BRW      14$                                               ;
                     52 00000000G0043     D0 0005D 3$:                   MOVL     COB$$AL_WRITE_RAB[R3], RAB                        ; 4855
              20     A2             0A    B0 00065                       MOVW     #10, 32(RAB)                                      ; 4856
              24     A2       0C    AE    9E 00069                       MOVAB    NEXT_CHAR, 36(RAB)                                ; 4857
              07     A2             10    88 0006E                       BISB2    #16, 7(RAB)                                       ; 4858
              40     A2 00000000'   EF    9E 00072                       MOVAB    XABTRM, 64(RAB)                                   ; 4859
                                    52    DD 0007A 4$:                   PUSHL    RAB                                               ; 4860
        00000000G    00             01    FB 0007C                       CALLS    #1, SYS$GET                                       ;
        000182DA     8F             50    D1 00083                       CMPL     R0, #99034                                        ;
                                    0B    12 0008A                       BNEQ     5$                                                ;
                                    52    DD 0008C                       PUSHL    RAB                                               ;
        00000000G    00             01    FB 0008E                       CALLS    #1, SYS$WAIT                                      ;
                                    E3    11 00095                       BRB      4$                                                ;
                     50       08    A2    D0 00097 5$:                   MOVL     8(RAB), R0                                        ; 4862
                     37             50    E8 0009B                       BLBS     R0, 6$                                            ;
        000181C0     8F             50    D1 0009E                       CMPL     R0, #98752                                        ; 4868
                                    2E    13 000A5                       BEQL     6$                                                ;
        0001827A     8F             50    D1 000A7                       CMPL     R0, #98938                                        ; 4869
                                    25    13 000AE                       BEQL     6$                                                ;
        000181C8     8F             50    D1 000B0                       CMPL     R0, #98760                                        ; 4870
                                    1C    13 000B7                       BEQL     6$                                                ;
        000181A8     8F             50    D1 000B9                       CMPL     R0, #98728                                        ; 4871
                                    13    13 000C0                       BEQL     6$                                                ;
                              0C    A2    DD 000C2                       PUSHL    12(RAB)                                           ; 4874
                                    50    DD 000C5                       PUSHL    R0                                                ; 4873
                              44    A2    9F 000C7                       PUSHAB   68(RAB)                                           ;
                                    01    DD 000CA                       PUSHL    #1                                                ;
```

```
                                           00000000G  8F  DD 000CC           PUSHL   #COB$_ERRDURACC                    ;
                                        67            05  FB 000D2           CALLS   #5, LIB$STOP                       ;
                                        6E        0C  AE  9E 000D5 6$:       MOVAB   NEXT_CHAR, TERM_PTR                ; 4882
                                        01        0E  A2  91 000D9           CMPB    14(RAB), #1                        ; 4888
                                                      1D  12 000DD           BNEQ    8$                                 ;
                                        6E        0C  A2  9E 000DF           MOVAB   12(RAB), TERM_PTR                  ; 4891
                                        50        0C  A2  9A 000E3           MOVZBL  12(RAB), R0                        ; 4892
                                        09            50  91 000E7           CMPB    R0, #9                             ; 4894
                                                      05  13 000EA           BEQL    7$                                 ;
                                        0D            50  91 000EC           CMPB    R0, #13                            ;
                                                      46  12 000EF           BNEQ    11$                                ;
                                        50        0C  AC  D0 000F1 7$:       MOVL    KEY, R0                            ; 4898
                                    04  B0        00  BE  90 000F5           MOVB    @TERM_PTR, @4(R0)                  ;
                                                      46  11 000FA           BRB     12$                                ; 4899
                              0001827A  8F        08  A2  D1 000FC 8$:       CMPL    8(RAB), #98938                     ; 4912
                                                      1D  12 00104           BNEQ    10$                                ;
                                                  05  AC  95 00106           TSTB    UNIT+1                             ; 4921
                                                      0B  12 00109           BNEQ    9$                                 ;
                                           00000000G  8F  DD 0010B           PUSHL   #COB$_EOFON_ACC                    ; 4923
                                        67            01  FB 00111           CALLS   #1, LIB$STOP                       ;
                                                      49  11 00114           BRB     17$                                ;
                                                  0C  AC  DD 00116 9$:       PUSHL   KEY                                ; 4925
                                                  04  AC  DD 00119           PUSHL   UNIT                               ;
                                  0106  C6            02  FB 0011C           CALLS   #2, COB$$CONTROL_Z                 ;
                                                      3C  11 00121           BRB     17$                                ; 4926
                                                  0C  AC  DD 00123 10$:      PUSHL   KEY                                ; 4937
                                        7E        0E  A2  9A 00126           MOVZBL  14(RAB), -(SP)                     ; 4936
                                                  08  AE  9F 0012A           PUSHAB  TERM_PTR                           ;
                             00000000G  00            03  FB 0012D           CALLS   #3, COB$$CONTROL_KEY               ;
                                        0B            50  E8 00134           BLBS    R0, 12$                            ;
                                                      54  DD 00137 11$:      PUSHL   R4                                 ; 4940
                                                      02  DD 00139           PUSHL   #2                                 ;
                                        66            02  FB 0013B           CALLS   #2, COB$$RMS_PUT_BYTE              ;
                                                      55  D4 0013E           CLRL    LEGAL                              ; 4941
                                                      03  11 00140           BRB     13$                                ; 4936
                                        55            01  D0 00142 12$:      MOVL    #1, LEGAL                          ; 4944
                                                    FF0E  31 00145 13$:      BRW     2$                                 ; 4852
                          05            54            0A  E1 00148 14$:      BBC     #10, R4, 15$                       ; 4957
                                        68            05  90 0014C           MOVB    #5, COB$$AB_PREV                   ; 4959
                                                      0A  11 0014F           BRB     16$                                ;
                                                      54  DD 00151 15$:      PUSHL   R4                                 ; 4965
                                                      7E  D4 00153           CLRL    -(SP)                              ;
                                        66            02  FB 00155           CALLS   #2, COB$$RMS_PUT_BYTE              ;
                                        68            04  90 00158           MOVB    #4, COB$$AB_PREV                   ; 4966
                                        50            01  D0 0015B 16$:      MOVL    #1, R0                             ; 4969
                                                          04 0015E           RET                                        ;
                                                      50  D4 0015F 17$:      CLRL    R0                                 ; 4970
                                                          04 00161           RET                                        ;

; Routine Size:  354 bytes,    Routine Base:  _COB$CODE + 1290


; 3470           4971  1
; 3471           4972  1     END                                       ! End of module COB$ACCEPT
; 3472           4973  0     ELUDOM
;                                       PSECT SUMMARY
;
;       Name                        Bytes                       Attributes
;
;  _COB$DATA                           88  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  _COB$CODE                         5106  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(2)




;                                 Library Statistics
;
;                                              -------- Symbols --------        Pages         Processing
;       File                                   Total    Loaded   Percent        Mapped        Time
;
;  _$255$DUA28:[SYSLIB]STARLET.L32;1            9776       153         1           581          00:00.7
;  _$255$DUA28:[COBRTL.OBJ]SMGLIB.L32;1          469        10         2            38          00:00.2




;                                       COMMAND QUALIFIERS
;
;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS$:COBACCEPT/OBJ=OBJ$:COBACCEPT MSRC$:COBACCEPT/UPDATE=(ENH$:COBACCEPT
;       )

; Size:           4745 code + 449 data bytes
; Run Time:          01:09.6
; Elapsed Time:      05:34.6
; Lines/CPU Min:      4290
; Lexemes/CPU-Min: 29151
; Memory Used:  626 pages
; Compilation Complete
```

0061 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY