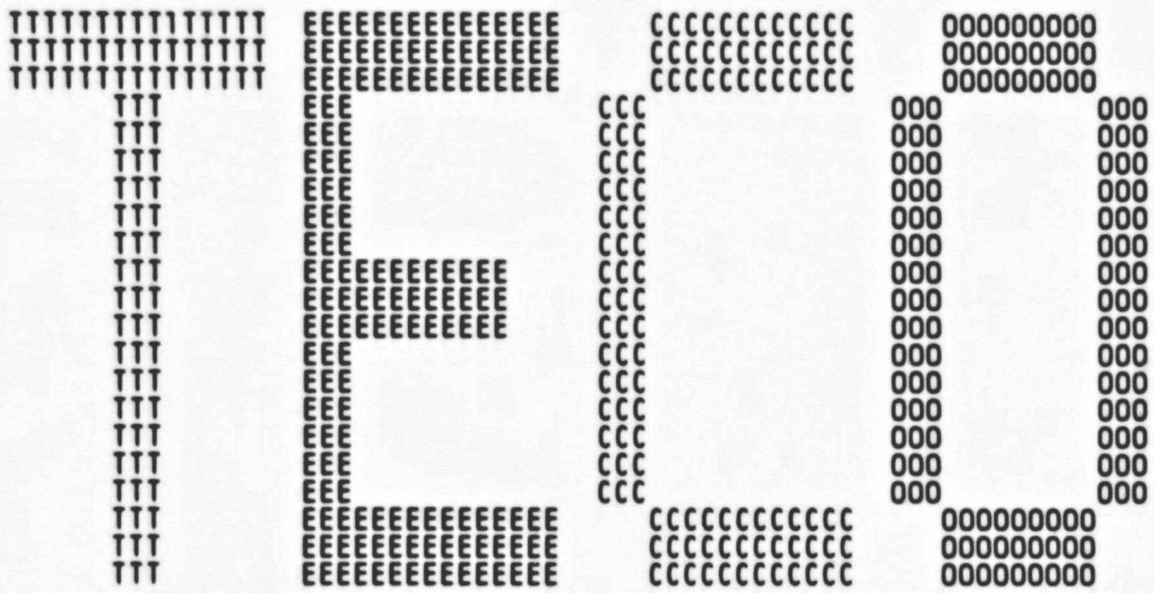

_$25

Symb

TECO
TECO
TECO
TECO
TECO
TECO
TECO
TECO
TECO
TECO
TEMP
TEXI
TFLG
TIME
TLIS
TOPD
TTOB
TTOB
TTOI
TTOM
TTOP
TXST
TYPE
TYPE
TYPE
TYPE
TYPE
UPPE
USRP
USRS
VERS
VTSI
VTSI
WATC
XITM
XITS
ZERO
ZMAX
ZZ

```
TTTTTTTTTT  EEEEEEEEEE    CCCCCCCC    OOOOOO    NN      NN    AAAAAA    TTTTTTTTTT
TTTTTTTTTT  EEEEEEEEEE    CCCCCCCC    OOOOOO    NN      NN    AAAAAA    TTTTTTTTTT
    TT      EE            CC          OO    OO  NN      NN   AA    AA       TT
    TT      EE            CC          OO    OO  NN      NN   AA    AA       TT
    TT      EE            CC          OO    OO  NNNN    NN   AA    AA       TT
    TT      EE            CC          OO    OO  NNNN    NN   AA    AA       TT
    TT      EEEEEEEE      CC          OO    OO  NN  NN  NN   AA    AA       TT
    TT      EEEEEEEE      CC          OO    OO  NN  NN  NN   AA    AA       TT
    TT      EE            CC          OO    OO  NN    NNNN   AAAAAAAAAA     TT
    TT      EE            CC          OO    OO  NN    NNNN   AAAAAAAAAA     TT
    TT      EE            CC          OO    OO  NN      NN   AA    AA       TT      ....
    TT      EE            CC          OO    OO  NN      NN   AA    AA       TT      ....
    TT      EEEEEEEEEE      CCCCCCCC    OOOOOO    NN      NN   AA    AA       TT      ....
    TT      EEEEEEEEEE      CCCCCCCC    OOOOOO    NN      NN   AA    AA       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
```

Table of contents

```
0000     1 .title  TECONAT VAX-11 TECO
0000     2 .ident  /V39.02/
0000     3
0000     4 ;*****************************************************************************
0000     5 ;*                                                                           *
0000     6 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8 ;*  ALL RIGHTS RESERVED.                                                     *
0000     9 ;*                                                                           *
0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    11 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    15 ;*  TRANSFERRED.                                                             *
0000    16 ;*                                                                           *
0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    18 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    19 ;*  CORPORATION.                                                             *
0000    20 ;*                                                                           *
0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23 ;*                                                                           *
0000    24 ;*                                                                           *
0000    25 ;*****************************************************************************
0000    26
0000    27 .show   meb
0000    28 .sbttl  VAX-11 TECO
0000    29
0000    30 ; last edit on 25-Jan-1984 by Mark Bramhall
```

```
0000     32 .sbttl  General macros
0000     33
0000     34 .macro  unorg
0000     35 .endm   unorg
0000     36
0000     37 .macro  org     sect, offset
0000     38 .macro  unorg
0000     39 .noshow meb
0000     40 .psect  sect
0000     41 .show   meb
0000     42 .endm   unorg
0000     43         tmporg  sect, <offset>
0000     44 .endm   org
0000     45
0000     46 .macro  deforg  sect
0000     47         org     sect
0000     48 .if     ne      .-sect
0000     49 .error  ; DEFORG not a beginning of "sect"
0000     50 .endc   ; ne    .-sect
0000     51 .endm   deforg
0000     52
0000     53 .macro  tmporg  sect, offset
0000     54 .noshow meb
0000     55 .psect  sect
0000     56 .       =       .
0000     57 .if     ndf     sect
0000     58 sect:
0000     59 .endc   ; ndf   sect
0000     60 .if     nb      <offset>
0000     61 .       =       offset+sect
0000     62 .endc   ; nb    <offset>
0000     63 .show   meb
0000     64 .endm   tmporg
0000     65
0000     66 .macro  .dsect  start=0
0000     67 .noshow meb
0000     68 $$$$$$  =       start
0000     69 .psect  ..abs.., nopic,usr,ovr,abs,lcl,noshr,noexe,nord,nowrt
0000     70 .if     ndf     ..abs..
0000     71 ..abs..:
0000     72 .endc   ; ndf   ..abs..
0000     73 .       =       $$$$$$+..abs..
0000     74 .show   meb
0000     75 .endm   .dsect
0000     76
0000     77 .macro  .bsect  bit=0
0000     78 .dsect  <1@<bit>>
0000     79 .endm   .bsect
0000     80
0000     81 .macro  .mvsect prefix, suffix, bit=0
0000     82 .macro  .mvdef  sym
0000     83 .if     nb      <sym>
0000     84 prefix'$v_'suffix''sym: .blkb
0000     85 .bsect  prefix'$v_'suffix''sym
0000     86 prefix'$m_'suffix''sym: .blkb
0000     87 .dsect  prefix'$v_'suffix''sym+1
0000     88 .iff    ; nb    <sym>
```

```
0000     89                             .blkb
0000     90 .endc   ; nb     <sym>
0000     91 .endm   .mydef
0000     92 .dsect  <bit>
0000     93 .endm   .mvsect
0000     94
0000     95 .macro  .equate symbol, value
0000     96 .dsect  <value>
0000     97 symbol:
0000     98         unorg
0000     99 .endm   .equate
0000    100
0000    101 .macro  .assume arg1, cond, arg2
0000    102 .if     cond    <arg1>-<arg2>
0000    103 .iff    ; cond  <arg1>-<arg2>
0000    104 .error  ; "arg1 cond arg2" fails
0000    105 .endc   ; cond  <arg1>-<arg2>
0000    106 .endm   .assume
0000    107
0000    108 .macro  .desc   tag, amt=63
0000    109 .noshow meb
0000    110 $$$$$$  =       .+8
0000    111 .align  quad
0000    112 .if     ne      .+8-$$$$$$
0000    113 .warn   ; not quadword aligned
0000    114 .       =       $$$$$$-8
0000    115 .endc   ; ne    .+8-$$$$$$
0000    116 .show   meb
0000    117 .if     nb      <tag>
0000    118 tag'_buf =      $$$$$$
0000    119 tag'_siz =      amt
0000    120          .long  tag'_siz
0000    121          .long  tag'_buf
0000    122 .iff    ; nb    <tag>
0000    123          .long  amt
0000    124          .long  $$$$$$
0000    125 .endc   ; nb    <tag>
0000    126 .assume .       eq      $$$$$$
0000    127            .blkb amt
0000    128 .noshow meb
0000    129 .align  quad
0000    130 .show   meb
0000    131 .endm   .desc
0000    132
0000    133 .macro  bc      bit, dst, ?tag
0000    134 .if     idn     <%extract(0,1,bit)>,<#>
0000    135 .if     le      <1@%extract(1,3000,bit)>-63
0000    136          bicb   s^#1@%extract(1,3000,bit), dst
0000    137 .mexit
0000    138 .endc   ; le    <1@%extract(1,3000,bit)>-63
0000    139 .endc   ; idn   <%extract(0,1,bit)>,<#>
0000    140          bbsc   bit, dst, tag
0000    141 tag:
0000    142 .endm   bc
0000    143
0000    144 .macro  bs      bit, dst, ?tag
0000    145 .if     idn     <%extract(0,1,bit)>,<#>
```

```
0000   146 .if     le      <1@%extract(1,3000,bit)>-63
0000   147          bisb    s^#1@%extract(1,3000,bit), dst
0000   148 .mexit
0000   149 .endc   ; le     <1@%extract(1,3000,bit)>-63
0000   150 .endc   ; idn    <%extract(0,1,bit)>,<#>
0000   151          bbcs    bit, dst, tag
0000   152 tag:
0000   153 .endm   bs
0000   154
0000   155 .macro  getdesc tag, reg, mode=w
0000   156          movab   mode^tag'_buf, reg
0000   157          movl    reg, -(reg)
0000   158 .assume tag+8    eq      tag'_buf
0000   159          movzwl  #tag'_siz, -(reg)
0000   160 .endm   getdesc
```

```
                        .sbttl  System definitions

                        .noshow meb

                                $clidef                         ; cli defs
                                $dscdef                         ; string descriptor defs
                                $dvidef                         ; get device information defs
                                $jpidef                         ; get job/process information defs
                                $libclidef                      ; cli defs
                                $psldef                         ; processor status longword defs
                                $stsdef                         ; status code definitions
                                $ttdef                          ; terminal modes/characteristics defs
                                $tt2def                         ; terminal modes/characteristics defs

                        .show   meb

                        ;+
                        ; Entry to compatibility mode exception handler with:
                        ;
                        ;       -32(r0) =  saved r0             (ctl$al_cmcntx)
                        ;       -28(r0) =  saved r1
                        ;       -24(r0) =  saved r2
                        ;       -20(r0) =  saved r3
                        ;       -16(r0) =  saved r4
                        ;       -12(r0) =  saved r5
                        ;       -08(r0) =  saved r6 (compatibility sp)
                        ;       -04(r0) =  exception code
                        ;        00(r0) =  saved exception pc
                        ;        04(r0) =  saved exception psl (compatibility ps)
                        ;
                        ; Exception codes are:
                        ;
                        ;       0 = reserved instruction
                        ;       1 = bpt
                        ;       2 = iot
                        ;       3 = emt
                        ;       4 = trap
                        ;       5 = illegal instruction
                        ;       6 = odd address
                        ;       7 = t-bit
                        ;-

                        .dsect                                  ; ctl$al_cmcntx data storage offsets

00000004  0000          i_r0:           .blkl                   ; saved r0
00000008  0004          i_r1:           .blkl                   ; saved r1
0000000C  0008          i_r2:           .blkl                   ; saved r2
00000010  000C          i_r3:           .blkl                   ; saved r3
00000014  0010          i_r4:           .blkl                   ; saved r4
00000018  0014          i_r5:           .blkl                   ; saved r5
0000001C  0018          i_sp:           .blkl                   ; saved r6 (compatibility sp)
00000020  001C          i_code:         .blkl                   ; saved exception code
00000024  0020          i_pc:           .blkl                   ; saved pc
00000028  0024          i_ps:           .blkl                   ; saved psl (compatibility ps)
          0028
          0028          .equate i_bias, i_pc-i_r0               ; bias from ctl$al_cmcntx to entry r0
          0020          i_bias:
```

```
0020   219 .sbttl  Internal macros
0020   220
0020   221 .macro  rad50   code
0020   222 $$$$$$  =       0
0020   223 .irpc   char, <code>
0020   224 $$$$$$  =       <$$$$$$*40>+<^a/char/-64>
0020   225 .endr   ; char, <code>
0020   226 .endm   rad50
0020   227
0020   228 .macro  err     code, text
0020   229          bsbw    err
0020   230          tmporg  tecodat
0020   231 $$$$$$  =       .
0020   232          unorg
0020   233            .long $$$$$$
0020   234          tmporg  tecodat
0020   235 rad50    <code>
0020   236             .word  $$$$$$
0020   237             .ascic text
0020   238          unorg
0020   239 .endm   err
```

```
                     0020   241 .sbttl  "ET" (edit typeout) bits
                     0020   242
                     0020   243 .mvsect tec, et$
                     0000   244
                     0000   245 .mvdef  bin                ;    +1., output in binary (image) mode
00000001             0000         tec$v_et$bin:   .blkb
00000002             0001         tec$m_et$bin:   .blkb  .
                     0001   246 .mvdef  crt                ;    +2., do scope type rubout and control/u
00000002             0001         tec$v_et$crt:   .blkb
00000004             0002         tec$m_et$crt:   .blkb  .
                     0002   247 .mvdef  lc                 ;    +4., accept lower case input
00000003             0002         tec$v_et$lc:    .blkb
00000008             0004         tec$m_et$lc:    .blkb  .
                     0003   248 .mvdef  nch                ;    +8., no echo during input for ctrl/t
00000004             0003         tec$v_et$nch:   .blkb
00000010             0008         tec$m_et$nch:   .blkb  .
                     0004   249 .mvdef  cco                ;   +16., cancel control/o on output
00000005             0004         tec$v_et$cco:   .blkb
00000020             0010         tec$m_et$cco:   .blkb  .
                     0005   250 .mvdef  cke                ;   +32., return -1 if error/no input on ctrl/t
00000006             0005         tec$v_et$cke:   .blkb
00000040             0020         tec$m_et$cke:   .blkb  .
                     0006   251 .mvdef  det                ;   +64., detach and detached flag
00000007             0006         tec$v_et$det:   .blkb
00000080             0040         tec$m_et$det:   .blkb  .
                     0007   252 .mvdef  xit                ;  +128., "no prompt yet" flag
00000008             0007         tec$v_et$xit:   .blkb
00000100             0080         tec$m_et$xit:   .blkb  .
                     0008   253 .mvdef  tru                ;  +256., truncate long output lines
00000009             0008         tec$v_et$tru:   .blkb
00000200             0100         tec$m_et$tru:   .blkb  .
                     0009   254 .mvdef  ias                ;  +512., interactive scope available for "watch"
0000000A             0009         tec$v_et$ias:   .blkb
00000400             0200         tec$m_et$ias:   .blkb  .
                     000A   255 .mvdef  rfs                ; +1024., refresh scope available for "watch"
0000000B             000A         tec$v_et$rfs:   .blkb
00000800             0400         tec$m_et$rfs:   .blkb  .
                     000B   256 .mvdef                     ; +2048., reserved by TECO-8
0000000C             000B                         .blkb
                     000C   257 .mvdef  8bt                ; +4096., terminal is an 8-bit terminal
0000000D             000C         tec$v_et$8bt:   .blkb
00002000             1000         tec$m_et$8bt:   .blkb  .
                     000D   258 .mvdef  grv                ; +8192., accept "`" as escape during command input
0000000E             000D         tec$v_et$grv:   .blkb
00004000             2000         tec$m_et$grv:   .blkb  .
                     000E   259 .mvdef                     ;+16384., unused
0000000F             000E                         .blkb
                     000F   260 .mvdef  cc                 ;-32768., allow program to trap control/c
00000010             000F         tec$v_et$cc:    .blkb
00010000             8000         tec$m_et$cc:    .blkb  .
                     0010   261
                     0010   262 .sbttl  "ED" (edit mode) bits
                     0010   263
                     0010   264 .mvsect tec, ed$
                     0000   265
                     0000   266 .mvdef  ctl                ;    +1., don't allow "^" as meaning control character
00000001             0000         tec$v_ed$ctl:   .blkb
```

```
00000002  0001          tec$m_ed$ctl:   .blkb   .
          0001  267 .mvdef  ynk                 ;    +2., allow yanks, etc. to clobber text buffer
00000002  0001          tec$v_ed$ynk:   .blkb
00000004  0002          tec$m_ed$ynk:   .blkb   .
          0002  268 .mvdef  exp                 ;    +4., don't allow arbitrary expansion(s)
00000003  0002          tec$v_ed$exp:   .blkb
00000008  0004          tec$m_ed$exp:   .blkb   .
          0003  269 .mvdef                      ;    +8., reserved by TECO-8
00000004  0003                          .blkb
          0004  270 .mvdef  srh                 ;   +16., don't reset "dot" on search failure
00000005  0004          tec$v_ed$srh:   .blkb
00000020  0010          tec$m_ed$srh:   .blkb   .
          0005  271 .mvdef  imd                 ;   +32., allow immediate mode commands
00000006  0005          tec$v_ed$imd:   .blkb
00000040  0020          tec$m_ed$imd:   .blkb   .
          0006  272 .mvdef  inc                 ;   +64., only move "dot" by one on iterative search failure
00000007  0006          tec$v_ed$inc:   .blkb
00000080  0040          tec$m_ed$inc:   .blkb   .
          0007  273 .mvdef  wch                 ;  +128., don't do automatic "w" command before prompt
00000008  0007          tec$v_ed$wch:   .blkb
00000100  0080          tec$m_ed$wch:   .blkb   .
          0008  274 .mvdef                      ;  +256., unused
00000009  0008                          .blkb
          0009  275 .mvdef                      ;  +512., unused
0000000A  0009                          .blkb
          000A  276 .mvdef                      ; +1024., unused
0000000B  000A                          .blkb
          000B  277 .mvdef                      ; +2048., unused
0000000C  000B                          .blkb
          000C  278 .mvdef                      ; +4096., unused
0000000D  000C                          .blkb
          000D  279 .mvdef                      ; +8192., unused
0000000E  000D                          .blkb
          000E  280 .mvdef                      ;+16384., unused
0000000F  000E                          .blkb
          000F  281 .mvdef                      ;+32768., unused
00000010  000F                          .blkb
```

```
                     0010   283 .sbttl  Internal definitions
                     0010   284
                     0010   285 .dsect  tt$_vt05                        ; define vt05's...
                     0001   286 t$vt05::                                ; ... for compatibility mode
                     0001   287
                     0001   288 .dsect  tt2$m_edit@-16                  ; define 'edit' functions...
                     1000   289 t$edit::                                ; ... for compatibility mode
                     1000   290 .assume t$edit  ne      0
                     1000   291
                     1000   292 .dsect  tt2$m_deccrt@-16                ; define 'dec crt'...
                     2000   293 t$dec::                                 ; ... for compatibility mode
                     2000   294 .assume t$dec   ne      0
                     2000   295
                     2000   296 .dsect  tt2$m_ansicrt@-16               ; define 'ansi crt'...
                     0100   297 t$ansi::                                ; ... for compatibility mode
                     0100   298 .assume t$ansi  ne      0
                     0100   299
                     0100   300 .equate initial_siz,    5000           ; initial text & q-reg size
                     1388       initial_siz:
                     1388   301
                     1388   302 .equate ter_i_siz,       512           ; terminal input buffer size
                     0200       ter_i_siz:
                     0200   303
                     0200   304 .equate ter_o_siz,       512           ; terminal output buffer size
                     0200       ter_o_siz:
                     0200   305
                     0200   306 .equate input_nor_siz,  2048           ; normal input record buffer size
                     0800       input_nor_siz:
                     0800   307
                     0800   308 .equate indir_cmd_siz,  2048           ; "ei" record buffer size
                     0800       indir_cmd_siz:
                     0800   309
                     0800   310 .equate input_alt_siz,  2048           ; alternate input record buffer size
                     0800       input_alt_siz:
                     0800   311
                     0800   312 .equate output_sys_siz,  512           ; sys$output output record buffer size
                     0200       output_sys_siz:
                     0200   313
                     0200   314 .equate input_vfc_siz,    12           ; vfc input buffer size
                     000C       input_vfc_siz:
                     000C   315
                     000C   316 .mvsect fab, tec                       ; definitions for fab$l_tecsts
                     0000   317
                     0000   318 .mvdef  eof                            ; at end-of-file
00000001             0000       fab$v_teceof:   .blkb
00000002             0001       fab$m_teceof:   .blkb   .
                     0001   319 .mvdef  noist                          ; not first time through
00000002             0001       fab$v_tecnoist: .blkb
00000004             0002       fab$m_tecnoist: .blkb   .
                     0002   320 .mvdef  buf                            ; use buffered data instead of file
00000003             0002       fab$v_tecbuf:   .blkb
00000008             0004       fab$m_tecbuf:   .blkb   .
                     0003   321 .mvdef  icr                            ; <cr> ignored, need <cr><lf> on eof
00000004             0003       fab$v_tecicr:   .blkb
00000010             0008       fab$m_tecicr:   .blkb   .
                     0004   322 .mvdef  ecr                            ; extra <cr> output, do <lf> next
00000005             0004       fab$v_tececr:   .blkb
```

```
00000020  0010          fab$m_tececr:   .blkb   .
          0005   323  .mvdef  nxt                    ; pre-fetched character exists (a +3)
00000006  0005          fab$v_tecnxt:   .blkb
00000040  0020          fab$m_tecnxt:   .blkb   .
          0006   324  .mvdef  fmt                    ; /-cr, /cr, or /ft specified
00000007  0006          fab$v_tecfmt:   .blkb
00000080  0040          fab$m_tecfmt:   .blkb   .
          0007   325  .mvdef  rw                     ; /rw - rewind magtape before opens
00000008  0007          fab$v_tecrw:    .blkb
00000100  0080          fab$m_tecrw:    .blkb   .
          0008   326  .mvdef  sh                     ; /sh - shared open
00000009  0008          fab$v_tecsh:    .blkb
00000200  0100          fab$m_tecsh:    .blkb   .
          0009   327  .mvdef  b2                     ; /b2 - basic-plus-2 mode
0000000A  0009          fab$v_tecb2:    .blkb
00000400  0200          fab$m_tecb2:    .blkb   .
          000A   328  .mvdef  nv                     ; /nv - always create a new version
0000000B  000A          fab$v_tecnv:    .blkb
00000800  0400          fab$m_tecnv:    .blkb   .
          000B   329  .mvdef  stm                    ; /stm - stream format specified
0000000C  000B          fab$v_tecstm:   .blkb
00001000  0800          fab$m_tecstm:   .blkb   .
          000C   330  .mvdef  var                    ; /var - variable format specified
0000000D  000C          fab$v_tecvar:   .blkb
00002000  1000          fab$m_tecvar:   .blkb   .
          000D   331  .mvdef                         ; unused...
0000000E  000D                          .blkb
          000E   332  .mvdef                         ; unused...
0000000F  000E                          .blkb
          000F   333  .mvdef                         ; unused...
00000010  000F                          .blkb
          0010   334
          0010   335  .assume .       eq      16
```

```
                        .sbttl  .PSECT definitions

                        .psect  tecodat,        page,nopic,usr,con,rel,lcl,  shr,noexe,rd,nowrt
                        deforg  tecodat
                        .psect  tecodatini,     page,nopic,usr,con,rel,lcl,  shr,noexe,rd,nowrt
                        deforg  tecodatini

                        .psect  tecobuf,        page,nopic,usr,con,rel,lcl,noshr,noexe,rd,  wrt
                        deforg  tecobuf
                        .psect  tecoctl,        page,nopic,usr,con,rel,lcl,noshr,noexe,rd,  wrt
                        deforg  tecoctl
                        .psect  tecoctlini,     page,nopic,usr,con,rel,lcl,noshr,noexe,rd,  wrt
                        deforg  tecoctlini

                        .psect  tecoexe,        page,nopic,usr,con,rel,lcl,  shr,  exe,rd,nowrt
                        deforg  tecoexe
                        .psect  tecoexelbr,     page,nopic,usr,ovr,rel,gbl,  shr,  exe,rd,nowrt
                        deforg  tecoexelbr
                        .psect  tecoexeini,     page,nopic,usr,con,rel,lcl,  shr,  exe,rd,nowrt
                        deforg  tecoexeini
```

```
                        0000   358 .sbttl  Pure data
                        0000   359
                        0000   360          org     tecodat
                        0000   361
                        0000   362 .align  page
                        0000   363
                        0000   364 ter_o_table:                          ; terminal output translate 'til table
                        0000   365 .rept   256
                        0000   366 .if     eq      <<.-ter_o_table>&127>-27
                        0000   367          .byte   27
                        0000   368 .iff    ; eq    <<.-ter_o_table>&127>-27
                        0000   369          .byte   .-ter_o_table
                        0000   370 .endc   ; eq    <<.-ter_o_table>&127>-27
                        0000   371 .endr
                  00    0000             .byte   .-ter_o_table
                  01    0001             .byte   .-ter_o_table
                  02    0002             .byte   .-ter_o_table
                  03    0003             .byte   .-ter_o_table
                  04    0004             .byte   .-ter_o_table
                  05    0005             .byte   .-ter_o_table
                  06    0006             .byte   .-ter_o_table
                  07    0007             .byte   .-ter_o_table
                  08    0008             .byte   .-ter_o_table
                  09    0009             .byte   .-ter_o_table
                  0A    000A             .byte   .-ter_o_table
                  0B    000B             .byte   .-ter_o_table
                  0C    000C             .byte   .-ter_o_table
                  0D    000D             .byte   .-ter_o_table
                  0E    000E             .byte   .-ter_o_table
                  0F    000F             .byte   .-ter_o_table
                  10    0010             .byte   .-ter_o_table
                  11    0011             .byte   .-ter_o_table
                  12    0012             .byte   .-ter_o_table
                  13    0013             .byte   .-ter_o_table
                  14    0014             .byte   .-ter_o_table
                  15    0015             .byte   .-ter_o_table
                  16    0016             .byte   .-ter_o_table
                  17    0017             .byte   .-ter_o_table
                  18    0018             .byte   .-ter_o_table
                  19    0019             .byte   .-ter_o_table
                  1A    001A             .byte   .-ter_o_table
                  1B    001B             .byte   27
                  1C    001C             .byte   .-ter_o_table
                  1D    001D             .byte   .-ter_o_table
                  1E    001E             .byte   .-ter_o_table
                  1F    001F             .byte   .-ter_o_table
                  20    0020             .byte   .-ter_o_table
                  21    0021             .byte   .-ter_o_table
                  22    0022             .byte   .-ter_o_table
                  23    0023             .byte   .-ter_o_table
                  24    0024             .byte   .-ter_o_table
                  25    0025             .byte   .-ter_o_table
                  26    0026             .byte   .-ter_o_table
                  27    0027             .byte   .-ter_o_table
                  28    0028             .byte   .-ter_o_table
                  29    0029             .byte   .-ter_o_table
                  2A    002A             .byte   .-ter_o_table
```

```
                2B  002B            .byte   .-ter_o_table
                2C  002C            .byte   .-ter_o_table
                2D  002D            .byte   .-ter_o_table
                2E  002E            .byte   .-ter_o_table
                2F  002F            .byte   .-ter_o_table
                30  0030            .byte   .-ter_o_table
                31  0031            .byte   .-ter_o_table
                32  0032            .byte   .-ter_o_table
                33  0033            .byte   .-ter_o_table
                34  0034            .byte   .-ter_o_table
                35  0035            .byte   .-ter_o_table
                36  0036            .byte   .-ter_o_table
                37  0037            .byte   .-ter_o_table
                38  0038            .byte   .-ter_o_table
                39  0039            .byte   .-ter_o_table
                3A  003A            .byte   .-ter_o_table
                3B  003B            .byte   .-ter_o_table
                3C  003C            .byte   .-ter_o_table
                3D  003D            .byte   .-ter_o_table
                3E  003E            .byte   .-ter_o_table
                3F  003F            .byte   .-ter_o_table
                40  0040            .byte   .-ter_o_table
                41  0041            .byte   .-ter_o_table
                42  0042            .byte   .-ter_o_table
                43  0043            .byte   .-ter_o_table
                44  0044            .byte   .-ter_o_table
                45  0045            .byte   .-ter_o_table
                46  0046            .byte   .-ter_o_table
                47  0047            .byte   .-ter_o_table
                48  0048            .byte   .-ter_o_table
                49  0049            .byte   .-ter_o_table
                4A  004A            .byte   .-ter_o_table
                4B  004B            .byte   .-ter_o_table
                4C  004C            .byte   .-ter_o_table
                4D  004D            .byte   .-ter_o_table
                4E  004E            .byte   .-ter_o_table
                4F  004F            .byte   .-ter_o_table
                50  0050            .byte   .-ter_o_table
                51  0051            .byte   .-ter_o_table
                52  0052            .byte   .-ter_o_table
                53  0053            .byte   .-ter_o_table
                54  0054            .byte   .-ter_o_table
                55  0055            .byte   .-ter_o_table
                56  0056            .byte   .-ter_o_table
                57  0057            .byte   .-ter_o_table
                58  0058            .byte   .-ter_o_table
                59  0059            .byte   .-ter_o_table
                5A  005A            .byte   .-ter_o_table
                5B  005B            .byte   .-ter_o_table
                5C  005C            .byte   .-ter_o_table
                5D  005D            .byte   .-ter_o_table
                5E  005E            .byte   .-ter_o_table
                5F  005F            .byte   .-ter_o_table
                60  0060            .byte   .-ter_o_table
                61  0061            .byte   .-ter_o_table
                62  0062            .byte   .-ter_o_table
                63  0063            .byte   .-ter_o_table
```

```
64   0064          .byte   .-ter_o_table
65   0065          .byte   .-ter_o_table
66   0066          .byte   .-ter_o_table
67   0067          .byte   .-ter_o_table
68   0068          .byte   .-ter_o_table
69   0069          .byte   .-ter_o_table
6A   006A          .byte   .-ter_o_table
6B   006B          .byte   .-ter_o_table
6C   006C          .byte   .-ter_o_table
6D   006D          .byte   .-ter_o_table
6E   006E          .byte   .-ter_o_table
6F   006F          .byte   .-ter_o_table
70   0070          .byte   .-ter_o_table
71   0071          .byte   .-ter_o_table
72   0072          .byte   .-ter_o_table
73   0073          .byte   .-ter_o_table
74   0074          .byte   .-ter_o_table
75   0075          .byte   .-ter_o_table
76   0076          .byte   .-ter_o_table
77   0077          .byte   .-ter_o_table
78   0078          .byte   .-ter_o_table
79   0079          .byte   .-ter_o_table
7A   007A          .byte   .-ter_o_table
7B   007B          .byte   .-ter_o_table
7C   007C          .byte   .-ter_o_table
7D   007D          .byte   .-ter_o_table
7E   007E          .byte   .-ter_o_table
7F   007F          .byte   .-ter_o_table
80   0080          .byte   .-ter_o_table
81   0081          .byte   .-ter_o_table
82   0082          .byte   .-ter_o_table
83   0083          .byte   .-ter_o_table
84   0084          .byte   .-ter_o_table
85   0085          .byte   .-ter_o_table
86   0086          .byte   .-ter_o_table
87   0087          .byte   .-ter_o_table
88   0088          .byte   .-ter_o_table
89   0089          .byte   .-ter_o_table
8A   008A          .byte   .-ter_o_table
8B   008B          .byte   .-ter_o_table
8C   008C          .byte   .-ter_o_table
8D   008D          .byte   .-ter_o_table
8E   008E          .byte   .-ter_o_table
8F   008F          .byte   .-ter_o_table
90   0090          .byte   .-ter_o_table
91   0091          .byte   .-ter_o_table
92   0092          .byte   .-ter_o_table
93   0093          .byte   .-ter_o_table
94   0094          .byte   .-ter_o_table
95   0095          .byte   .-ter_o_table
96   0096          .byte   .-ter_o_table
97   0097          .byte   .-ter_o_table
98   0098          .byte   .-ter_o_table
99   0099          .byte   .-ter_o_table
9A   009A          .byte   .-ter_o_table
1B   009B          .byte   27
9C   009C          .byte   .-ter_o_table
```

```
009D  009D          .byte   .-ter_o_table
009E  009E          .byte   .-ter_o_table
009F  009F          .byte   .-ter_o_table
00A0  00A0          .byte   .-ter_o_table
00A1  00A1          .byte   .-ter_o_table
00A2  00A2          .byte   .-ter_o_table
00A3  00A3          .byte   .-ter_o_table
00A4  00A4          .byte   .-ter_o_table
00A5  00A5          .byte   .-ter_o_table
00A6  00A6          .byte   .-ter_o_table
00A7  00A7          .byte   .-ter_o_table
00A8  00A8          .byte   .-ter_o_table
00A9  00A9          .byte   .-ter_o_table
00AA  00AA          .byte   .-ter_o_table
00AB  00AB          .byte   .-ter_o_table
00AC  00AC          .byte   .-ter_o_table
00AD  00AD          .byte   .-ter_o_table
00AE  00AE          .byte   .-ter_o_table
00AF  00AF          .byte   .-ter_o_table
00B0  00B0          .byte   .-ter_o_table
00B1  00B1          .byte   .-ter_o_table
00B2  00B2          .byte   .-ter_o_table
00B3  00B3          .byte   .-ter_o_table
00B4  00B4          .byte   .-ter_o_table
00B5  00B5          .byte   .-ter_o_table
00B6  00B6          .byte   .-ter_o_table
00B7  00B7          .byte   .-ter_o_table
00B8  00B8          .byte   .-ter_o_table
00B9  00B9          .byte   .-ter_o_table
00BA  00BA          .byte   .-ter_o_table
00BB  00BB          .byte   .-ter_o_table
00BC  00BC          .byte   .-ter_o_table
00BD  00BD          .byte   .-ter_o_table
00BE  00BE          .byte   .-ter_o_table
00BF  00BF          .byte   .-ter_o_table
00C0  00C0          .byte   .-ter_o_table
00C1  00C1          .byte   .-ter_o_table
00C2  00C2          .byte   .-ter_o_table
00C3  00C3          .byte   .-ter_o_table
00C4  00C4          .byte   .-ter_o_table
00C5  00C5          .byte   .-ter_o_table
00C6  00C6          .byte   .-ter_o_table
00C7  00C7          .byte   .-ter_o_table
00C8  00C8          .byte   .-ter_o_table
00C9  00C9          .byte   .-ter_o_table
00CA  00CA          .byte   .-ter_o_table
00CB  00CB          .byte   .-ter_o_table
00CC  00CC          .byte   .-ter_o_table
00CD  00CD          .byte   .-ter_o_table
00CE  00CE          .byte   .-ter_o_table
00CF  00CF          .byte   .-ter_o_table
00D0  00D0          .byte   .-ter_o_table
00D1  00D1          .byte   .-ter_o_table
00D2  00D2          .byte   .-ter_o_table
00D3  00D3          .byte   .-ter_o_table
00D4  00D4          .byte   .-ter_o_table
00D5  00D5          .byte   .-ter_o_table
```

```
                D6  00D6                .byte   .-ter_o_table
                D7  00D7                .byte   .-ter_o_table
                D8  00D8                .byte   .-ter_o_table
                D9  00D9                .byte   .-ter_o_table
                DA  00DA                .byte   .-ter_o_table
                DB  00DB                .byte   .-ter_o_table
                DC  00DC                .byte   .-ter_o_table
                DD  00DD                .byte   .-ter_o_table
                DE  00DE                .byte   .-ter_o_table
                DF  00DF                .byte   .-ter_o_table
                E0  00E0                .byte   .-ter_o_table
                E1  00E1                .byte   .-ter_o_table
                E2  00E2                .byte   .-ter_o_table
                E3  00E3                .byte   .-ter_o_table
                E4  00E4                .byte   .-ter_o_table
                E5  00E5                .byte   .-ter_o_table
                E6  00E6                .byte   .-ter_o_table
                E7  00E7                .byte   .-ter_o_table
                E8  00E8                .byte   .-ter_o_table
                E9  00E9                .byte   .-ter_o_table
                EA  00EA                .byte   .-ter_o_table
                EB  00EB                .byte   .-ter_o_table
                EC  00EC                .byte   .-ter_o_table
                ED  00ED                .byte   .-ter_o_table
                EE  00EE                .byte   .-ter_o_table
                EF  00EF                .byte   .-ter_o_table
                F0  00F0                .byte   .-ter_o_table
                F1  00F1                .byte   .-ter_o_table
                F2  00F2                .byte   .-ter_o_table
                F3  00F3                .byte   .-ter_o_table
                F4  00F4                .byte   .-ter_o_table
                F5  00F5                .byte   .-ter_o_table
                F6  00F6                .byte   .-ter_o_table
                F7  00F7                .byte   .-ter_o_table
                F8  00F8                .byte   .-ter_o_table
                F9  00F9                .byte   .-ter_o_table
                FA  00FA                .byte   .-ter_o_table
                FB  00FB                .byte   .-ter_o_table
                FC  00FC                .byte   .-ter_o_table
                FD  00FD                .byte   .-ter_o_table
                FE  00FE                .byte   .-ter_o_table
                FF  00FF                .byte   .-ter_o_table
                    0100        372
                    0100        373 file_spec_table:                        ; file spec buffer translate 'til table
                    0100        374 .rept   256
                    0100        375 $$$$$$  =       .-file_spec_table
                    0100        376 .iif    eq      $$$$$$-  0,     $$$$$$ = 0
                    0100        377 .iif    eq      $$$$$$-128,     $$$$$$ = 0
                    0100        378 .iif    eq      $$$$$$-255,     $$$$$$ = 0
                    0100        379 .if     gt      $$$$$$-< 0+128>
                    0100        380 .if     lt      $$$$$$-<32+128>
                    0100        381 $$$$$$  =       $$$$$$&127
                    0100        382 .endc   ; lt    $$$$$$-<32+128>
                    0100        383 .endc   ; gt    $$$$$$-< 0+128>
                    0100        384         .byte   $$$$$$
                    0100        385 .endr
                00  0100                    .byte   $$$$$$
```

```
01  0101                .byte   $$$$$$
02  0102                .byte   $$$$$$
03  0103                .byte   $$$$$$
04  0104                .byte   $$$$$$
05  0105                .byte   $$$$$$
06  0106                .byte   $$$$$$
07  0107                .byte   $$$$$$
08  0108                .byte   $$$$$$
09  0109                .byte   $$$$$$
0A  010A                .byte   $$$$$$
0B  010B                .byte   $$$$$$
0C  010C                .byte   $$$$$$
0D  010D                .byte   $$$$$$
0E  010E                .byte   $$$$$$
0F  010F                .byte   $$$$$$
10  0110                .byte   $$$$$$
11  0111                .byte   $$$$$$
12  0112                .byte   $$$$$$
13  0113                .byte   $$$$$$
14  0114                .byte   $$$$$$
15  0115                .byte   $$$$$$
16  0116                .byte   $$$$$$
17  0117                .byte   $$$$$$
18  0118                .byte   $$$$$$
19  0119                .byte   $$$$$$
1A  011A                .byte   $$$$$$
1B  011B                .byte   $$$$$$
1C  011C                .byte   $$$$$$
1D  011D                .byte   $$$$$$
1E  011E                .byte   $$$$$$
1F  011F                .byte   $$$$$$
20  0120                .byte   $$$$$$
21  0121                .byte   $$$$$$
22  0122                .byte   $$$$$$
23  0123                .byte   $$$$$$
24  0124                .byte   $$$$$$
25  0125                .byte   $$$$$$
26  0126                .byte   $$$$$$
27  0127                .byte   $$$$$$
28  0128                .byte   $$$$$$
29  0129                .byte   $$$$$$
2A  012A                .byte   $$$$$$
2B  012B                .byte   $$$$$$
2C  012C                .byte   $$$$$$
2D  012D                .byte   $$$$$$
2E  012E                .byte   $$$$$$
2F  012F                .byte   $$$$$$
30  0130                .byte   $$$$$$
31  0131                .byte   $$$$$$
32  0132                .byte   $$$$$$
33  0133                .byte   $$$$$$
34  0134                .byte   $$$$$$
35  0135                .byte   $$$$$$
36  0136                .byte   $$$$$$
37  0137                .byte   $$$$$$
38  0138                .byte   $$$$$$
39  0139                .byte   $$$$$$
```

```
3A  013A                .byte   $$$$$$
3B  013B                .byte   $$$$$$
3C  013C                .byte   $$$$$$
3D  013D                .byte   $$$$$$
3E  013E                .byte   $$$$$$
3F  013F                .byte   $$$$$$
40  0140                .byte   $$$$$$
41  0141                .byte   $$$$$$
42  0142                .byte   $$$$$$
43  0143                .byte   $$$$$$
44  0144                .byte   $$$$$$
45  0145                .byte   $$$$$$
46  0146                .byte   $$$$$$
47  0147                .byte   $$$$$$
48  0148                .byte   $$$$$$
49  0149                .byte   $$$$$$
4A  014A                .byte   $$$$$$
4B  014B                .byte   $$$$$$
4C  014C                .byte   $$$$$$
4D  014D                .byte   $$$$$$
4E  014E                .byte   $$$$$$
4F  014F                .byte   $$$$$$
50  0150                .byte   $$$$$$
51  0151                .byte   $$$$$$
52  0152                .byte   $$$$$$
53  0153                .byte   $$$$$$
54  0154                .byte   $$$$$$
55  0155                .byte   $$$$$$
56  0156                .byte   $$$$$$
57  0157                .byte   $$$$$$
58  0158                .byte   $$$$$$
59  0159                .byte   $$$$$$
5A  015A                .byte   $$$$$$
5B  015B                .byte   $$$$$$
5C  015C                .byte   $$$$$$
5D  015D                .byte   $$$$$$
5E  015E                .byte   $$$$$$
5F  015F                .byte   $$$$$$
60  0160                .byte   $$$$$$
61  0161                .byte   $$$$$$
62  0162                .byte   $$$$$$
63  0163                .byte   $$$$$$
64  0164                .byte   $$$$$$
65  0165                .byte   $$$$$$
66  0166                .byte   $$$$$$
67  0167                .byte   $$$$$$
68  0168                .byte   $$$$$$
69  0169                .byte   $$$$$$
6A  016A                .byte   $$$$$$
6B  016B                .byte   $$$$$$
6C  016C                .byte   $$$$$$
6D  016D                .byte   $$$$$$
6E  016E                .byte   $$$$$$
6F  016F                .byte   $$$$$$
70  0170                .byte   $$$$$$
71  0171                .byte   $$$$$$
72  0172                .byte   $$$$$$
```

```
73 0173 .byte $$$$$$
74 0174 .byte $$$$$$
75 0175 .byte $$$$$$
76 0176 .byte $$$$$$
77 0177 .byte $$$$$$
78 0178 .byte $$$$$$
79 0179 .byte $$$$$$
7A 017A .byte $$$$$$
7B 017B .byte $$$$$$
7C 017C .byte $$$$$$
7D 017D .byte $$$$$$
7E 017E .byte $$$$$$
7F 017F .byte $$$$$$
00 0180 .byte $$$$$$
01 0181 .byte $$$$$$
02 0182 .byte $$$$$$
03 0183 .byte $$$$$$
04 0184 .byte $$$$$$
05 0185 .byte $$$$$$
06 0186 .byte $$$$$$
07 0187 .byte $$$$$$
08 0188 .byte $$$$$$
09 0189 .byte $$$$$$
0A 018A .byte $$$$$$
0B 018B .byte $$$$$$
0C 018C .byte $$$$$$
0D 018D .byte $$$$$$
0E 018E .byte $$$$$$
0F 018F .byte $$$$$$
10 0190 .byte $$$$$$
11 0191 .byte $$$$$$
12 0192 .byte $$$$$$
13 0193 .byte $$$$$$
14 0194 .byte $$$$$$
15 0195 .byte $$$$$$
16 0196 .byte $$$$$$
17 0197 .byte $$$$$$
18 0198 .byte $$$$$$
19 0199 .byte $$$$$$
1A 019A .byte $$$$$$
1B 019B .byte $$$$$$
1C 019C .byte $$$$$$
1D 019D .byte $$$$$$
1E 019E .byte $$$$$$
1F 019F .byte $$$$$$
A0 01A0 .byte $$$$$$
A1 01A1 .byte $$$$$$
A2 01A2 .byte $$$$$$
A3 01A3 .byte $$$$$$
A4 01A4 .byte $$$$$$
A5 01A5 .byte $$$$$$
A6 01A6 .byte $$$$$$
A7 01A7 .byte $$$$$$
A8 01A8 .byte $$$$$$
A9 01A9 .byte $$$$$$
AA 01AA .byte $$$$$$
AB 01AB .byte $$$$$$
```

```
AC  01AC        .byte   $$$$$$
AD  01AD        .byte   $$$$$$
AE  01AE        .byte   $$$$$$
AF  01AF        .byte   $$$$$$
B0  01B0        .byte   $$$$$$
B1  01B1        .byte   $$$$$$
B2  01B2        .byte   $$$$$$
B3  01B3        .byte   $$$$$$
B4  01B4        .byte   $$$$$$
B5  01B5        .byte   $$$$$$
B6  01B6        .byte   $$$$$$
B7  01B7        .byte   $$$$$$
B8  01B8        .byte   $$$$$$
B9  01B9        .byte   $$$$$$
BA  01BA        .byte   $$$$$$
BB  01BB        .byte   $$$$$$
BC  01BC        .byte   $$$$$$
BD  01BD        .byte   $$$$$$
BE  01BE        .byte   $$$$$$
BF  01BF        .byte   $$$$$$
C0  01C0        .byte   $$$$$$
C1  01C1        .byte   $$$$$$
C2  01C2        .byte   $$$$$$
C3  01C3        .byte   $$$$$$
C4  01C4        .byte   $$$$$$
C5  01C5        .byte   $$$$$$
C6  01C6        .byte   $$$$$$
C7  01C7        .byte   $$$$$$
C8  01C8        .byte   $$$$$$
C9  01C9        .byte   $$$$$$
CA  01CA        .byte   $$$$$$
CB  01CB        .byte   $$$$$$
CC  01CC        .byte   $$$$$$
CD  01CD        .byte   $$$$$$
CE  01CE        .byte   $$$$$$
CF  01CF        .byte   $$$$$$
D0  01D0        .byte   $$$$$$
D1  01D1        .byte   $$$$$$
D2  01D2        .byte   $$$$$$
D3  01D3        .byte   $$$$$$
D4  01D4        .byte   $$$$$$
D5  01D5        .byte   $$$$$$
D6  01D6        .byte   $$$$$$
D7  01D7        .byte   $$$$$$
D8  01D8        .byte   $$$$$$
D9  01D9        .byte   $$$$$$
DA  01DA        .byte   $$$$$$
DB  01DB        .byte   $$$$$$
DC  01DC        .byte   $$$$$$
DD  01DD        .byte   $$$$$$
DE  01DE        .byte   $$$$$$
DF  01DF        .byte   $$$$$$
E0  01E0        .byte   $$$$$$
E1  01E1        .byte   $$$$$$
E2  01E2        .byte   $$$$$$
E3  01E3        .byte   $$$$$$
E4  01E4        .byte   $$$$$$
```

```
                E5  01E5                .byte   $$$$$$
                E6  01E6                .byte   $$$$$$
                E7  01E7                .byte   $$$$$$
                E8  01E8                .byte   $$$$$$
                E9  01E9                .byte   $$$$$$
                EA  01EA                .byte   $$$$$$
                EB  01EB                .byte   $$$$$$
                EC  01EC                .byte   $$$$$$
                ED  01ED                .byte   $$$$$$
                EE  01EE                .byte   $$$$$$
                EF  01EF                .byte   $$$$$$
                F0  01F0                .byte   $$$$$$
                F1  01F1                .byte   $$$$$$
                F2  01F2                .byte   $$$$$$
                F3  01F3                .byte   $$$$$$
                F4  01F4                .byte   $$$$$$
                F5  01F5                .byte   $$$$$$
                F6  01F6                .byte   $$$$$$
                F7  01F7                .byte   $$$$$$
                F8  01F8                .byte   $$$$$$
                F9  01F9                .byte   $$$$$$
                FA  01FA                .byte   $$$$$$
                FB  01FB                .byte   $$$$$$
                FC  01FC                .byte   $$$$$$
                FD  01FD                .byte   $$$$$$
                FE  01FE                .byte   $$$$$$
                00  01FF                .byte   $$$$$$
                    0200    386
                    0200    387 .align  quad
                    0200    388
                    0200    389 ter_i_nor7_trm:                     ; normal 7-bit terminator mask
00000020'           0200    390         .long   20$-10$
00000208'           0204    391         .long   10$
                    0208    392
FFFFE0FF            0208    393 10$:    .long   ^c<     <1a8>! -        ; all  of   0- 31 except bs,
                    020C    394                 <1a9>! -        ;                         tab,
                    020C    395                 <1a10>! -       ;                         lf,
                    020C    396                 <1a11>! -       ;                         vt,
                    020C    397                 <1a12>>         ;                         ff
00000000            020C    398         .long   0               ; none of  32- 63
00000000            0210    399         .long   0               ; none of  64- 95
80000001            0214    400         .long   <       <1a<96-96>>! -  ; none of  96-127 except accent grave,
                    0218    401                 <1a<127-96>>>   ;                         del
FFFFFFFF            0218    402         .long   ^c<0>           ; all  of 128-159
FFFFFFFF            021C    403         .long   ^c<0>           ; all  of 160-191
FFFFFFFF            0220    404         .long   ^c<0>           ; all  of 192-223
FFFFFFFF            0224    405         .long   ^c<0>           ; all  of 224-255
                    0228    406 20$:
                    0228    407
                    0228    408 ter_i_any_trm:                      ; any character terminator mask
00000020'           0228    409         .long   20$-10$
00000230'           022C    410         .long   10$
                    0230    411
FFFFFFFF            0230    412 10$:    .long   ^c<0>           ; all of   0- 31
FFFFFFFF            0234    413         .long   ^c<0>           ; all of  32- 63
FFFFFFFF            0238    414         .long   ^c<0>           ; all of  64- 95
FFFFFFFF            023C    415         .long   ^c<0>           ; all of  96-127
```

```
                     FFFFFFFF  0240   416          .long   ^c<0>                   ; all of 128-159
                     FFFFFFFF  0244   417          .long   ^c<0>                   ; all of 160-191
                     FFFFFFFF  0248   418          .long   ^c<0>                   ; all of 192-223
                     FFFFFFFF  024C   419          .long   ^c<0>                   ; all of 224-255
                               0250   420 20$:
                               0250   421
                               0250   422 switch_list:                             ; file specification switches
           00020040'00005243   0250   423          .long   ^a/CR/,         fab$m_tecfmt!< fab$m_cr@16>
           00000040 00524320   0258   424          .long   ^a/-CR/,        fab$m_tecfmt!<          0@16>
           00010040'00005446   0260   425          .long   ^a/FT/,         fab$m_tecfmt!<fab$m_ftn@16>
           00010040'004E5446   0268   426          .long   ^a/FTN/,        fab$m_tecfmt!<fab$m_ftn@16>
           00000200 00003242   0270   427          .long   ^a/B2/,         fab$m_tecb2
           00000080 00005752   0278   428          .long   ^a/RW/,         fab$m_tecrw
           00000100 00004853   0280   429          .long   ^a/SH/,         fab$m_tecsh
           00000100 00524853   0288   430          .long   ^a/SHR/,        fab$m_tecsh
           00000400 0000564E   0290   431          .long   ^a/NV/,         fab$m_tecnv
           00000800 004D5453   0298   432          .long   ^a/STM/,        fab$m_tecstm
           00001000 00524156   02A0   433          .long   ^a/VAR/,        fab$m_tecvar
           00000000 00000000   02A8   434          .long   0,              0
                               02B0   435
                               02B0   436 colon_eg_list:                           ; :EG special functions
           004D454D 000002D8'  02B0   437          .long   10$,    ^a/MEM/         ; :EGMEM$ is "memory" logical
           00494E49 000002EA'  02B8   438          .long   20$,    ^a/INI/         ; :EGINI$ is "private init" logical
           00455456 000002FA'  02C0   439          .long   30$,    ^a/VTE/         ; :EGVTE$ is "keypad editor" logical
           0042494C 0000030C'  02C8   440          .long   40$,    ^a/LIB/         ; :EGLIB$ is "macro library" logical
           004D5953 00000000   02D0   441          .long   0,      ^a/SYM/         ; :EGSYM$ is DCL symbol manipulation
                               02D8   442
45 4D 24 43 45 54 000002E0'010E0000'  02D8   443 10$:     .ascid  "TEC$MEMORY"
                     59 52 4F 4D      02E6
4E 49 24 43 45 54 000002F2'010E0000'  02EA   444 20$:     .ascid  "TEC$INIT"
                           54 49      02F8
54 56 24 43 45 54 00000302'010E0000'  02FA   445 30$:     .ascid  "TEC$VTEDIT"
                           54 49 44 45  0308
49 4C 24 43 45 54 00000314'010E0000'  030C   446 40$:     .ascid  "TEC$LIBRARY"
                     59 52 41 52 42     031A
                               031F   447
                               031F   448 .align  byte
                               031F   449
                               031F   450 indir_cmd_dna:                           ; defaults for "ei" files
                     43 45 54 2E  031F   451          .ascii  ".TEC"
                               0323   452 .equate indir_cmd_dns,  .-indir_cmd_dna
                               0004       indir_cmd_dns:
                               0323   453
                               0323   454 quota_msg_desc:                          ; over quota message
63 78 45 25 0A 0D 0000032B'010E0000'  0323   455          .ascid  <13><10>"%Exceeding disk quota"
20 6B 73 69 64 20 67 6E 69 64 65 65  0331
                     61 74 6F 75 71  033D
                               0342   456
                               0342   457 sizing_msg_desc:                         ; memory sizing message
73 65 67 61 70 20 0000034A'010E0000'  0342   458          .ascid  " pages]"
                           5D        0350
                               0351   459
                               0351   460          org     tecodatini
                               0000   461
                               0000   462 .align  long
                               0000   463
                               0000   464 getdvi_itmlst:                          ; $GETDVI item list
```

```
                     0004 0004  0000   465          .word   4, dvi$_devclass
                 00000000 00000000'  0004   466          .long   devclass, 0
                     0006 0004  000C   467          .word   4, dvi$_devtype
                 00000000 00000001'  0010   468          .long   devtype, 0
                     0008 0004  0018   469          .word   4, dvi$_devbufsiz
                 00000000 00000002'  001C   470          .long   devbufsiz, 0
                     000A 0004  0024   471          .word   4, dvi$_devdepend
                 00000000 00000004'  0028   472          .long   devdepend, 0
                     001C 0004  0030   473          .word   4, dvi$_devdepend2
                 00000000 00000008'  0034   474          .long   devdepend2, 0
                     0020 0040  003C   475          .word   64, dvi$_devnam
                 00000000 0000000C'  0040   476          .long   devnam, 0
                     000C 0004  0048   477          .word   4, dvi$_unit
                 00000000 0000004C'  004C   478          .long   unit, 0
                          0000 0000  0054   479          .word   0, 0
                                     0058   480
                                     0058   481 .align  byte
                                     0058   482
                                     0058   483 ter_i_devnam_fna:                    ; terminal input device name string
           54 55 50 4E 49 24 53 59 53  0058   484          .ascii  "SYS$INPUT"
                          00000009   0061   485 ter_i_devnam_fns = .-ter_i_devnam_fna   ; terminal input device name length
                                     0061   486
                                     0061   487 ter_i_devnam:                        ; terminal input device name desc
                 00000058'00000009   0061   488          .long   ter_i_devnam_fns, ter_i_devnam_fna
                                     0069   489
                                     0069   490 ter_o_devnam_fna:                    ; terminal output device name string
        54 55 50 54 55 4F 24 53 59 53  0069   491          .ascii  "SYS$OUTPUT"
                          0000000A   0073   492 ter_o_devnam_fns = .-ter_o_devnam_fna   ; terminal output device name length
                                     0073   493
                                     0073   494 ter_o_devnam:                        ; terminal output device name desc
                 00000069'0000000A   0073   495          .long   ter_o_devnam_fns, ter_o_devnam_fna
                                     007B   496
                                     007B   497 ter_c_devnam:                        ; terminal command device name desc
4F 43 24 53 59 53 00000083'010E0000'  007B   498          .ascid  "SYS$COMMAND"
                       44 4E 41 4D 4D  0089
                                     008E   499
                                     008E   500 ini_dcd_lognam:                     ; logical for private command decoder
           4F 43 45 54 00000096'010E0000'  008E   501          .ascid  "TECO"
                                     009A   502
                                     009A   503 cli_verb_teco:                      ; verb for EDIT/TECO
           4F 43 45 54 000000A2'010E0000'  009A   504          .ascid  "TECO"
                                     00A6   505
                                     00A6   506 cli_no_ini:                         ; qualifier for /NOCOMMAND
49 4E 49 4F 4E 2F 000000AE'010E0000'  00A6   507          .ascid  "/NOINI"
                                     00B4   508
                                     00B4   509 cli_no_create:                      ; qualifier for /NOCREATE
45 52 43 4F 4E 2F 000000BC'010E0000'  00B4   510          .ascid  "/NOCREATE"
                             45 54 41  00C2
                                     00C5   511
                                     00C5   512 cli_no_memory:                      ; qualifier for /NOMEMORY
4D 45 4D 4F 4E 2F 000000CD'010E0000'  00C5   513          .ascid  "/NOMEMORY"
                             59 52 4F  00D3
                                     00D6   514
                                     00D6   515 cli_inspect:                        ; qualifier for /READ_ONLY
45 50 53 4E 49 2F 000000DE'010E0000'  00D6   516          .ascid  "/INSPECT"
                                54 43  00E4
                                     00E6   517
```

```
                                00E6   518 cli_null:                       ; a null string for string building
                 000000EE'010E0000' 00E6   519         .ascid  ""
                                00EE   520
                                00EE   521 cli_space:                      ; a space for string building
              20 000000F6'010E0000' 00EE   522         .ascid  " "
                                00F7   523
                                00F7   524 cli_equals:                     ; an equals sign for string building
              3D 000000FF'010E0000' 00F7   525         .ascid  "="
                                0100   526
                                0100   527 cli_dollar:                     ; a dollar sign for string building
              24 00000108'010E0000' 0100   528         .ascid  "$"
                                0109   529
                                0109   530 cli_init:                       ; logical name for command file
4E 49 24 43 45 54 00000111'010E0000' 0109   531         .ascid  "TEC$INIT"
                             54 49    0117
                                0119   532
                                0119   533 cli_qual_command:               ; to fetch /COMMAND qualifier
4E 41 4D 4D 4F 43 00000121'010E0000' 0119   534         .ascid  "COMMAND"
                                44     0127
                                0128   535
                                0128   536 cli_qual_create:                ; to fetch /CREATE qualifier
45 54 41 45 52 43 00000130'010E0000' 0128   537         .ascid  "CREATE"
                                0136   538
                                0136   539 cli_qual_memory:                ; to fetch /MEMORY qualifier
59 52 4F 4D 45 4D 0000013E'010E0000' 0136   540         .ascid  "MEMORY"
                                0144   541
                                0144   542 cli_qual_execute:               ; to fetch /EXECUTE qualifier
54 55 43 45 58 45 0000014C'010E0000' 0144   543         .ascid  "EXECUTE"
                             45     0152
                                0153   544
                                0153   545 cli_parm_p1:                    ; to fetch P1 parameter
           31 50 0000015B'010E0000' 0153   546         .ascid  "P1"
                                015D   547
                                015D   548 cli_qual_output:                ; to fetch /OUTPUT qualifier
54 55 50 54 55 4F 00000165'010E0000' 015D   549         .ascid  "OUTPUT"
                                016B   550
                                016B   551 cli_qual_read_only:             ; to fetch /READ_ONLY qualifier
4F 5F 44 41 45 52 00000173'010E0000' 016B   552         .ascid  "READ_ONLY"
                             59 4C 4E  0179
```

```
                        017C    554 .sbttl  Impure data
                        017C    555
                        017C    556         org     tecoctl
                        0000    557
                        0000    558 .align  quad
                        0000    559
                        0000    560 ter_o_status1:                          ; terminal output i/o status block #1
00000000 00000001       0000    561         .quad   1
                        0008    562
                        0008    563 ter_o_status2:                          ; terminal output i/o status block #2
00000000 00000001       0008    564         .quad   1
                        0010    565
                        0010    566 ter_o_pos:                              ; terminal output position IOSB
00000000 00000000       0010    567         .quad   0
                        0018    568
                        0018    569 ter_i_status:                           ; terminal input i/o status block
00000000 00000000       0018    570         .quad   0
                        0020    571
                        0020    572 ter_i:                                  ; terminal input buffer descriptor
            00000000    0020    573         .long   0                       ;  no initial size of input
            00000400'   0024    574         .long   ter_i_buf               ;  but pointer to input buffer is set
                        0028    575
                        0028    576 ter_i_nor8_trm:                         ; normal 8-bit terminator mask
            00000020'   0028    577         .long   20$-10$
            00000030'   002C    578         .long   10$
                        0030    579
            FFFFE0FF    0030    580 10$:    .long   ^c<     <1@8>! -        ; all  of   0- 31 except bs,
                        0034    581                     <1@9>! -        ;                         tab,
                        0034    582                     <1@10>! -       ;                         lf,
                        0034    583                     <1@11>! -       ;                         vt,
                        0034    584                     <1@12>>         ;                         ff
            00000000    0034    585         .long   0                       ; none of  32- 63
            00000000    0038    586         .long   0                       ; none of  64- 95
            80000001    003C    587         .long   <       <1@<96-96>>! -  ; none of  96-127 except accent grave,
                        0040    588                     <1@<127-96>>>   ;                         del
            FFFFFFFF    0040    589         .long   ^c<0>                   ; all  of 128-159
            FFFFFFFF    0044    590         .long   ^c<0>                   ; all  of 160-191 (initially...)
            FFFFFFFF    0048    591         .long   ^c<0>                   ; all  of 192-223 (initially...)
            FFFFFFFF    004C    592         .long   ^c<0>                   ; all  of 224-255 (initially...)
                        0050    593 20$:
                        0050    594
                        0050    595 tmp_string:                             ; a temporary string...
                        0050    596         .desc   tmp_string
            0000003F    0050                .long   tmp_string_siz
            00000058'   0054                .long   tmp_string_buf
            00000097    0058                .blkb 63
                        0098    597
                        0098    598 tmp_string2:                            ; another temporary string...
                        0098    599         .desc   tmp_string2
            0000003F    0098                .long   tmp_string2_siz
            000000A0'   009C                .long   tmp_string2_buf
            000000DF    00A0                .blkb 63
                        00E0    600
                        00E0    601 .align  long
                        00E0    602
                        00E0    603 ter_i_nor_trm_ptr:                      ; normal input terminator mask pointer
            00000200'   00E0    604         .long   ter_i_nor7_trm          ;  (pre-set) normal 7-bit input
```

```
                        00E4   605
                        00E4   606 saved_sp:                                ; saved sp value for error exits
000000E8                00E4   607         .blkl
                        00E8   608
                        00E8   609 still_free:                              ; amount of memory still free
00000000                00E8   610         .long   0
                        00EC   611
                        00EC   612 ctrlz_cnt:                               ; count of consecutive control/z's
00000000                00EC   613         .long   0
                        00F0   614
                        00F0   615 ter_o_cc:                                ; terminal output carriage control
0000                    00F0   616         .word   0
                        00F2   617 output_sys_vfc:                          ; sys$output print control
0001                    00F2   618         .word   1
                        00F4   619
                        00F4   620 .align  word
                        00F4   621
                        00F4   622 ter_i_chan:                              ; channel # for terminal input
0000                    00F4   623         .word   0
                        00F6   624
                        00F6   625 ter_o_chan:                              ; channel # for terminal output
0000                    00F6   626         .word   0
                        00F8   627
                        00F8   628 ter_c_chan:                              ; channel # for terminal control/c ast
0000                    00F8   629         .word   0
                        00FA   630
                        00FA   631 ter_o_unit:                              ; terminal output device unit
0000                    00FA   632         .word   0
                        00FC   633
                        00FC   634 .align  byte
                        00FC   635
                        00FC   636 ctrlc_flag:                              ; exit on second control/c flip/flop
00                      00FC   637         .byte   0
                        00FD   638
                        00FD   639 ctrlo_flag:                              ; control/o in effect flag
00                      00FD   640         .byte   0
                        00FE   641
                        00FE   642 ter_o_force:                             ; terminal output being forced flag
00                      00FE   643         .byte   0
                        00FF   644
                        00FF   645 ter_o_pend:                              ; terminal output pending count
FF                      00FF   646         .byte   -1
                        0100   647
                        0100   648 exiting_flag:                            ; exiting flag
01                      0100   649         .byte   1                       ;  preset to force CTRL/T disable
                        0101   650
                        0101   651 .align  long
                        0104   652
                        0104   653 err_msgvec:                              ; error message vector
00000110                0104   654         .blkl   3
                        0110   655
                        0110   656 .noshow meb
                        0110   657
                        0110   658 input_nor_fab:                           ; fab for normal input
                        0110   659         $fab -                          ; allocate a fab
                        0110   660                 fac=get, -              ;  allow gets
                        0110   661                 fna=file_spec_buf, -    ;  file name will come from here
```

```
                     0110   662                 nam=input_nor_nam, -    ;  catch the resultant filespec
                     0110   663                 rat=cr, -               ;  (pre-set) attributes of implied cc
                     0110   664                 rfm=var, -              ;  (pre-set) record format of variable
                     0110   665                 xab=input_nor_xab       ;  catch the protection code
                     0160   666
                     0160   667 .show   meb
                     0160   668
                     0160   669 .equate fab$l_tecsts,   .-input_nor_fab ; our private fab status longword
                     0050       fab$l_tecsts:
00000164             0160   670         .blkl                           ;  private status longword
                     0164   671 .equate fab$l_tecrab,   .-input_nor_fab ; our private fab to rab pointer
                     0054       fab$l_tecrab:
00000230'            0164   672         .long   input_nor_rab           ;  private pointer to the correct rab
                     0168   673 .equate fab$q_tecque,   .-input_nor_fab ; our private fab data line queue
                     0058       fab$q_tecque:
00000170             0168   674         .blkq                           ;  private data line queue
                     0170   675 .equate fab$l_tecdsp,   .-input_nor_fab ; our private fab dispatch longword
                     0060       fab$l_tecdsp:
00000174             0170   676         .blkl                           ;  private dispatch longword
                     0174   677 .equate fab$l_tecctl,   .-input_nor_fab ; our private fab control bytes
                     0064       fab$l_tecctl:
00000178             0174   678         .blkl                           ;  private control bytes
                     0178   679
                     0178   680 .noshow meb
                     0178   681
                     0178   682 input_nor_nam:                          ; nam for normal input
                     0178   683         $nam -                          ; allocate a nam
                     0178   684                 rsa=input_nor_spec, -   ;  resultant filespec goes here
                     0178   685                 rss=nam$c_maxrss        ;  allowing the maximum length
                     01D8   686
                     01D8   687 input_nor_xab:                          ; xab for normal input
                     01D8   688         $xabpro                         ; allocate a protection code xab
                     0230   689
                     0230   690 input_nor_rab:                          ; rab for normal input
                     0230   691         $rab -                          ; allocate a rab
                     0230   692                 fab=input_nor_fab, -    ;  for normal input fab
                     0230   693                 rhb=input_nor_vfc, -    ;  use this vfc buffer
                     0230   694                 rop=<loc,rah>, -        ;  use locate mode & read ahead
                     0230   695                 ubf=input_nor_buf, -    ;  use this record buffer
                     0230   696                 usz=input_nor_siz       ;  with this size
                     0274   697
                     0274   698 output_nor_fab:                         ; fab for normal output
                     0274   699         $fab -                          ; allocate a fab
                     0274   700                 fac=<put,trn>, -        ;  allow puts & truncates
                     0274   701                 fna=file_spec_buf, -    ;  file name will come from here
                     0274   702                 nam=output_nor_nam, -   ;  catch the resultant filespec
                     0274   703                 org=seq                 ;  organization is sequential
                     02C4   704 .assume fab$l_tecsts eq .-output_nor_fab
000002C8             02C4   705         .blkl                           ;  private status longword
                     02C8   706 .assume fab$l_tecrab eq .-output_nor_fab
00000334'            02C8   707         .long   output_nor_rab          ;  private pointer to the correct rab
                     02CC   708 .assume fab$q_tecque eq .-output_nor_fab
000002D4             02CC   709         .blkq                           ;  private data line queue
                     02D4   710
                     02D4   711 output_nor_nam:                         ; nam for normal output
                     02D4   712         $nam -                          ; allocate a nam
                     02D4   713                 rsa=output_nor_spec, -  ;  resultant filespec goes here
```

```
                     02D4   714                    rss=nam$c_maxrss        ;  allowing the maximum length
                     0334   715
                     0334   716 output_nor_rab:                            ; rab for normal output
                     0334   717         $rab -                            ; allocate a rab
                     0334   718                    fab=output_nor_fab, -   ;  for normal output fab
                     0334   719                    rop=<tpt,wbh>           ;  use truncate on put & write behind
                     0378   720
                     0378   721 indir_cmd_fab:                            ; fab for "ei"
                     0378   722         $fab -                            ; allocate a fab
                     0378   723                    dna=indir_cmd_dna, -    ;  default string is '.tec'
                     0378   724                    dns=indir_cmd_dns, -    ;  which is this long
                     0378   725                    fac=get, -              ;  allow gets
                     0378   726                    fna=file_spec_buf, -    ;  file name will come from here
                     0378   727                    fns=ini_dcd_fns, -      ;  (pre-set) for "SYS$LOGIN:TECO"
                     0378   728                    fop=sqo, -              ;  (pre-set) sequential only option
                     0378   729                    nam=indir_cmd_nam, -    ;  catch the resultant filespec
                     0378   730                    rat=cr, -               ;  (pre-set) attributes of implied cc
                     0378   731                    rfm=var, -              ;  (pre-set) record format of variable
                     0378   732                    shr=get                 ;  (pre-set) sharing is other gets
                     03C8   733 .assume fab$l_tecsts eq .-indir_cmd_fab
            00000000 03C8   734         .long   0                         ;  (pre-set) private status longword
                     03CC   735 .assume fab$l_tecrab eq .-indir_cmd_fab
            00000440' 03CC   736         .long   indir_cmd_rab             ;  private pointer to the correct rab
                     03D0   737 .assume fab$q_tecque eq .-indir_cmd_fab
000003D0'000003D0'   03D0   738 10$:     .long   10$, 10$                  ;  (pre-set) private data line queue
                     03D8   739 .assume fab$l_tecdsp eq .-indir_cmd_fab
            00000B91' 03D8   740         .long   getbyt_first              ;  (pre-set) private dispatch longword
                     03DC   741 .assume fab$l_tecctl eq .-indir_cmd_fab
         00 00 00 0A 03DC   742         .byte   10, 0, 0, 0               ;  (pre-set) private control bytes
                     03E0   743
                     03E0   744 indir_cmd_nam:                            ; nam for "ei"
                     03E0   745         $nam -                            ; allocate a nam
                     03E0   746                    rsa=indir_cmd_spec, -   ;  resultant filespec goes here
                     03E0   747                    rss=nam$c_maxrss        ;  allowing the maximum length
                     0440   748
                     0440   749 indir_cmd_rab:                            ; rab for "ei"
                     0440   750         $rab -                            ; allocate a rab
                     0440   751                    fab=indir_cmd_fab, -    ;  for "ei" fab
                     0440   752                    rhb=indir_cmd_vfc, -    ;  use this vfc buffer
                     0440   753                    rop=<loc,rah>, -        ;  use locate mode & read ahead
                     0440   754                    ubf=indir_cmd_buf, -    ;  use this record buffer
                     0440   755                    usz=indir_cmd_siz       ;  with this size
                     0484   756
                     0484   757 input_alt_fab:                            ; fab for alternate input
                     0484   758         $fab -                            ; allocate a fab
                     0484   759                    fac=get, -              ;  allow gets
                     0484   760                    fna=file_spec_buf, -    ;  file name will come from here
                     0484   761                    nam=input_alt_nam, -    ;  catch the resultant filespec
                     0484   762                    rat=cr, -               ;  (pre-set) attributes of implied cc
                     0484   763                    rfm=var, -              ;  (pre-set) record format of variable
                     0484   764                    xab=input_alt_xab       ;  catch the protection code
                     04D4   765 .assume fab$l_tecsts eq .-input_alt_fab
            000004D8 04D4   766         .blkl                             ;  private status longword
                     04D8   767 .assume fab$l_tecrab eq .-input_alt_fab
            000005A4' 04D8   768         .long   input_alt_rab             ;  private pointer to the correct rab
                     04DC   769 .assume fab$q_tecque eq .-input_alt_fab
            000004E4 04DC   770         .blkq                             ;  private data line queue
```

```
                    04E4   771 .assume fab$l_tecdsp eq .-input_alt_fab
000004E8            04E4   772         .blkl                           ;  private dispatch longword
                    04E8   773 .assume fab$l_tecctl eq .-input_alt_fab
000004EC            04E8   774         .blkl                           ;  private control bytes
                    04EC   775
                    04EC   776 input_alt_nam:                          ; nam for alternate input
                    04EC   777         $nam -                          ; allocate a nam
                    04EC   778                 rsa=input_alt_spec, -   ;  resultant filespec goes here
                    04EC   779                 rss=nam$c_maxrss        ;  allowing the maximum length
                    054C   780
                    054C   781 input_alt_xab:                          ; xab for alternate input
                    054C   782         $xabpro                         ; allocate a protection code xab
                    05A4   783
                    05A4   784 input_alt_rab:                          ; rab for alternate input
                    05A4   785         $rab -                          ; allocate a rab
                    05A4   786                 fab=input_alt_fab, -    ;  for alternate input fab
                    05A4   787                 rhb=input_alt_vfc, -    ;  use this vfc buffer
                    05A4   788                 rop=<loc,rah>, -        ;  use locate mode & read ahead
                    05A4   789                 ubf=input_alt_buf, -    ;  use this record buffer
                    05A4   790                 usz=input_alt_siz       ;  with this size
                    05E8   791
                    05E8   792 output_alt_fab:                         ; fab for alternate output
                    05E8   793         $fab -                          ; allocate a fab
                    05E8   794                 fac=<put,trn>, -        ;  allow puts & truncates
                    05E8   795                 fna=file_spec_buf, -    ;  file name will come from here
                    05E8   796                 nam=output_alt_nam, -   ;  catch the resultant filespec
                    05E8   797                 org=seq                 ;  organization is sequential
                    0638   798 .assume fab$l_tecsts eq .-output_alt_fab
0000063C            0638   799         .blkl                           ;  private status longword
                    063C   800 .assume fab$l_tecrab eq .-output_alt_fab
000006A8'           063C   801         .long   output_alt_rab          ;  private pointer to the correct rab
                    0640   802 .assume fab$q_tecque eq .-output_alt_fab
00000648            0640   803         .blkq                           ;  private data line queue
                    0648   804
                    0648   805 output_alt_nam:                         ; nam for alternate output
                    0648   806         $nam -                          ; allocate a nam
                    0648   807                 rsa=output_alt_spec, -  ;  resultant filespec goes here
                    0648   808                 rss=nam$c_maxrss        ;  allowing the maximum length
                    06A8   809
                    06A8   810 output_alt_rab:                         ; rab for alternate output
                    06A8   811         $rab -                          ; allocate a rab
                    06A8   812                 fab=output_alt_fab, -   ;  for alternate output fab
                    06A8   813                 rop=<tpt,wbh>           ;  use truncate on put & write behind
                    06EC   814
                    06EC   815 en_fab:                                 ; for for "en"
                    06EC   816         $fab -                          ; allocate a fab
                    06EC   817                 fna=file_spec_buf, -    ;  file name will come from here
                    06EC   818                 nam=en_nam              ;  "en" name block
                    073C   819 .assume fab$l_tecsts eq .-en_fab
00000001            073C   820         .long   fab$m_teceof            ;  private status longword (fnf)
                    0740   821
                    0740   822 en_nam:                                 ; nam for "en"
                    0740   823         $nam -                          ; allocate a nam
                    0740   824                 esa=en_spec, -          ;  "en" parse filespec
                    0740   825                 ess=nam$c_maxrss, -     ;  allowing the maximum length
                    0740   826                 rsa=en_occur, -         ;  "en" occurance filespec
                    0740   827                 rss=nam$c_maxrss        ;  allowing the maximum length
```

```
                        07A0   828
                        07A0   829 input_sys_fab:                          ; fab for sys$input input
                        07A0   830         $fab -                          ; allocate a fab
                        07A0   831                 fac=get, -              ;  allow gets
                        07A0   832                 fna=ter_i_devnam_fna, - ;  file name as for terminal input
                        07A0   833                 fns=ter_i_devnam_fns, - ;   with the correct length
                        07A0   834                 rat=cr, -               ;  (pre-set) attributes of implied cc
                        07A0   835                 rfm=var                 ;  (pre-set) record format of variable
                        07F0   836 .assume fab$l_tecsts eq .-input_sys_fab
                00000000 07F0   837         .long   0                       ;  (pre-set) private status longword
                        07F4   838 .assume fab$l_tecrab eq .-input_sys_fab
                00000808' 07F4  839         .long   input_sys_rab           ;  private pointer to the correct rab
                        07F8   840 .assume fab$q_tecque eq .-input_sys_fab
        000007F8'000007F8' 07F8 841 10$:    .long   10$, 10$                ;  (pre-set) private data line queue
                        0800   842 .assume fab$l_tecdsp eq .-input_sys_fab
                00000B91' 0800  843         .long   getbyt_first            ;  (pre-set) private dispatch longword
                        0804   844 .assume fab$l_tecctl eq .-input_sys_fab
             00 00 00 0A 0804   845         .byte   10, 0, 0, 0             ;  (pre-set) private control bytes
                        0808   846
                        0808   847 input_sys_rab:                          ; rab for sys$input input
                        0808   848         $rab -                          ; allocate a rab
                        0808   849                 fab=input_sys_fab, -    ;  for sys$input input fab
                        0808   850                 rhb=input_sys_vfc, -    ;  use this vfc buffer
                        0808   851                 rop=<loc,rah>, -        ;  use locate mode & read ahead
                        0808   852                 ubf=ter_i_buf, -        ;  use this record buffer
                        0808   853                 usz=ter_i_siz           ;  with this size
                        084C   854
                        084C   855 output_sys_fab:                         ; fab for sys$output output
                        084C   856         $fab -                          ; allocate a fab
                        084C   857                 fac=put, -              ;  allow puts
                        084C   858                 fna=ter_o_devnam_fna, - ;  file name as for terminal output
                        084C   859                 fns=ter_o_devnam_fns, - ;   with the correct length
                        084C   860                 fsz=2, -                ;  fixed control area is 2 bytes
                        084C   861                 org=seq, -              ;  organization is sequential
                        084C   862                 rat=prn, -              ;  use print file format
                        084C   863                 rfm=vfc                 ;  format is variable w/ fixed control
                        089C   864
                        089C   865 output_sys_rab:                         ; rab for sys$output output
                        089C   866         $rab -                          ; allocate a rab
                        089C   867                 fab=output_sys_fab, -   ;  for sys$output output fab
                        089C   868                 rbf=output_sys_buf, -   ;  use this output buffer
                        089C   869                 rhb=output_sys_vfc, -   ;  use this print control buffer
                        089C   870                 rsz=0, -                ;  no partial record initially
                        089C   871                 rop=wbh                 ;  use write behind
                        08E0   872
                        08E0   873 .show   meb
                        08E0   874
                        08E0   875 file_spec_opt:                          ; file specification options
                000008E4 08E0  876         .blkl
                        08E4   877
                        08E4   878 file_spec_swt:                          ; file specification switch
                000008E8 08E4  879         .blkl
                        08E8   880
                        08E8   881 ter_oob_msk:                            ; terminal out-of-band re-enable mask
                00000000 08E8  882         .long   0                       ;  preset to nothing to re-enable
                        08EC   883
                        08EC   884 .align  byte
```

```
                        08EC    885
                        08EC    886 file_spec_len:                      ; file specification length
000008ED                08EC    887         .blkb
                        08ED    888
                        08ED    889 file_spec_buf:                      ; file specification buffer
45 54 3A 4E 49 47 4F 4C 24 53 59 53 08ED    890         .ascii  "SYS$LOGIN:TECO"        ;  (pre-set for :EISYS$LOGIN:TECO$)
                              4F 43 08F9
0000000E                08FB    891 ini_dcd_fns = .-file_spec_buf
000009ED                08FB    892         .blkb   nam$c_maxrss+1-<.-file_spec_buf>
                        09ED    893
                        09ED    894 input_nor_spec:                     ; resultant filespec for normal input
00000AEC                09ED    895         .blkb   nam$c_maxrss
                        0AEC    896
                        0AEC    897 output_nor_spec:                    ; resultant filespec for normal output
00000BEB                0AEC    898         .blkb   nam$c_maxrss
                        0BEB    899
                        0BEB    900 indir_cmd_spec:                     ; resultant filespec for "ei"
00000CEA                0BEB    901         .blkb   nam$c_maxrss
                        0CEA    902
                        0CEA    903 input_alt_spec:                     ; resultant spec for alternate input
00000DE9                0CEA    904         .blkb   nam$c_maxrss
                        0DE9    905
                        0DE9    906 output_alt_spec:                    ; resultant spec for alternate output
00000EE8                0DE9    907         .blkb   nam$c_maxrss
                        0EE8    908
                        0EE8    909 en_spec:                            ; "en" parse filespec
00000FE7                0EE8    910         .blkb   nam$c_maxrss
                        0FE7    911
                        0FE7    912 en_occur:                           ; "en" occurance filespec
000010E6                0FE7    913         .blkb   nam$c_maxrss
                        10E6    914
                        10E6    915 input_nor_vfc:                      ; normal input vfc buffer
000010F2                10E6    916         .blkb   input_vfc_siz
                        10F2    917
                        10F2    918 indir_cmd_vfc:                      ; "ei" vfc buffer
000010FE                10F2    919         .blkb   input_vfc_siz
                        10FE    920
                        10FE    921 input_alt_vfc:                      ; alternate input vfc buffer
0000110A                10FE    922         .blkb   input_vfc_siz
                        110A    923
                        110A    924 input_sys_vfc:                      ; sys$input vfc buffer
00001116                110A    925         .blkb   input_vfc_siz
                        1116    926
                        1116    927         org     tecoctlini
                        0000    928
                        0000    929 .align  long
                        0000    930
                        0000    931 getdvi_info:                        ; $GETDVI returned information
                        0000    932 devclass:                           ;  device class
00000001                0000    933         .blkb
                        0001    934 devtype:                            ;  device type
00000002                0001    935         .blkb
                        0002    936 devbufsiz:                          ;  device buffer size (width)
00000004                0002    937         .blkw
                        0004    938 devdepend:                          ;  device dependent bits
00000008                0004    939         .blkl
                        0008    940 devdepend2:                         ;  device dependent bits #2
```

```
0000000C  0008   941                .blkl
0000000C  000C   942 getdvi_info_len = .-getdvi_info
          000C   943
          000C   944 devnam:                                          ;  device name string
0000004C  000C   945                .blkb   64
          004C   946
          004C   947 unit:                                           ;  device unit number
00000050  004C   948                .blkl
          0050   949
          0050   950 .noshow meb
          0050   951
          0050   952 cli_req_getcmd:                                 ; request block to get command line
          0050   953                $clireqdesc -                    ; define cli request descriptor block
          0050   954                        rqtype=cli$k_getcmd      ;  to get the whole command line
          006C   955
          006C   956 .show   meb
          006C   957
          006C   958 cli_result:                                     ; cli parse result string
     0000 006C   959                .word   0
       0E 006E   960                .byte   dsc$k_dtype_t
       02 006F   961                .byte   dsc$k_class_d
00000000 0070   962                .long   0
          0074   963
          0074   964 cli_command_line:                               ; built up cli parse command line
     0000 0074   965                .word   0
       0E 0076   966                .byte   dsc$k_dtype_t
       02 0077   967                .byte   dsc$k_class_d
00000000 0078   968                .long   0
```

```
                        007C   970 .sbttl  Permanent I/O buffers
                        007C   971
                        007C   972          org     tecobuf
                        0000   973
                        0000   974 .align  page
                        0000   975
                        0000   976 ter_o_buf1:                                  ; terminal output buffer #1
00000200                0000   977          .blkb   ter_o_siz
                        0200   978
                        0200   979 ter_o_buf2:                                  ; terminal output buffer #2
00000400                0200   980          .blkb   ter_o_siz
                        0400   981
                        0400   982 ter_i_buf:                                   ; terminal input buffer
1B 59 4D                0400   983          .ascii  "MY"<27>                    ; (pre-set) initial command string
                        0403   984 .equate ter_i_buf_pre,  .-ter_i_buf
                        0003       ter_i_buf_pre:
00000600                0403   985          .blkb   ter_i_siz-<.-ter_i_buf>
                        0600   986
                        0600   987 input_nor_buf:                               ; normal input record buffer
00000E00                0600   988          .blkb   input_nor_siz
                        0E00   989
                        0E00   990 indir_cmd_buf:                               ; "ei" record buffer
00001600                0E00   991          .blkb   indir_cmd_siz
                        1600   992
                        1600   993 input_alt_buf:                               ; alternate input record buffer
00001E00                1600   994          .blkb   input_alt_siz
                        1E00   995
                        1E00   996 output_sys_buf:                              ; sys$output output record buffer
00002000                1E00   997          .blkb   output_sys_siz
```

```
                                2000   999 .sbttl  Main startup entry point
                                2000  1000
                                2000  1001         org     tecoexeini
                                0000  1002
                          0FFC  0000  1003 .entry  tec$teco, ^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11> ; startup entry point
                                0002  1004
                                0002  1005 .sbttl  Initialization code
                                0002  1006
                                0002  1007 ; set up permanent pointer registers
                                0002  1008
                     5A    01   CE  0002  1009         mnegl   #1, r10                 ; junk up the ctl$al_cmcntx pointer
                5B   00000000'EF  3E  0005  1010         movaw   r5set, r11              ; point to teco's read/write area
                                000C  1011
                                000C  1012 ; do default read/write area setup
                                000C  1013
          00000000'EF    0000'8F   B0  000C  1014         movw    #io$m_noformat, io$bin  ; set "how to do binary output"
          00000000'EF    0000'8F   B0  0015  1015         movw    #io$m_canctrlo, io$cco  ; set "how to cancel control/o"
               0000'CB   0000'8F   B0  001E  1016         movw    #spset, w^tecosp(r11)   ; set sp stack reset value
               0000'CB   0000'8F   B0  0025  1017         movw    #pdlsrt, w^tecopd(r11)  ; set teco's pdl start
                 00'AB   0000'8F   B0  002C  1018         movw    #pdlsrt, b^pdl(r11)     ;  and init the pdl
               0000'CB   0000'8F   B0  0032  1019         movw    #schsrt, w^schbuf(r11)  ; set teco's search buffer start
               0000'CB   0000'8F   B0  0039  1020         movw    #filsrt, w^filbuf(r11)  ; set teco's filename buffer start
               0000'CB   0000'8F   B0  0040  1021         movw    #tagsrt, w^tagbuf(r11)  ; set teco's tag buffer start
                 00'AB   0000'8F   B0  0047  1022         movw    #inpnor, b^inpntr(r11)  ; set input pointer to normal input
                 00'AB   0000'8F   B0  004D  1023         movw    #oupnor, b^oupntr(r11)  ; set output pointer to normal output
               0000'CB   0000'8F   B0  0053  1024         movw    #tecoch, w^tecojp(r11)  ; set default jump dispatch table
                    50     0000'CB   3E  005A  1025         movaw   w^rwsize(r11), r0       ; get start of free memory
                    51  00001388 8F   D0  005F  1026         movl    #initial_siz, r1        ; set initial text buffer/q-reg sizes
                          00'AB   50   B0  0066  1027         movw    r0, b^txstor(r11)       ; set start of text storage
                          C0'AB   51   B0  006A  1028         movw    r1, b^zmax(r11)         ;  and its size
                             50   51   C0  006E  1029         addl    r1, r0                  ;   then skip its size
                          00'AB   50   B0  0071  1030         movw    r0, b^qrstor(r11)       ; set start of q-register storage
                             50   51   C0  0075  1031         addl    r1, r0                  ;  then skip its (trial) size
                    52  0000FFFF 8F   50  C3  0078  1032         subl3   r0, #65535, r2          ; find amount of memory still free
          00E8'CF   52  000001FF 8F   CB  0080  1033         bicl3   #512-1, r2, w^still_free ; store rounded down to page multiple
                       52   00E8'CF   C2  008A  1034         subl    w^still_free, r2        ;  then find the remainder
                    00'AB   52   51   A1  008F  1035         addw3   r1, r2, b^qmax(r11)     ; set true q-register storage size
                    00'AB   0080 8F   B0  0094  1036         movw    #tec$m_et$xit, b^etype(r11) ; set default "et" flags
                      00'AB   5F 8F   9B  009A  1037         movzbw  #^a/_/, b^symspc(r11)   ;  _ is a symbol character too
                      00'AB   01   AE  009F  1038         mnegw   #1, b^outdne(r11)       ; say all sorts of output done
                 51   002C'CF   10   C1  00A3  1039         addl3   #128/8, w^ter_i_nor8_trm+4, r1 ; address bits 128-255 of mask
                             52   D4  00A9  1040         clrl    r2                      ; start at (relative) bit 0
                          00 61   52   E2  00AB  1041 10$:    bbss    r2, (r1), 20$           ; ensure that this bit is set
                      00000000'EF42   B5  00AF  1042 20$:    tstw    cnv8bt[r2]              ; a hex pair for conversion?
                             04   19  00B6  1043         blss    30$                     ; yes, leave the bit set
                          00 61   52   E5  00B8  1044         bbcc    r2, (r1), 30$           ; no, clear the bit
                E7 52   00000080 8F   F2  00BC  1045 30$:    aoblss  #128, r2, 10$           ; loop for all 128 bits...
```

```
                        00C4  1047 ; get terminal input device's characteristics
                        00C4  1048
                        00C4  1049 getdvi_ter_i:                            ; get terminal input characteristics
                        00C4  1050         $getdvi_s -                      ; get device characteristics
                        00C4  1051                 devnam=w^ter_i_devnam, - ;  of terminal input device
                        00C4  1052                 itmlst=w^getdvi_itmlst   ;  using this item list
              7E   7C   00C4                       CLRQ    -(SP)
              00   DD   00C6                       PUSHL   #0
              00   DD   00C8                       PUSHL   #0
        0000'CF    DF   00CA                       PUSHAL  w^getdvi_itmlst
        0061'CF    7F   00CE                       PUSHAQ  w^ter_i_devnam
              7E   00   3C   00D2                 MOVZWL  #0,-(SP)
              00   DD   00D5                       PUSHL   #0
00000000'GF   08   FB   00D7                       CALLS   #8,G^SYS$GETDVI
          28   50   E9   00DE  1053         blbc    r0, 20$                  ; non-terminal if any failure
 00'8F 0000'CF    91   00E1  1054         cmpb    w^devclass, #dc$_term    ; a terminal?
              20   12   00E7  1055         bneq    20$                      ; nope
 04 0004'CF   07   E1   00E9  1056         bbc     #tt$v_lower, w^devdepend, 10$ ; lowercase?
                        00EF  1057         bs      #tec$v_et$lc, b^etype(r11) ; yes, allow lower case input
      00'AB   04   88   00EF                 bisb    s^#1@tec$v_et$lc, b^etype(r11)
                        00F3  1058 10$:     $assign_s -                      ; assign channel for terminal input
                        00F3  1059                 devnam=w^ter_i_devnam, - ;  terminal input device name
                        00F3  1060                 chan=w^ter_i_chan        ;  into terminal input channel
              7E   7C   00F3                       CLRQ    -(SP)
        00F4'CF    3F   00F5                       PUSHAW  w^ter_i_chan
        0061'CF    7F   00F9                       PUSHAQ  w^ter_i_devnam
00000000'GF   04   FB   00FD                       CALLS   #4,G^SYS$ASSIGN
          FFA0'    30   0104  1061         bsbw    success_else_die         ; abort unless success completion
              4F   11   0107  1062         brb     60$                      ; go continue
                        0109  1063
                        0109  1064 20$:     bs      #tec$v_et$lc, b^etype(r11) ; lower case for non-terminals
      00'AB   04   88   0109                 bisb    s^#1@tec$v_et$lc, b^etype(r11)
                        010D  1065         $open -                          ; open
                        010D  1066                 fab=w^input_sys_fab      ;  sys$input
        07A0'CF    DF   010D                       PUSHAL  w^input_sys_fab
 00000000'GF    01   FB   0111                       CALLS   #$$.TMPT,G^SYS$OPEN
010C'CF   07AC'CF    D0   0118  1067         movl    w^input_sys_fab+fab$l_stv, w^err_msgvec+8 ; save the STV value
          FF85'      30   011F  1068         bsbw    success_else_die         ; abort unless success completion
     03   07BF'CF    91   0122  1069         cmpb    w^input_sys_fab+fab$b_rfm, #fab$c_vfc ; vfc record format?
              0E   12   0127  1070         bneq    30$                      ; nope
 50   00000000'8F    D0   0129  1071         movl    #rms$_fsz, r0            ; pre-set bad vfc size error
     0C   07DF'CF    91   0130  1072         cmpb    w^input_sys_fab+fab$b_fsz, #input_vfc_siz ; vfc size correct?
              1E   1A   0135  1073         bgtru   50$                      ; nope, go die with an error
     04   07BF'CF    91   0137  1074 30$:   cmpb    w^input_sys_fab+fab$b_rfm, #fab$c_stm ; stream record format?
              05   12   013C  1075         bneq    40$                      ; nope
     07BE'CF   02   88   013E  1076         bisb    #fab$m_cr, w^input_sys_fab+fab$b_rat ; yep, set implied lf/cr
                        0143  1077 40$:     $connect -                       ; connect
                        0143  1078                 rab=w^input_sys_rab      ;  to it
        0808'CF    DF   0143                       PUSHAL  w^input_sys_rab
 00000000'GF    01   FB   0147                       CALLS   #$$.TMPT,G^SYS$CONNECT
010C'CF   0814'CF    D0   014E  1079         movl    w^input_sys_rab+rab$l_stv, w^err_msgvec+8 ; save the STV value
          FF4F'      30   0155  1080 50$:   bsbw    success_else_die         ; abort unless success completion
                        0158  1081 60$:                                     ; continue
```

```
                        0158  1083 ; get terminal output device's characteristics
                        0158  1084
                        0158  1085 getdvi_ter_o:                          ; get terminal output characteristics
                        0158  1086         $getdvi_s -                     ; get device characteristics
                        0158  1087                 devnam=w^ter_o_devnam, - ;  of terminal output device
                        0158  1088                 itmlst=w^getdvi_itmlst  ;  using this item list
                     7E 7C  0158                 CLRQ    -(SP)
                     00 DD  015A                 PUSHL   #0
                     00 DD  015C                 PUSHL   #0
               0000'CF DF   015E                 PUSHAL  w^getdvi_itmlst
               0073'CF 7F   0162                 PUSHAQ  w^ter_o_devnam
               7E 00   3C   0166                 MOVZWL  #0,-(SP)
                     00 DD  0169                 PUSHL   #0
         00000000'GF   08 FB  016B                 CALLS   #8,G^SYS$GETDVI
                     00 DD  0172  1089         pushl   #0                      ; guess at no terminal data to look at
                  73 50 E9  0174  1090         blbc    r0, 70$                 ; non-terminal if any failure
            00'8F 0000'CF 91  0177  1091         cmpb    w^devclass, #dc$_term   ; a terminal?
                     6B 12  017D  1092         bneq    70$                     ; nope
                  50 000C'CF 9E  017F  1093         movab   w^devnam, r0            ; yep, address the device name string
                  5F 8F 80 91  0184  1094 10$:     cmpb    (r0)+, #^a/_/           ; a leading underscore?
                     FA 13  0188  1095         beql    10$                     ; yes, go ignore it
               5454 8F   FF A0  B1  018A  1096         cmpw    -1(r0), #^a/TT/         ; is it "TTc" format?
                     0F 12  0190  1097         bneq    20$                     ; no, so no decode
      00FB'CF   01 A0   41 8F  83  0192  1098         subb3   #^a/A/, 1(r0), w^ter_o_unit+1 ; set controler # * 256.
            00FA'CF 004C'CF A0  019A  1099         addw    w^unit, w^ter_o_unit    ; plus unit number
                  50 0004'CF DE  01A1  1100 20$:     moval   w^devdepend, r0         ; address device dependent bits
                  03 60   07 E1  01A6  1101         bbc     #tt$v_lower, (r0), 30$  ; lowercase?
                        00'AB B7  01AA  1102         decw    b^euflag(r11)           ; yes, say no case flagging
                  05 60   09 E0  01AD  1103 30$:     bbs     #tt$v_wrap, (r0), 40$   ; wrap?
                                 01B1  1104         bs      #tec$v_et$tru, b^etype(r11) ; no, indicate truncated lines
               00 00'AB   08 E3  01B1                  bbcs    #tec$v_et$tru, b^etype(r11), 30002$
                                 01B6          30002$:
                  05 60   0F E1  01B6  1105 40$:     bbc     #tt$v_eightbit, (r0), 50$ ; eight-bit?
                                 01BA  1106         bs      #tec$v_et$8bt, b^etype(r11) ; yes, indicate 8-bit terminal
               00 00'AB   0C E3  01BA                  bbcs    #tec$v_et$8bt, b^etype(r11), 30003$
                                 01BF          30003$:
                  11 60   0C E1  01BF  1107 50$:     bbc     #tt$v_scope, (r0), 60$  ; scope?
      00000000'EF 0000'CF   0C 28  01C3  1108         movc    #getdvi_info_len, w^getdvi_info, errbuf ; save scope info
               6E 00000002'EF 9E  01CD  1109         movab   errbuf+2, (sp)          ;  and form pointer to the data @ +2
                                 01D4  1110 60$:     $assign_s -                     ; assign channel for terminal output
                                 01D4  1111                 devnam=w^ter_o_devnam, - ;  terminal outut device name
                                 01D4  1112                 chan=w^ter_o_chan       ;  into terminal output channel
                     7E 7C  01D4                 CLRQ    -(SP)
               00F6'CF 3F   01D6                 PUSHAW  w^ter_o_chan
               0073'CF 7F   01DA                 PUSHAQ  w^ter_o_devnam
         00000000'GF   04 FB  01DE                 CALLS   #4,G^SYS$ASSIGN
                  FEBF' 30  01E5  1113         bsbw    success_else_die        ; abort unless success completion
                     32 11  01E8  1114         brb     80$                     ; go continue
                                 01EA  1115
                  00'AB B7  01EA  1116 70$:     decw    b^euflag(r11)           ; no case flagging for non-terminals
                                 01ED  1117         bs      #tec$v_et$8bt, b^etype(r11) ; 8-bit mode for non-terminals
               00 00'AB   0C E3  01ED                  bbcs    #tec$v_et$8bt, b^etype(r11), 30004$
                                 01F2          30004$:
                                 01F2  1118         $create -                       ; create
                                 01F2  1119                 fab=w^output_sys_fab    ;  sys$output
               084C'CF DF   01F2                 PUSHAL  w^output_sys_fab
         00000000'GF   01 FB  01F6                 CALLS   #$$.TMP1,G^SYS$CREATE
```

```
                010C'CF    0858'CF   DO  01FD  1120          movl    w^output_sys_fab+fab$l_stv, w^err_msgvec+8 ; save STV value
                              FEA0'  30  0204  1121          bsbw    success_else_die           ; abort unless success completion
                                     0207  1122          $connect -                         ; connect
                                     0207  1123                  rab=w^output_sys_rab       ;  to it
                           089C'CF   DF  0207                PUSHAL  w^output_sys_rab
                 00000000'GF   01    FB  020B                CALLS   #$$.TMP1,G^SYS$CONNECT
                010C'CF    08A8'CF   DO  0212  1124          movl    w^output_sys_rab+rab$l_stv, w^err_msgvec+8 ; save STV value
                              FE8B'  30  0219  1125          bsbw    success_else_die           ; abort unless success completion
                                     021C  1126 80$:                                           ; continue
```

```
                     021C  1128 ; get terminal command device's characteristics
                     021C  1129
                     021C  1130 getdvi_ter_c:                                ; get terminal command characteristics
                     021C  1131         $getdvi_s -                          ; get device characteristics
                     021C  1132                   devnam=w^ter_c_devnam, - ;  of terminal command device
                     021C  1133                   itmlst=w^getdvi_itmlst   ;  using this item list
           7E   7C   021C                         CLRQ    -(SP)
           00   DD   021E                         PUSHL   #0
           00   DD   0220                         PUSHL   #0
     0000'CF    DF   0222                         PUSHAL  w^getdvi_itmlst
     007B'CF    7F   0226                         PUSHAQ  w^ter_c_devnam
     7E    00   3C   022A                         MOVZWL  #0,-(SP)
           00   DD   022D                         PUSHL   #0
00000000'GF    08   FB   022F                     CALLS   #8,G^SYS$GETDVI
        22  50   E9   0236  1134           blbc    r0, 10$                  ; non-terminal if any failure
 00'8F  0000'CF   91   0239  1135           cmpb    w^devclass, #dc$_term    ; a terminal?
           1A   12   023F  1136           bneq    10$                      ; nope
                     0241  1137           $assign_s -                      ; assign channel for control/c ast's
                     0241  1138                     devnam=w^ter_c_devnam, - ;  terminal command device name
                     0241  1139                     chan=w^ter_c_chan      ;  into terminal control/c ast channel
           7E   7C   0241                         CLRQ    -(SP)
     00F8'CF    3F   0243                         PUSHAW  w^ter_c_chan
     007B'CF    7F   0247                         PUSHAQ  w^ter_c_devnam
00000000'GF    04   FB   024B                     CALLS   #4,G^SYS$ASSIGN
        FE52'   30   0252  1140           bsbw    success_else_die         ; abort unless success completion
        FF74'   30   0255  1141           bsbw    enable_ctrlcast          ; go enable the control/c ast
        FE4C'   30   0258  1142           bsbw    success_else_die         ; abort unless success completion
                     025B  1143 10$:                                       ; continue
```

```
                        025B  1145 ; pre-load q-register y (command decoder) and z (command line) if needed
                        025B  1146
                        025B  1147 .enable lsb
                        025B  1148
                        025B  1149 pre_load_q_regs:                        ; pre-load q-registers y and z
     56  0050'CF   DE  025B  1150         moval   w^cli_req_getcmd, r6      ; get address of request block
                   66   DF  0260  1151         pushal  (r6)                ; set address of request block
00000000'GF        01   FB  0262  1152         calls   #1, g^sys$cli       ; go get the command line
                   50   B5  0269  1153         tstw    r0                  ; the special "no command" status?
                   03   12  026B  1154         bneq    10$                 ; nope
                 0210   31  026D  1155         brw     210$                ; yep, go continue
                        0270  1156
                 FE34'  30  0270  1157 10$:    bsbw    success_else_die    ; abort unless success completion
   00'8F         03 A6  91  0273  1158         cmpb    cli$b_rqstat(r6), #cli$k_verb_edit ; is this EDIT/TECO?
                   03   13  0278  1159         beql    20$                 ; yes
                 011E   31  027A  1160         brw     110$                ; no
                        027D  1161
     55  009A'CF   7E  027D  1162 20$:    movaq   w^cli_verb_teco, r5       ; set TECO verb
                 01EB   30  0282  1163         bsbw    200$                ;  as command line verb
               0119'CF  7F  0285  1164         pushaq  w^cli_qual_command  ; /COMMAND
00000000'GF        01   FB  0289  1165         calls   #1, g^cli$present   ;  present?
                   40 50  E9  0290  1166         blbc    r0, 30$             ; no, go use /NOINI
               006C'CF  7F  0293  1167         pushaq  w^cli_result        ; address result string
               0119'CF  7F  0297  1168         pushaq  w^cli_qual_command  ; /COMMAND=file
00000000'GF        02   FB  029B  1169         calls   #2, g^cli$get_value ;  have a value?
                   36 50  E9  02A2  1170         blbc    r0, 40$             ; no
               006C'CF  7F  02A5  1171         pushaq  w^cli_result        ; yes, form
               0100'CF  7F  02A9  1172         pushaq  w^cli_dollar        ;  "$file"
               006C'CF  7F  02AD  1173         pushaq  w^cli_result        ;   in the result
00000000'GF        03   FB  02B1  1174         calls   #3, g^str$concat    ;    using concatenation
                 FDEC'  30  02B8  1175         bsbw    success_else_die    ; abort unless success completion
                        02BB  1176         $crelog_s -                     ; create a logical
                        02BB  1177                 tblflg = #2, -          ;  use the process table
                        02BB  1178                 lognam = w^cli_init, -  ;  logical is TEC$INIT
                        02BB  1179                 eqlnam = w^cli_result   ;  with this equivalence string
                   00   DD  02BB                   PUSHL   #0
               006C'CF  7F  02BD                   PUSHAQ  w^cli_result
               0109'CF  7F  02C1                   PUSHAQ  w^cli_init
                   02   DD  02C5                   PUSHL   #2
00000000'GF        04   FB  02C7                   CALLS   #4,G^SYS$CRELOG
                 FDD6'  30  02CE  1180         bsbw    success_else_die    ; abort unless success completion
                   08   11  02D1  1181         brb     40$                 ; else continue
                        02D3  1182
     55  00A6'CF   7E  02D3  1183 30$:    movaq   w^cli_no_ini, r5          ; set /NOINI
                 0195   30  02D8  1184         bsbw    200$                ;  and go add it
               0128'CF  7F  02DB  1185 40$:    pushaq  w^cli_qual_create   ; /CREATE
00000000'GF        01   FB  02DF  1186         calls   #1, g^cli$present   ;  present?
                   08 50  E8  02E6  1187         blbs    r0, 50$             ; yes
     55  00B4'CF   7E  02E9  1188         movaq   w^cli_no_create, r5       ; no, set /NOCREATE
                 017F   30  02EE  1189         bsbw    200$                ;  and go add it
               0136'CF  7F  02F1  1190 50$:    pushaq  w^cli_qual_memory   ; /MEMORY
00000000'GF        01   FB  02F5  1191         calls   #1, g^cli$present   ;  present?
                   08 50  E8  02FC  1192         blbs    r0, 60$             ; yes
     55  00C5'CF   7E  02FF  1193         movaq   w^cli_no_memory, r5       ; no, set /NOMEMORY
                 0169   30  0304  1194         bsbw    200$                ;  and go add it
     55  00EE'CF   7E  0307  1195 60$:    movaq   w^cli_space, r5           ; set "," as the next separator
               006C'CF  7F  030C  1196         pushaq  w^cli_result        ; address result string
```

```
                0144'CF   7F   0310  1197          pushaq  w^cli_qual_execute      ; /EXECUTE=file
      00000000'GF    02   FB   0314  1198          calls   #2, g^cli$get_value     ;  present with a value?
              54    50   D0   031B  1199          movl    r0, r4                 ; remember /EXECUTE's existence
              18    50   E9   031E  1200          blbc    r0, 70$                ; no
00000078'FF   474E554D   8F   D0   0321  1201          movl    #^a/MUNG/, @cli_command_line+4 ; yes, change verb to MUNG
                0141   30   032C  1202          bsbw    200$                   ; add the " " separator
                0139   30   032F  1203          bsbw    190$                   ; add /EXECUTE's value
           55   00EE'CF   7E   0332  1204          movaq   w^cli_space, r5        ; set " " as the next separator
                  25   11   0337  1205          brb     80$                    ;  and go check out P1
                       0339  1206
                006C'CF   7F   0339  1207 70$:     pushaq  w^cli_result           ; address result string
                015D'CF   7F   033D  1208          pushaq  w^cli_qual_output      ; /OUTPUT=file
      00000000'GF    02   FB   0341  1209          calls   #2, g^cli$get_value    ;  present with a value?
              13    50   E9   0348  1210          blbc    r0, 80$                ; no
                0122   30   034B  1211          bsbw    200$                   ; yes, add the " " separator
                011A   30   034E  1212          bsbw    190$                   ; add /OUTPUT's value
           55   00F7'CF   7E   0351  1213          movaq   w^cli_equals, r5       ; set "="
                0117   30   0356  1214          bsbw    200$                   ;  and add it
           55   00E6'CF   7E   0359  1215          movaq   w^cli_null, r5         ; set "" as the next separator
                006C'CF   7F   035E  1216 80$:     pushaq  w^cli_result           ; address result string
                0153'CF   7F   0362  1217          pushaq  w^cli_parm_p1          ; parameter P1
      00000000'GF    02   FB   0366  1218          calls   #2, g^cli$get_value    ;  present with a value?
              06    50   E9   036D  1219          blbc    r0, 90$                ; no
                00FD   30   0370  1220          bsbw    200$                   ; yes, add the " " or "" separator
                00F5   30   0373  1221          bsbw    190$                   ; add P1's value
              16    54   E8   0376  1222 90$:     blbs    r4, 100$               ; all done if /EXECUTE exists
                016B'CF   7F   0379  1223          pushaq  w^cli_qual_read_only   ; /READ_ONLY
      00000000'GF    01   FB   037D  1224          calls   #1, g^cli$present      ;  present?
              08    50   E9   0384  1225          blbc    r0, 100$               ; no
           55   00D6'CF   7E   0387  1226          movaq   w^cli_inspect, r5      ; yes, set /INSPECT
                  00E1   30   038C  1227          bsbw    200$                   ;  and go add it
     08 A6     0074'CF   B0   038F  1228 100$:    movw    w^cli_command_line, cli$w_rqsize(r6) ; set command line length
     0C A6     0078'CF   D0   0395  1229          movl    w^cli_command_line+4, cli$a_rqaddr(r6) ;  and address
           53    00'AB   3C   039B  1230 110$:    movzwl  b^qrstor(r11), r3      ; get pointer to q-register storage
                50    18   9A   039F  1231          movzbl  #^a/Y/-^a/A/, r0       ; we're loading q-register y
                51   0000'8F   3C   03A2  1232          movzwl  #tecocmd_siz, r1       ; get command decoder's size
                52   0000'CF   9E   03A7  1233          movab   w^tecocmd, r2          ;  and a pointer to it
                      00A1   30   03AC  1234          bsbw    180$                   ; go load a q-register
                50    19   9A   03AF  1235          movzbl  #^a/Z/-^a/A/, r0       ; we're loading q-register z
                  51    08 A6   3C   03B2  1236          movzwl  cli$w_rqsize(r6), r1   ; get command line's size
                  52    0C A6   D0   03B6  1237          movl    cli$a_rqaddr(r6), r2   ; get a pointer to command line
                       0093   30   03BA  1238          bsbw    180$                   ; go load a q-register
                56   0378'CF   DE   03BD  1239          moval   w^indir_cmd_fab, r6    ; get the indirect command fab
                              03C2  1240          getdesc tmp_string, r0         ; reset & get desc for temp string
                50   0058'CF   9E   03C2                movab   w^tmp_string_buf, r0
                     70    50   D0   03C7                movl    r0, -(r0)
                     70    3F   3C   03CA                movzwl  #tmp_string_siz, -(r0)
                              03CD  1241          $trnlog_s -                    ; translate
                              03CD  1242                    lognam=ini_dcd_lognam, - ;  logical name "TECO"
                              03CD  1243                    rslbuf=(r0)            ;  putting result into the temp
                           00   DD   03CD                     PUSHL   #0
                           00   DD   03CF                     PUSHL   #0
                           00   DD   03D1                     PUSHL   #0
                           60   7F   03D3                     PUSHAQ  (r0)
                           00   DD   03D5                     PUSHL   #0
                 0000008E'EF   7F   03D7                     PUSHAQ  ini_dcd_lognam
00000000'GF    06   FB   03DD                     CALLS   #6,G^SYS$TRNLOG
```

```
                0000'8F    50   B1  03E4  1244            cmpw    r0, #ss$_notran         ; did anything happen?
                      0F   13   B1  03E9  1245            beql    130$                    ; nope, use pre-set "SYS$LOGIN:TECO"
                  FCB9'    30   B1  03EB  1246            bsbw    success_else_die        ; abort unless success completion
        2C B6    4F434554 8F   D0  03EE  1247            movl    #^a/TECO/, @fab$l_fna(r6) ; set file name of simply "TECO"
                  34 A6    04   90  03F6  1248            movb    #4, fab$b_fns(r6)       ;  and its length
                                    03FA  1249 130$:      $open -                         ; open
                                    03FA  1250                    fab=(r6)                ;  the command decoder file file
                           66   DF  03FA                  PUSHAL  (r6)
                00000000'GF     01   FB  03FC                  CALLS   #$$.TMP1,G^SYS$OPEN
                        3F 50   E9  0403  1251            blbc    r0, 170$                ; branch if failure of any type
                     03   1F A6   91  0406  1252            cmpb    fab$b_rfm(r6), #fab$c_vfc ; vfc record format?
                           06   12  040A  1253            bneq    140$                    ; nope
                     0C   3F A6   91  040C  1254            cmpb    fab$b_fsz(r6), #input_vfc_siz ; vfc size correct?
                           2A   1A  0410  1255            bgtru   160$                    ; nope, so don't use it...
                     04   1F A6   91  0412  1256 140$:      cmpb    fab$b_rfm(r6), #fab$c_stm ; stream record format?
                           04   12  0416  1257            bneq    150$                    ; nope
                     1E A6    02   88  0418  1258            bisb    #fab$m_cr, fab$b_rat(r6) ; always say implied lf/cr for stream
                                    041C  1259 150$:      $connect -                      ; connect
                                    041C  1260                    rab=@fab$l_tecrab(r6)   ;  the correct rab
                        54 B6    DF   041C                  PUSHAL  @fab$l_tecrab(r6)
                00000000'GF     01   FB  041F                  CALLS   #$$.TMP1,G^SYS$CONNECT
                        13 50   E9  0426  1261            blbc    r0, 160$                ; branch if failure of any type
                57   00000000'EF   DE  0429  1262            moval   cmdprm, r7              ; get where to store fab pointer
                        67   56   D0  0430  1263            movl    r6, (r7)                ; set file as open
                          08A6'  30   B1  0433  1264            bsbw    set_filename            ; set the file's name, etc.
                     00'AB   57   B0  0436  1265            movw    r7, b^indir(r11)        ; set indirect as active
                           44   11  043A  1266            brb     210$                    ; go continue
                                    043C  1267
                                    043C  1268 160$:      $close -                        ; close
                                    043C  1269                    fab=(r6)                ;  the (bad) command decoder file file
                           66   DF  043C                  PUSHAL  (r6)
                00000000'GF     01   FB  043E                  CALLS   #$$.TMP1,G^SYS$CLOSE
                    0020'CF     03   3C  0445  1270 170$:      movzwl  #ter_i_buf_pre, w^ter_i ; set count for pre-buffered input
                      00'AB     02   B0  044A  1271            movw    #2, b^indir(r11)        ; preset for initial command string
                           30   11  044E  1272            brb     210$                    ; go continue
                                    0450  1273
                  00'AB40     51   D0  0450  1274 180$:      movl    r1, b^qarray(r11)[r0]   ; set q-reg's size (clobbers next!)
                    00'AB     51   A0  0455  1275            addw    r1, b^qz(r11)           ; count as q-reg space used
                00'AB   00'AB   B1   0459  1276            cmpw    b^qz(r11), b^qmax(r11)  ; did we run out of space?
                           05   1A  045E  1277            bgtru   abort_exit              ; whoops, we did
                63   62    51   28  0460  1278            movc    r1, (r2), (r3)          ; load the q-register
                           05   0464  1279            rsb                             ; exit
                                    0465  1280
                                    0465  1281 abort_exit:                               ; set abort error code and exit
                        50   00'   3C  0465  1282            movzwl  s^#ss$_abort, r0        ; set a fatal error code
                           FC3C'   31   0468  1283            brw     success_else_die        ;  and go exit with it
                                    046B  1284
                55   006C'CF     7E   046B  1285 190$:      movaq   w^cli_result, r5        ; address the cli result string
                           65   7F  0470  1286 200$:      pushaq  (r5)                    ; arg #2 is string to add
                     0074'CF     7F   0472  1287            pushaq  w^cli_command_line      ; arg #1 is string to add to
                00000000'GF     02   FB  0476  1288            calls   #2, g^str$append        ; go append to string
                           FC27'   31   047D  1289            brw     success_else_die        ; error check and exit
                                    0480  1290
                                    0480  1291 210$:                                   ; continue
                                    0480  1292
                                    0480  1293 .disable lsb
```

```
                        0480  1295 ; clean up and enter compatibility mode
                        0480  1296
                        0480  1297 clean_up_and_start:                    ; clean up & enter compatibility mode
                00F6'CF   B5  0480  1298         tstw    w^ter_o_chan       ; is there a terminal output channel?
                      18  13  0484  1299         beql    10$                ; nope, so no mode(s) to reset
                0004'CF   DD  0486  1300         pushl   w^devdepend        ; save the original characteristics
0004'CF   00100008 8F   CA  048A  1301         bicl    #tt$m_halfdup!tt$m_escape, w^devdepend ; full dup & not escape
                0D5D'   30  0493  1302         bsbw    tec$setmode        ; go reset the terminal mode(s)
              0004'CF   8E  D0  0496  1303         movl    (sp)+, w^devdepend ; restore original characteristics
                FC09'   30  049B  1304         bsbw    success_else_die   ; abort unless success completion
                        049E  1305 10$:      $dclcmh_s -                   ; declare compatibility mode handler
                        049E  1306                 addres=w^tec$cmtrap, - ;  traps will come here
                        049E  1307                 type=#1                ;  and we want compatibility traps
                      01  DD  049E                 PUSHL   #1
                      00  DD  04A0                 PUSHL   #0
                0000'CF   DF  04A2                 PUSHAL  w^tec$cmtrap
          00000000'GF   03  FB  04A6                 CALLS   #3,G^SYS$DCLCMH
                FBF7'   30  04AD  1308         bsbw    success_else_die   ; abort unless success completion
                      52  7C  04B0  1309         clrq    r2                 ; clear r2, r3
                    54    8E  D0  04B2  1310         movl    (sp)+, r4          ; set r4 if scope terminal, else clear
                    55    5B  D0  04B5  1311         movl    r11, r5            ; set teco's r5 r/w area pointer
              56    00000000'EF   3E  04B8  1312         movaw   spset, r6          ; set to reset the sp stack
                83C00000 8F   DD  04BF  1313         pushl   #psl$m_cm!<psl$c_user@psl$v_curmod>!<psl$c_user@psl$v_prvmod>
                00000000'EF   3F  04C5  1314         pushaw  tecost             ; stack startup pc/psl pair
                FBA9'   31  04CB  1315         brw     start_teco         ; go purge working set, then start up!
```

```
                                04CE  1317 .sbttl  Compatibility mode trap handler
                                04CE  1318
                                04CE  1319          org     tecoexe
                                0000  1320
                                0000  1321 .align  page
                                0000  1322
                                0000  1323 tec$cmtrap:                          ; compatibility mode traps come here
          5A   50    20    C3   0000  1324          subl3   #i_bias, r0, r10    ; form ctl$al_cmcntx pointer
               51    90    9A   0004  1325          movzbl  @(r0)+, r1          ; get low byte of instruction
               60    01    CA   0007  1326          bicl    #1, (r0)            ; pre-clear c-bit in saved ps
               70    02    C0   000A  1327          addl    #2, -(r0)           ; adjust saved pc
             08 51    07    E5   000D  1328          bbcc    #7, r1, 10$         ; check for automatic "rts pc"
             60    18 BA    3C   0011  1329          movzwl  @i_sp(r10), (r0)    ; move return address to saved pc
             18 AA    02    C0   0015  1330          addl    #2, i_sp(r10)       ;  then 'pop' the stack
                      10    10   0019  1331 10$:     bsbb    20$                 ; go dispatch on exception code
                   50    8A   7D   001B  1332          movq    (r10)+, r0          ; restore r0, r1
                   52    8A   7D   001E  1333          movq    (r10)+, r2          ; restore r2, r3
                   54    8A   7D   0021  1334          movq    (r10)+, r4          ; restore r4, r5
                   56    8A   7D   0024  1335          movq    (r10)+, r6          ; restore sp, junk r7
                   7E    6A   7D   0027  1336          movq    (r10), -(sp)        ; stack pc/psl pair
                              02   002A  1337          rei                         ; back to whatever...
                                   002B  1338
          00E4'CF    5E    D0   002B  1339 20$:     movl    sp, w^saved_sp      ; save calling sp for error exits
    0003'8F   01    1C AA    AF   0030  1340          casew   i_code(r10), #1, #<<40$-30$>/2>-1 ; enter proper routine...
                                   0037  1341 30$:                             ; reference only
                         0207'     0037  1342           .word  tec$wait-30$         ; 1 => bpt
                         033A'     0039  1343           .word  tec$output-30$       ; 2 => iot
                         000B'     003B  1344           .word  50$-30$              ; 3 => emt
                         053F'     003D  1345           .word  tec$input-30$        ; 4 => trap
                                   003F  1346 40$:                             ; reference only
                      0423'  31    003F  1347          brw     abort_exit          ; whoops, we must abort...
                                   0042  1348
                                   0042  1349 .macro  other   name
                                   0042  1350 $'name  ==      <<.-60$>/2>+1
                                   0042  1351           .word  tec$'name-60$
                                   0042  1352 .endm   other
                                   0042  1353
    0015'8F   01    51    AF     0042  1354 50$:     casew   r1, #1, #<<70$-60$>/2>-1 ; 'emt' is other...
                                   0048  1355 60$:                             ; reference only
                                   0048  1356          other   width               ; new terminal width
                         119E'     0048                .word  tec$width-60$
                                   004A  1357          other   eight               ; change 8-bit terminal mode
                         1171'     004A                .word  tec$eight-60$
                                   004C  1358          other   truln               ; change truncate lines mode
                         1166'     004C                .word  tec$truln-60$
                                   004E  1359          other   ejflg               ; get ej flag information
                         1110'     004E                .word  tec$ejflg-60$
                                   0050  1360          other   gexit               ; process special functions
                         12A9'     0050                .word  tec$gexit-60$
                                   0052  1361          other   sizer               ; get additional memory
                         1374'     0052                .word  tec$sizer-60$
                                   0054  1362          other   date                ; get date
                         140B'     0054                .word  tec$date-60$
                                   0056  1363          other   time                ; get time
                         1423'     0056                .word  tec$time-60$
                                   0058  1364          other   getfl               ; get files opened
                         0D7F'     0058                .word  tec$getfl-60$
```

```
                005A  1365          other   inpsv                   ; switch input file
0C6F'           005A                .word   tec$inpsv-60$
                005C  1366          other   outsv                   ; switch output file
0C63'           005C                .word   tec$outsv-60$
                005E  1367          other   bakup                   ; page backwards
0836'           005E                .word   tec$bakup-60$
                0060  1368          other   getbf                   ; get input
08CA'           0060                .word   tec$getbf-60$
                0062  1369          other   putbf                   ; put output
09FF'           0062                .word   tec$putbf-60$
                0064  1370          other   clsfl                   ; close input & output files
0CCB'           0064                .word   tec$clsfl-60$
                0066  1371          other   clsof                   ; close output files
0CCD'           0066                .word   tec$clsof-60$
                0068  1372          other   aller                   ; finish up on error processing
0D6C'           0068                .word   tec$aller-60$
                006A  1373          other   kilfl                   ; delete output file
0D14'           006A                .word   tec$kilfl-60$
                006C  1374          other   delln                   ; echo line deletion
0721'           006C                .word   tec$delln-60$
                006E  1375          other   delch                   ; echo character deletion
0757'           006E                .word   tec$delch-60$
                0070  1376          other   xitnw                   ; stop teco terminal hacks
11D8'           0070                .word   tec$xitnw-60$
                0072  1377          other   texit                   ; exit from teco
1468'           0072                .word   tec$texit-60$
                0074  1378 70$:                                      ; reference only
03EE'  31       0074  1379          brw     abort_exit              ; whoops, we must abort...
```

```
                        0077  1381 .sbttl  Initial start up
                        0077  1382
                        0077  1383 start_teco:                                ; initial start of compatibility mode
                        0077  1384          $purgws_s -                       ; purge our working set
                        0077  1385                    inadr=b^10$             ;  do the whole thing...
                87'AF    7F  0077                     PUSHAQ  b^10$
00000000'GF       01    FB  007A                     CALLS   #1,G^SYS$PURGWS
                003E    30  0081  1386          bsbw    success_or_announce   ; announce any failure...
                  50    7C  0084  1387          clrq    r0                    ; clear r0, r1
                        02  0086  1388          rei                           ; enter compatibility mode...
                        0087  1389
                00000000  0087  1390 10$:     .long   0                       ; purge from 00000000
                7FFFFFFF  008B  1391          .long   1a31-1                  ;  to 7FFFFFFF
```

```
                                008F  1393 .sbttl  Error processing, etc.
                                008F  1394
                                008F  1395 .enable lsb
                                008F  1396
    51    010C'CF    DE    008F  1397 10$:     moval   w^err_msgvec+8, r1      ; point to message vector spot
          71    50   D0    0094  1398         movl    r0, -(r1)               ; load the error status code
71    000F0001 8F    D0    0097  1399         movl    #<<8!4!2!1>@16>!1, -(r1) ; fac, sev, id, & text; arg count = 1
     00'    06   A1    B1    009E  1400         cmpw    6(r1), s^#rms$_normal/65536 ; is this an RMS error?
                02   12    00A2  1401         bneq    20$                     ; nope
                61   B6    00A4  1402         incw    (r1)                    ; yep, include the STV value in count
                05   00A6  1403 20$:    rsb                             ; exit
                     00A7  1404
                     00A7  1405 success_else_die:                       ; abort unless success completion
              FC 50   E8    00A7  1406         blbs    r0, 20$                 ; exit if success
                50   DD    00AA  1407         pushl   r0                      ; failure, save the exit reason code
                14   10    00AC  1408         bsbb    success_or_announce     ; go announce the failure
          6E    07   CA    00AE  1409         bicl    #sts$m_severity, (sp)   ; trim out severity code in status
8E  10000004 8F    C9    00B1  1410         bisl3   #sts$m_inhib_msg!sts$k_severe, - ; inhibit msg & severe error
                50         00B8  1411                 (sp)+, r0               ;  get forced on in status code
                     00B9  1412         $exit_s -                       ; force an image exit
                     00B9  1413                 code = r0               ;  with the reason code
                50   DD    00B9                  PUSHL   r0
00000000'GF     01   FB    00BB                  CALLS   #1,G^SYS$EXIT
                     00C2  1414
                     00C2  1415 success_or_announce:                    ; check for success; announce if not
              7F 50   E8    00C2  1416         blbs    r0, 60$                 ; exit if success
                C8   10    00C5  1417         bsbb    10$                     ; go format the message vector
                     00C7  1418         $putmsg_s -                     ; use $PUTMSG to format the message
                     00C7  1419                 msgvec = (r1)           ;  using the built up message vector
                00   DD    00C7                  PUSHL   #0
                00   DD    00C9                  PUSHL   #0
                00   DD    00CB                  PUSHL   #0
                61   DF    00CD                  PUSHAL  (r1)
    00000000'GF     04   FB    00CF                  CALLS   #4,G^SYS$PUTMSG
                     00D6  1420         ;bsbw   success_or_announce     ; announce any failure...
                05   00D6  1421         rsb                             ; exit
                     00D7  1422
                     00D7  1423 success_or_abrt:                        ; check for success; abort if not
              6A 50   E8    00D7  1424         blbs    r0, 60$                 ; exit if success
20 AA   00000000'EF   3E    00DA  1425         movaw   ioerr, i_pc(r10)        ; else exit to teco's error processor
                1D   11    00E2  1426         brb     30$                     ; continue
                     00E4  1427
                     00E4  1428 success_or_cls:                         ; check success; close & error if not
              5D 50   E8    00E4  1429         blbs    r0, 60$                 ; exit if success
          51  54   A6    00E7  1430         movl    fab$l_tecrab(r6), r1    ; get the rab pointer
    010C'CF   0C   A1    00EB  1431         movl    rab$l_stv(r1), w^err_msgvec+8 ; save the STV value
                01   BB    00F1  1432         pushr   #^m<r0>                 ; save the failure status
                     00F3  1433         $close -                        ; close
                     00F3  1434                 fab=(r6)                ;  the (failing) fab
                66   DF    00F3                  PUSHAL  (r6)
    00000000'GF     01   FB    00F5                  CALLS   #$$.TMP1,G^SYS$CLOSE
                01   BA    00FC  1435         popr    #^m<r0>                 ; restore the failure status
                     00FE  1436 success_or_err:                         ; check for success; error if not
              43 50   E8    00FE  1437         blbs    r0, 60$                 ; exit if success
                8C   10    0101  1438 30$:    bsbb    10$                     ; go format the message vector
                     0103  1439 rad50   ERR
      6A    2222 8F    3C    0103  1440         movzwl  #$$$$$$, i_r0(r10)      ; set rad50 code for ERR into saved r0
```

```
                00000000'8F    04 A1    D1  0108  1441            cmpl    4(r1), #rms$_fnf        ; was the error file not found?
                               09 12    12  0110  1442            bneq    40$                     ; it was not
                         02 A1    01    B0  0112  1443            movw    #1, 2(r1)               ; it was, set only message text
                                        01  0116  1444 rad50      FNF
                      6A    27B6 8F    3C  0116  1445            movzwl  #$$$$$$, i_r0(r10)      ; set rad50 code for FNF into saved r0
                       00000000'EF     94  011B  1446 40$:       clrb    errbuf                  ; "empty" the error message buffer
                                       01  0121  1447            $putmsg_s -                     ; use $PUTMSG to format the message
                                       01  0121  1448                    msgvec = (r1), -        ;  using the built up message vector
                                       01  0121  1449                    actrtn = b^70$          ;  catch the message w/ action routine
                                    00 DD  0121                        PUSHL   #0
                                    00 DD  0123                        PUSHL   #0
                                 5E'AF DF  0125                        PUSHAL  b^70$
                                    61 DF  0128                        PUSHAL  (r1)
                          00000000'GF 04 FB  012A                       CALLS   #4,G^SYS$PUTMSG
                                FF8E  30  0131  1450            bsbw    success_or_announce     ; announce any failure...
                08 AA    00000000'EF  9E  0134  1451 50$:       movab   errbuf, i_r2(r10)       ; set error message addr into saved r2
                              24 AA  D6  013C  1452            incl    i_ps(r10)               ; set c-bit in saved ps
                      5E    00E4'CF  D0  013F  1453            movl    w^saved_sp, sp          ; restore calling sp for error exit
                                     05  0144  1454 60$:       rsb                             ; exit
                                         0145  1455
                                         0145  1456 err:                                     ; errors come here
                                 56  9E  D0  0145  1457            movl    @(sp)+, r6              ; get pointer to: code, len, text
                                 6A  86  3C  0148  1458            movzwl  (r6)+, i_r0(r10)        ; set rad50 code into saved r0
                                 50  86  9A  014B  1459            movzbl  (r6)+, r0               ; get length of text string
                FFFF'8F    00    66  50  2C  014E  1460            movc5   r0, (r6), #0, #errbfl-1, errbuf ; move message text
                            00000000'EF      0155
                                    63  94  015A  1461            clrb    (r3)                    ;  and ensure result is asciz
                                    D6  11  015C  1462            brb     50$                     ; go set the message text address
                                        015E  1463
                                 003C   015E  1464 70$:          .word   ^m<r2,r3,r4,r5>         ; $PUTMSG action routine
                                   52 04 BC  7D  0160  1465            movq    @4(ap), r2              ; get message text descriptor
                00000000'EF     FFFF'8F 00  3A  0164  1466            locc    #0, #errbfl-1, errbuf   ; find asciz ending of message buffer
                                FFFF'8F 50  B1  016E  1467            cmpw    r0, #errbfl-1           ; is the whole message buffer free?
                                        14  1E  0173  1468            bgequ   80$                     ; yes, no need for prefixing
                                54  52  03  A1  0175  1469            addw3   #3, r2, r4              ; add prefix size to message size
                                    54  50  B1  0179  1470            cmpw    r0, r4                  ; would the whole thing fit?
                                        13  1F  017C  1471            blssu   90$                     ; nope
                                    50  03  A2  017E  1472            subw    #3, r0                  ; yep, remove prefix size from free
                                81  0A0D 8F  B0  0181  1473            movw    #<10a8>!13, (r1)+       ; prefix message with <CR><LF>
                                    81  09  90  0186  1474            movb    #9, (r1)+               ;  and <TAB>
                            61  50  00  63  52  2C  0189  1475 80$:       movc5   r2, (r3), #0, r0, (r1)  ; move message text into buffer
                                        63  94  018F  1476            clrb    (r3)                    ;  and ensure result is asciz
                                        50  D4  0191  1477 90$:       clrl    r0                      ; don't put the message to user
                                            04  0193  1478            ret                             ;  when I return to you
                                                0194  1479
                                                0194  1480 .disable lsb
```

```
                        0194  1482 .sbttl  Control/c ASTs
                        0194  1483
                        0194  1484 tec$ctrlcast:                              ; control/c ast's come here
                0000    0194  1485         .word   ^m<>                       ; no need to save any registers
 14 00000000'EF   0F     E4  0196  1486         bbsc    #tec$v_et$cc, etype+r5set, 10$ ; user trapping control/c's?
    00000001'EF   01     8E  019E  1487         mnegb   #1, tflg+1+r5set       ; no, set the stop soon indicator
 05 00000000'EF   07     E1  01A5  1488         bbc     #tec$v_et$xit, etype+r5set, 10$ ; exit on control/c's?
       00FC'CF   00     92  01AD  1489         mcomb   #0, w^ctrlc_flag       ; yes, ensure flipped flop
                 18     10  01B2  1490 10$:    bsbb    enable_ctrlcast        ; go re-enable the control/c ast
               FF0B     30  01B4  1491         bsbw    success_or_announce    ; announce any failure...
       00FD'CF   00     92  01B7  1492         mcomb   #0, w^ctrlo_flag       ; say control/o is now in effect
                        01BC  1493         $cancel_s -                        ; cancel all pending I/O
                        01BC  1494                 chan=w^ter_o_chan          ;  on the terminal output channel
                        01BC                       MOVZWL  w^ter_o_chan,-(SP)
    00000000'GF   01     FB  01C1                  CALLS   #1,G^SYS$CANCEL
               FEF7     30  01C8  1495         bsbw    success_or_announce    ; announce any failure...
                 04     01  01CB  1496         ret                            ; return
                        01CC  1497
                        01CC  1498 enable_ctrlcast:                           ; enable the control/c ast
                        01CC  1499         $qiow_s -                          ; set up the control/c ast with a qio
                        01CC  1500                 chan=w^ter_c_chan, -       ;  using the control/c ast channel
                        01CC  1501                 func=#io$_setmode!io$m_ctrlcast, - ;  function = control/c ast
                        01CC  1502                 p1=b^tec$ctrlcast          ;  go to here on control/c ast's
                   7E    7C  01CC                  CLRQ    -(SP)
                   7E    7C  01CE                  CLRQ    -(SP)
                   00    DD  01D0                  PUSHL   #0
                BF AF    DF  01D2                  PUSHAL  b^tec$ctrlcast
                   7E    7C  01D5                  CLRQ    -(SP)
                   00    DD  01D7                  PUSHL   #0
    7E    0000'8F    3C  01D9                  MOVZWL  #io$_setmode!io$m_ctrlcast,-(SP)
    7E    00F8'CF    3C  01DE                  MOVZWL  w^ter_c_chan,-(SP)
                   00    DD  01E3                  PUSHL   #0
 00000000'GF       0C    FB  01E5                  CALLS   #12,G^SYS$QIOW
                         05  01EC  1503         rsb                                ; exit
```

```
                        01ED  1505 .sbttl  Terminal output waits
                        01ED  1506
                        01ED  1507 .enable lsb
                        01ED  1508
                        01ED  1509 tec$wait_done_lf:                       ; wait for all output (inc. <LF>'s)
        00FE'CF    03   90  01ED  1510          movb    #2!1, w^ter_o_force  ; say we're forcing <LF>'s also
0000'8F  00000000'EF   B1  01F2  1511          cmpw    ttoptr, #ttobuf      ; is the terminal output buffer empty?
              15       12  01FB  1512          bneq    tec$wait_done        ; nope
        00F0'CF        D5  01FD  1513          tstl    w^ter_o_cc           ; any pending, saved <LF>?
              0F       13  0201  1514          beql    tec$wait_done        ; nope
      7E   00'AB       B0  0203  1515          movw    b^outdne(r11), -(sp) ; yep, save current output done flag
        00F0'CF        D4  0207  1516          clrl    w^ter_o_cc           ; forget about saving the <LF>
            047F       30  020B  1517          bsbw    echo_lf              ;  because we want to (re-)buffer it
      00'AB    8E      B0  020E  1518          movw    (sp)+, b^outdne(r11) ; restore original output done flag
                       0212  1519 tec$wait_done:                           ; wait for all output to complete
        00FE'CF    01   88  0212  1520          bisb    #1, w^ter_o_force    ; say we're forcing terminal output
              25       10  0217  1521          bsbb    tec$wait             ;  and go do an initial wait
        00FE'CF        94  0219  1522          clrb    w^ter_o_force        ; now back to non-forced output
        00FF'CF        95  021D  1523          tstb    w^ter_o_pend         ; is there any in-progress output?
              31       19  0221  1524          blss    20$                  ; nope, go exit
         EC   AF       9F  0223  1525          pushab  b^tec$wait_done      ; yep, check again after waiting
                       0226  1526 10$:        $waitfr_s -                   ; wait for single event flag
                       0226  1527                    efn=#1                ;  on this event flag
                 01    DD  0226                    PUSHL   #1
    00000000'GF  01    FB  0228                    CALLS   #1,G^SYS$WAITFR
            FE90       30  022F  1528          bsbw    success_or_announce  ; announce any failure...
                       0232  1529          $clref_s -                    ; do an event flag clear
                       0232  1530                    efn=#1                ;  on this event flag
                 01    DD  0232                    PUSHL   #1
    00000000'GF  01    FB  0234                    CALLS   #1,G^SYS$CLREF
            FE84       30  023B  1531          bsbw    success_or_announce  ; announce any failure...
                       023E  1532 tec$wait:                               ; wait for output completion
0000'8F  00000000'EF   B1  023E  1533          cmpw    ttoptr, #ttobuf      ; is the buffer empty?
              0B       13  0247  1534          beql    20$                  ; yes, go exit
        00000000'EF    B5  0249  1535          tstw    ttoint               ; no, can we initiate output?
              D5       18  024F  1536          bgeq    10$                  ; output's already pending, go wait
            011D       30  0251  1537          bsbw    tec$output           ; else initiate whatever output we can
                       05  0254  1538 20$:     rsb                          ; exit
                       0255  1539
                       0255  1540 .disable lsb
```

```
                                0255  1542 .sbttl  Terminal output
                                0255  1543
                                0255  1544 check_esc_csi:                          ; check ESCape/CSI sequences
                 2A 00FE'CF   E8 0255  1545         blbs    w^ter_o_force, 40$     ; no processing if forcing output
                    52   50   7D 025A  1546         movq    r0, r2                 ; copy remaining count/pointer
                    54   30   9A 025D  1547         movzbl  #^a/0/, r4             ;  and guess at an ESCape sequence
                 9B 8F   63   91 0260  1548         cmpb    (r3), #27+128          ; is it really (8-bit) CSI?
                         08   13 0264  1549         beql    10$                    ; yep
                         2F   10 0266  1550         bsbb    60$                    ; get the next character
                 5B 8F   63   91 0268  1551         cmpb    (r3), #^a/[/           ; is it (7-bit) CSI?
                         06   12 026C  1552         bneq    30$                    ; nope
                 54   40 8F   9A 026E  1553 10$:     movzbl  #^a/@/, r4             ; set correct final for CSI
                         23   10 0272  1554 20$:     bsbb    60$                    ; get the next sequence character
                    20   63   91 0274  1555 30$:     cmpb    (r3), #32              ; control character?
                         0B   1F 0277  1556         blssu   40$                    ; yes, call it finished...
                 7F 8F   63   91 0279  1557         cmpb    (r3), #127             ; is it DEL or higher (also controls)?
                         05   1E 027D  1558         bgequ   40$                    ; yes, call it finished...
                    54   63   91 027F  1559         cmpb    (r3), r4               ; a final character to finish it?
                         EE   1F 0282  1560         blssu   20$                    ; not finished, loop for the next...
                    85   81   90 0284  1561 40$:     movb    (r1)+, (r5)+           ; copy over the ESCape/CSI character
                         50   D7 0287  1562         decl    r0                     ;  and remove it from the count
                         0B   13 0289  1563         beql    50$                    ; no more, just go exit
   50  0000'CF   1B   61   50   2F 028B  1564         movtuc  r0, (r1), #27, w^ter_o_table, r0, (r5) ; try move rest of data
                        65   0293
                         BF   1D 0294  1565         bvs     check_esc_csi          ; another ESCape/CSI, go process it
                         05   0296  1566 50$:     rsb                            ; exit
                              0297  1567
                         53   D6 0297  1568 60$:     incl    r3                     ; advance the remaining pointer
                         52   D7 0299  1569         decl    r2                     ;  and decrement the remaining count
                         20   12 029B  1570         bneq    70$                    ; more to look at, go exit
                    6E   E4 AF   9E 029D  1571         movab   40$, (sp)              ; no more, change return to finish up
                 0000'8F    50   B1 02A1  1572         cmpw    r0, #ttobfl            ; about to re-buffer the whole buffer?
                         15   1E 02A6  1573         bgequ   70$                    ; yep, don't, it's junk anyway...
                    6E   55   D0 02A8  1574         movl    r5, (sp)               ; else save dst ptr (clobber return)
00000000'EF         61   50   28 02AB  1575         movc    r0, (r1), ttobuf       ; re-buffer unfinished ESCape/CSI seq
              00000000'EF    63   9E 02B3  1576         movab   (r3), ttoptr           ;  setting a new buffering pointer
                              02BA  1577         ; How about CTRL/O and/or Binary Output?
                    55   8E   D0 02BA  1578         movl    (sp)+, r5              ; restore destination pointer
                         05   02BD  1579 70$:     rsb                            ; exit
                              02BE  1580
                              02BE  1581 tec$output_ast:                          ; terminal output done ast comes here
                        0000  02BE  1582         .word   ^m<>
                              02C0  1583
                 0000'8F    04 BC   B1 02C0  1584         cmpw    @4(ap), #ss$_controlc  ; completed under control/c?
                         14   12 02C6  1585         bneq    10$                    ; nope
                    00FD'CF    00   92 02C8  1586         mcomb   #0, w^ctrlo_flag       ; say control/o is now in effect
                              02CD  1587         $cancel_s -                    ; cancel all pending I/O
                              02CD  1588                 chan=w^ter_o_chan      ;  on the terminal output channel
                    7E   00F6'CF    3C 02CD                MOVZWL  w^ter_o_chan,-(SP)
              00000000'GF    01   FB 02D2                CALLS   #1,G^SYS$CANCEL
                        FDE6   30 02D9  1589         bsbw    success_or_announce    ; announce any failure...
                    00FF'CF    97 02DC  1590 10$:     decb    w^ter_o_pend           ; one less output request is pending
                         06   18 02E0  1591         bgeq    20$                    ; other(s) pending, really count it
                 00000000'EF    B6 02E2  1592         incw    ttoint                 ; else fake the count
                 00000000'EF    B7 02E8  1593 20$:     decw    ttoint                 ; say terminal output completed
                         03   18 02EE  1594         bgeq    30$                    ; something else is going on, return
                        0082   30 02F0  1595         bsbw    tec$output_more        ; else try for more output
```

```
                    04   02F3   1596 30$:    ret                                ; return
                         02F4   1597
                         02F4   1598 .enable lsb
                         02F4   1599
                53   D6  02F4   1600 10$:    incl    r3                         ; correct the buffer pointer
          08BE'CF    B5  02F6   1601          tstw    w^output_sys_rab+rab$w_rsz ; yep, anything in the record yet?
                02   13  02FA   1602          beql    30$                        ; nothing there
                15   10  02FC   1603 20$:    bsbb    50$                        ; else clear out the buffer
          00F2'CF   96  02FE   1604 30$:    incb    w^output_sys_vfc           ; indicate 1 more <LF> of prefix
                61   14  0302   1605          bgtr    90$                        ; go continue if < 128 <LF>'s
          00F2'CF   97  0304   1606          decb    w^output_sys_vfc           ; else back to only 127 <LF>'s
                F2   11  0308   1607          brb     20$                        ;  and go dump that record first
                         030A   1608
   00F3'CF    8D 8F   90  030A   1609 40$:    movb    #<1@7>!13, w^output_sys_vfc+1 ; indicate a <CR> as postfix
             65'AF   9F  0310   1610          pushab  b^90$                      ; set to loop after record output
08C4'CF    1E00'CF   9E  0313   1611 50$:    movab   w^output_sys_buf, w^output_sys_rab+rab$l_rbf ; reset pointer
                         031A   1612          $put -                             ; put a record
                         031A   1613                  rab=w^output_sys_rab       ;  to sys$output
          089C'CF   DF  031A             PUSHAL  w^output_sys_rab
 00000000'GF    01   FB  031E             CALLS   #$$.TMP1,G^SYS$PUT
010C'CF    08A8'CF   D0  0325   1614          movl    w^output_sys_rab+rab$l_stv, w^err_msgvec+8 ; save STV value
          08BE'CF   B4  032C   1615          clrw    w^output_sys_rab+rab$w_rsz ; always reset the record length
          00F2'CF   B4  0330   1616          clrw    w^output_sys_vfc           ;  and the print control information
             FD70   31  0334   1617          brw     success_else_die           ; check for success completion; exit
                         0337   1618
                0C   BB  0337   1619 60$:    pushr   #^m<r2,r3>                 ; save r2 & r3
             52   51   D0  0339   1620          movl    r1, r2                     ; move count over to here
 53    00000000'EF   9E  033C   1621          movab   ttobuf, r3                 ; get terminal output buffer pointer
                0D   83  0343   1622 70$:    cmpb    (r3)+, #13                 ; is this a <CR>?
                C2   13  0346   1623          beql    40$                        ; yes, go set postfixing
                0A   73  0348   1624          cmpb    -(r3), #10                 ; is this a <LF>?
                A7   13  034B   1625          beql    10$                        ; yes, go set prefixing
          0200 8F   08BE'CF   B1  034D   1626          cmpw    w^output_sys_rab+rab$w_rsz, #output_sys_siz ; room left?
                02   1F  0354   1627          blssu   80$                        ; yep
                BB   10  0356   1628          bsbb    50$                        ; nope, must dump this buffer
             08C4'DF   83  0358   1629 80$:    movb    (r3)+, @w^output_sys_rab+rab$l_rbf ; store a byte in buffer
                08C4'CF   D6  035D   1630          incl    w^output_sys_rab+rab$l_rbf ; bump the record buffer pointer
                08BE'CF   B6  0361   1631          incw    w^output_sys_rab+rab$w_rsz ;  and count in record length
                  DB 52   F5  0365   1632 90$:    sobgtr  r2, 70$                    ; loop if more to go...
                0C   BA  0368   1633          popr    #^m<r2,r3>                 ; restore r2 & r3
           00000000'EF   B7  036A   1634 100$:   decw    ttoint                     ; take away the buffer interlock
                05   0370   1635          rsb                                ; exit
                         0371   1636
                         0371   1637 tec$output:                               ; terminal output initiation
            00'AB   02   A8  0371   1638          bisw    #2, b^outdne(r11)          ; say some output was done
                         0375   1639 tec$output_more:                          ; more terminal output initiation
           00000000'EF   B6  0375   1640          incw    ttoint                     ; interlock the terminal output buffer
      50   00000000'EF   00'   A1  037B   1641          addw3   s^#io$_writevblk, ttomod, r0 ; form function code
51  00000000'EF   00000000'8F   C3  0383   1642          subl3   #ttobuf, ttoptr, r1        ; get byte count in output buffer
           00000000'EF   B4  038F   1643          clrw    ttomod                     ; reset the terminal output mode
 00000000'EF   00000000'EF   9E  0395   1644          movab   ttobuf, ttoptr             ;  and empty terminal output buffer
            04 50   00'   E1  03A0   1645          bbc     s^#io$v_canctrlo, r0, 110$ ; canceling control/o?
              00FD'CF   94  03A4   1646          clrb    w^ctrlo_flag               ; yes, so cancel it
                51   D5  03A8   1647 110$:    tstl    r1                         ; anything to output?
                BE   15  03AA   1648          bleq    100$                       ; nothing there...
                00F6'CF   B5  03AC   1649          tstw    w^ter_o_chan               ; do we have a real terminal?
                85   13  03B0   1650          beql    60$                        ; nope
```

```
                00FD'CF   95   03B2  1651          tstb    w^ctrlo_flag            ; yep, but is control/o in effect?
                      B2  12   03B6  1652          bneq    100$                    ; that it is, junk this output...
                      3C  BB   03B8  1653          pushr   #^m<r2,r3,r4,r5>        ; save r2 through r5
             52   0000'CF   7E   03BA  1654          movaq   w^ter_o_status1, r2     ; guess at terminal output iosb #1
             53   0000'CF   9E   03BF  1655          movab   w^ter_o_buf1, r3        ;  and terminal output buffer #1
                      62  B5   03C4  1656          tstw    (r2)                    ; is that iosb currently in use?
                      0A  12   03C6  1657          bneq    120$                    ; nope, so use it
             52   0008'CF   7E   03C8  1658          movaq   w^ter_o_status2, r2     ; else use terminal output iosb #2
             53   0200'CF   9E   03CD  1659          movab   w^ter_o_buf2, r3        ;  and terminal output buffer #2
                      0D  BB   03D2  1660  120$:     pushr   #^m<r0,r2,r3>           ; save function, iosb, and buffer
       1B  00000000'EF   51   2F   03D4  1661          movtuc  r1, ttobuf, #27, w^ter_o_table, r1, (r3) ; move data to buffer
          63   51   0000'CF         03DC
                      03  1C   03E1  1662          bvc     130$                    ; nothing special...
                  FE6F    30   03E3  1663          bsbw    check_esc_csi           ; else go check ESCape/CSI sequences
                      0D  BA   03E6  1664  130$:     popr    #^m<r0,r2,r3>           ; restore function, iosb, and buffer
             51   55   53   C3   03E8  1665          subl3   r3, r5, r1              ;  and (re-)calculate the byte count
                      51  15   03EC  1666          bleq    160$                    ; nothing left, just go exit...
                00FF'CF   96   03EE  1667          incb    w^ter_o_pend            ; one more output request is pending
                      06  15   03F2  1668          bleq    140$                    ; but only one, don't say busy
                 00000000'EF   B6   03F4  1669          incw    ttoint                  ; else say terminal output now busy
                      00  DD   03FA  1670  140$:     pushl   #0                      ; preset no carriage control next time
              10 00FE'CF   01   E0   03FC  1671          bbs     #1, w^ter_o_force, 150$ ; are <LF>'s being forced out?
                 0A   FF A341   91   0402  1672          cmpb    -1(r3)[r1], #10         ; does the buffer end with a <LF>?
                      09  12   0407  1673          bneq    150$                    ; nope
                      51  D7   0409  1674          decl    r1                      ; yep, remove the <LF> from the count
              6E   008A0000 8F   D0   040B  1675          movl    #<<1@7>!10>@16, (sp)    ;  and set <LF> prefix for next time
                           0412  1676  150$:     $qio_s -                        ; start the terminal output write
                           0412  1677                 efn=#1, -                ;  using this event flag
                           0412  1678                 chan=w^ter_o_chan, -     ;  using the terminal's channel
                           0412  1679                 func=r0, -               ;  using the correct function
                           0412  1680                 iosb=(r2), -             ;  using this iosb
                           0412  1681                 astadr=tec$output_ast, - ;  catch the output completion
                           0412  1682                 astprm=r2, -             ;  passing the iosb pointer
                           0412  1683                 p1=(r3), -               ;  from this output buffer
                           0412  1684                 p2=r1, -                 ;   with this byte count
                           0412  1685                 p4=w^ter_o_cc            ;  using this carriage control
                      7E   7C   0412                 CLRQ    -(SP)
                00F0'CF   DD   0414                 PUSHL   w^ter_o_cc
                      00  DD   0418                 PUSHL   #0
                      51  DD   041A                 PUSHL   r1
                      63  DF   041C                 PUSHAL  (r3)
                      52  DD   041E                 PUSHL   r2
                  FE9A CF   DF   0420                 PUSHAL  tec$output_ast
                      62  7F   0424                 PUSHAQ  (r2)
                      7E  50   3C   0426                 MOVZWL  r0,-(SP)
             7E   00F6'CF   3C   0429                 MOVZWL  w^ter_o_chan,-(SP)
                      01  DD   042E                 PUSHL   #1
       00000000'GF    0C   FB   0430                 CALLS   #12,G^SYS$QIO
          00F0'CF    8E   D0   0437  1686                 movl    (sp)+, w^ter_o_cc       ; set carriage control for next time
                  FC68    30   043C  1687                 bsbw    success_else_die        ; check for success completion
                      3C  BA   043F  1688  160$:     popr    #^m<r2,r3,r4,r5>        ; restore r2 through r5
                00000000'EF   B7   0441  1689                 decw    ttoint                  ; take away the buffer interlock
                      05  05   0447  1690                 rsb                     ; exit
                           0448  1691
                           0448  1692 .disable lsb
```

```
                        0448  1694 .sbttl  Terminal input
                        0448  1695
                        0448  1696 .enable lsb
                        0448  1697
        00F4'CF   B5    0448  1698 10$:     tstw    w^ter_i_chan            ; have a real terminal?
              77  12    044C  1699          bneq    110$                    ; yep
              77  D4    044E  1700          clrl    -(r7)                   ; ensure the input buffer is empty
    56  07A0'CF  DE    0450  1701          moval   w^input_sys_fab, r6     ; get the sys$input fab pointer
            0809  30    0455  1702          bsbw    getbyt                  ; get the next byte
00000000'8F   50  D1    0458  1703          cmpl    r0, #rms$_eof           ; end-of-file?
              30  13    045F  1704          beql    50$                     ; yep
            FC73  30    0461  1705          bsbw    success_or_abrt         ; else check for success completion
  0F 58   00'     E1    0464  1706          bbc     s^#io$v_cvtlow, r8, 20$ ; converting lower case?
  61 8F     51    91    0468  1707          cmpb    r1, #^a7A/+32           ; yep, is it lower case?
              09  1F    046C  1708          blssu   20$                     ; not lower case
  7A 8F     51    91    046E  1709          cmpb    r1, #^a/Z/+32           ; might be...
              03  1A    0472  1710          bgtru   20$                     ; but it isn't
        51    20  8A    0474  1711          bicb    #32, r1                 ; make lower case into upper case
        6A    51  9A    0477  1712 20$:     movzbl  r1, i_r0(r10)           ; copy character to here
      00EC'CF   D4    047A  1713          clrl    w^ctrlz_cnt             ;  and clear control/z counter
  0C 58   00'     E0    047E  1714 30$:     bbs     s^#io$v_noecho, r8, 40$ ; skip echo if not echoing
  7F 8F     6A    91    0482  1715          cmpb    i_r0(r10), #127         ; is the terminator a delete?
              06  13    0486  1716          beql    40$                     ; yep, delete's are echoed elsewhere
        56    6A  9A    0488  1717          movzbl  i_r0(r10), r6           ; get the character to echo
            0202  30    048B  1718          bsbw    echo_char               ;  and go fully echo it
            0103  31    048E  1719 40$:     brw     190$                    ; go check out the input
                        0491  1720
        6A    1A  9A    0491  1721 50$:     movzbl  #^a/Z/-64, i_r0(r10)    ; set a control/z
    E4 00EC'CF    03  F2    0494  1722          aoblss  #3, w^ctrlz_cnt, 30$    ; continue if not third control/z
20 AA   00000000'EF   3E    049A  1723 60$:     movaw   texit, i_pc(r10)        ; set for exiting from teco
              17  11    04A2  1724          brb     90$                     ;  and go exit
                        04A4  1725
    50  04 A6    66  A9    04A4  1726 70$:     bisw3   (r6), 4(r6), r0         ; any character(s) or terminator(s)?
              73  12    04A9  1727          bneq    130$                    ; yep, so go use them...
        6A    01  AE    04AB  1728          mnegw   #1, i_r0(r10)           ; return a -1 for no input
              77  D4    04AE  1729          clrl    -(r7)                   ; ensure the input buffer is empty
            00F4  31    04B0  1730          brw     200$                    ;  and go exit
                        04B3  1731
20 AA   00000000'EF   3E    04B3  1732 80$:     movaw   teco, i_pc(r10)         ; set for restarting teco
            0020'CF   D4    04BB  1733 90$:     clrl    w^ter_i                 ; ensure the input buffer is empty
                  05  04BF  1734          rsb                             ;  and exit
                        04C0  1735
          5B 50   E8    04C0  1736 100$:    blbs    r0, 130$                ; call a random success normal...
              76  B5    04C3  1737          tstw    -(r6)                   ; correct the iosb pointer
      51  00E0'CF   D0    04C5  1738 110$:    movl    w^ter_i_nor_trm_ptr, r1 ; guess at the normal terminator mask
        05 58   00'     E1    04CA  1739          bbc     s^#io$v_timed, r8, 120$ ; checking for type ahead?
      51  0228'CF   7E    04CE  1740          movaq   w^ter_i_any_trm, r1     ; use the anything terminator mask
                        04D3  1741 120$:    $qiow_s -                       ; do a terminal input read
                        04D3  1742                      chan=w^ter_i_chan, -    ;  using the terminal input channel
                        04D3  1743                      func=r8, -              ;  using the correct function
                        04D3  1744                      iosb=(r6), -            ;  put i/o status here on completion
                        04D3  1745                      p1=a(r7), -             ;  using the terminal input buffer
                        04D3  1746                      p2=r9, -                ;   with the correct length
                        04D3  1747                      p3=#0, -                ;  using an immediate timeout value
                        04D3  1748                      p4=r1                   ;  using correct terminator mask
              7E  7C    04D3                            CLRQ    -(SP)
              51  DD    04D5                            PUSHL   r1
```

```
                00   DD  04D7                        PUSHL   #0
                59   DD  04D9                        PUSHL   r9
             00 B7   DF  04DB                        PUSHAL  @(r7)
                7E   7C  04DE                        CLRQ    -(SP)
                66   7F  04E0                        PUSHAQ  (r6)
             7E 58   3C  04E2                        MOVZWL  r8,-(SP)
          7E 00F4'CF 3C  04E5                        MOVZWL  w^ter_i_chan,-(SP)
                00   DD  04EA                        PUSHL   #0
  00000000'GF   0C   FB  04EC                        CALLS   #12,G^SYS$QIOW
              FBE1   30  04F3  1749          bsbw    success_or_abrt         ; check for success completion
           50   86   3C  04F6  1750          movzwl  (r6)+, r0               ; get the completion code
      0000'8F   50   B1  04F9  1751          cmpw    r0, #ss$_timeout        ; a timed out operation?
                A4   13  04FE  1752          beql    70$                     ; yes, go return a -1 to user
      0000'8F   50   B1  0500  1753          cmpw    r0, #ss$_controlc       ; was control/c typed?
                B9   12  0505  1754          bneq    100$                    ; none of the above...
  00FC'CF   00FC'CF  92  0507  1755          mcomb   w^ctrlc_flag, w^ctrlc_flag ; flip the control/c flop
                8A   13  050E  1756          beql    60$                     ; it flopped, exit from teco
         9F 6A   0F   E1  0510  1757          bbc     #tec$v_et$cc, i_r0(r10), 80$ ; allowing return of control/c?
           66   01   B0  0514  1758          movw    #1, (r6)                ; set data count = 1
             02 A6   D4  0517  1759          clrl    2(r6)                   ;  and no terminator(s)
         00 B7   03   90  051A  1760          movb    #^a/C/-64, @(r7)        ;   then buffer a control/c
           77   86   3C  051E  1761 130$:     movzwl  (r6)+, -(r7)            ; set count of characters obtained
                0C   13  0521  1762          beql    140$                    ; none
           00EC'CF   D4  0523  1763          clrl    w^ctrlz_cnt             ; some, reset control/z counter
         04 58   00'  E0  0527  1764          bbs     s^#io$v_noecho, r8, 140$ ; did they echo?
         00'AB   02   A8  052B  1765          bisw    #2, b^outdne(r11)       ; echoed, so say so
         50   02 A6   3C  052F  1766 140$:     movzwl  2(r6), r0               ; get the terminator's length
                51   13  0533  1767          beql    180$                    ; no length, so no terminator(s)
      51   67   87   C1  0535  1768          addl3   (r7)+, (r7), r1         ; else point just beyond the data
           56   61   9A  0539  1769          movzbl  (r1), r6                ; get the terminator itself
           77   50   C0  053C  1770          addl    r0, -(r7)               ; now count the terminator(s)
           1A   56   91  053F  1771          cmpb    r6, #^a/Z/-64           ; control/z?
                09   12  0542  1772          bneq    150$                    ; no
    07 00EC'CF   03   F2  0544  1773          aoblss  #3, w^ctrlz_cnt, 160$   ; continue if not third control/z
              FF4D   31  054A  1774          brw     60$                     ; else go exit from teco
                         054D  1775
           00EC'CF   D4  054D  1776 150$:     clrl    w^ctrlz_cnt             ; no, reset control/z counter
           0D   56   91  0551  1777 160$:     cmpb    r6, #13                 ; carriage return?
                13   12  0554  1778          bneq    170$                    ; no
                67   D6  0556  1779          incl    (r7)                    ; yes, count 1 more in buffer
      57   67   87   C1  0558  1780          addl3   (r7)+, (r7), r7         ; find position for line feed
           77   0A   90  055C  1781          movb    #10, -(r7)              ;  and store it also
        23 58   00'  E0  055F  1782          bbs     s^#io$v_noecho, r8, 180$ ; skip echo if not echoing
              012E   30  0563  1783          bsbw    echo_buffer             ; yes, echo the carriage return
           56   0A   9A  0566  1784          movzbl  #10, r6                 ;  and set to echo a line feed
        19 58   00'  E0  0569  1785 170$:     bbs     s^#io$v_noecho, r8, 180$ ; skip echo if not echoing
        7F 8F   56   91  056D  1786          cmpb    r6, #127                ; is the terminator a delete?
                13   13  0571  1787          beql    180$                    ; yep, delete's are echoed elsewhere
              011A   30  0573  1788          bsbw    echo_char               ; else fully echo the terminator
                         0576  1789 tec$input:                              ; terminal input
                         0576  1790          bs      s^#io$v_canctrlo, ctlofg ; do a control/o cancel
00 00000000'EF   00'  E3  0576                bbcs    s^#io$v_canctrlo, ctlofg, 30005$
                         057E        30005$:
         04 6A   01   E5  057E  1791          bbcc    #1, i_r0(r10), 180$     ; main prompt call?
           00EC'CF   D4  0582  1792          clrl    w^ctrlz_cnt             ; yes, reset control/z counter
      57   0020'CF   7E  0586  1793 180$:     movaq   w^ter_i, r7             ; get terminal input buffer desc
                87   D7  058B  1794          decl    (r7)+                   ; remove one character from buffer
```

```
                    35   19   058D  1795                 blss    230$                            ; nothing left, go get some more
                6A  97   9A   058F  1796                 movzbl  @(r7)+, i_r0(r10)               ; else get the character
                    77   D6   0592  1797                 incl    -(r7)                           ;  and bump the buffer pointer
                50  6A   9A   0594  1798 190$:           movzbl  i_r0(r10), r0                   ; get character about to be returned
                  FA66'  30   0597  1799                 bsbw    tecoexelbr                      ;  and go check it
                    0B   13   059A  1800                 beql    200$                            ; there was nothing there...
                 12 50   E9   059C  1801                 blbc    r0, 220$                        ; there's some kind of error...
20 AA   00000000'EF      3E   059F  1802                 movaw   tecocr, i_pc(r10)               ; set to re-start TECO w/ <CR><LF>
              00FC'CF    94   05A7  1803 200$:           clrb    w^ctrlc_flag                    ; turn off control/c exit flag if here
                    05   05AB  1804                 rsb                                     ;  and exit
                         05AC  1805
                6A  1B   9A   05AC  1806 210$:           movzbl  #27, i_r0(r10)                  ; set final initial command <ESC>
                    F6   11   05AF  1807                 brb     200$                            ;  and go exit with it...
                         05B1  1808
          010C'CF   51   D0   05B1  1809 220$:           movl    r1, w^err_msgvec+8              ; save any STV value
20 AA   00000000'EF      3E   05B6  1810                 movaw   ioerrs, i_pc(r10)               ; set exit to teco's error processor
                 10 AA   D4   05BE  1811                 clrl    i_r4(r10)                       ;  with R4 = 0 for filename exists
                  FB3A   31   05C1  1812                 brw     success_or_err                  ; now go die with the error...
                         05C4  1813
             56  00'AB   3C   05C4  1814 230$:           movzwl  b^indir(r11), r6                ; is an indirect command file active?
                    0A   13   05C8  1815                 beql    240$                            ; nope
                02  56   D1   05CA  1816                 cmpl    r6, #2                          ; really?
                    54   1A   05CD  1817                 bgtru   300$                            ; a real indirect command file...
                00'AB   B7   05CF  1818                 decw    b^indir(r11)                    ; "funny"(1) or initial command(2)?
                    D8   12   05D2  1819                 bneq    210$                            ; initial command, it's now "funny"(1)
                  FC16   30   05D4  1820 240$:           bsbw    tec$wait_done_lf                ; wait for terminal output to complete
                  0C73   30   05D7  1821                 bsbw    not_exiting                     ; not exiting if doing terminal input
  00E0'CF   0200'CF      7E   05DA  1822                 movaq   w^ter_i_nor7_trm, w^ter_i_nor_trm_ptr ; mask to 7-bit normal
     07 00'AB    0C      E1   05E1  1823                 bbc     #tec$v_et$8bt, b^etype(r11), 250$ ; is it an 8-bit terminal?
  00E0'CF   0028'CF      7E   05E6  1824                 movaq   w^ter_i_nor8_trm, w^ter_i_nor_trm_ptr ; mask to 8-bit normal
        67  0400'CF      9E   05ED  1825 250$:           movab   w^ter_i_buf, (r7)               ; reset terminal input buffer pointer
        56  0018'CF      7E   05F2  1826                 movaq   w^ter_i_status, r6              ; get pointer to terminal input iosb
                         3C   05F7  1827                 movzwl  #io$_readvblk! -                ; set function as read virtual,
                              05F8  1828                         io$m_nofiltr! -                 ;                 no filtering,
                              05F8  1829                         io$m_trmnoecho! -               ;                 no terminator echo,
 58   0000'8F                 05F8  1830                         io$m_dsablmbx, r8               ;                 keep any mailbox off
 59   01FF 8F   3C   05FC  1831                 movzwl  #ter_i_siz-1, r9                ; set terminal input buffer size
04 00'AB   02   E0   0601  1832                 bbs     #tec$v_et$lc, b^etype(r11), 260$ ; allowing lower case?
                         0606  1833                 bs      s^#io$v_cvtlow, r8              ; no, so don't
   00 58   00'  E3   0606                          bbcs    s^#io$v_cvtlow, r8, 30006$
                         060A                  30006$:
   04 6A   03   E1   060A  1834 260$:           bbc     #tec$v_et$nch, i_r0(r10), 270$ ; echoing?
                         060E  1835                 bs      s^#io$v_noecho, r8              ; no, so don't
   00 58   00'  E3   060E                          bbcs    s^#io$v_noecho, r8, 30007$
                         0612                  30007$:
   04 6A   05   E1   0612  1836 270$:           bbc     #tec$v_et$cke, i_r0(r10), 280$ ; checking type ahead?
                         0616  1837                 bs      s^#io$v_timed, r8               ; yes, so do
   00 58   00'  E3   0616                          bbcs    s^#io$v_timed, r8, 30008$
                         061A                  30008$:
      03 6A   E9   061A  1838 280$:           blbc    i_r0(r10), 290$                 ; single character input mode?
        59  01   D0   061D  1839                 movl    #1, r9                          ; yes, so buffer size is 1 character
          FE25   31   0620  1840 290$:           brw     10$                             ; long branch for real terminal input
                         0623  1841
              77   D4   0623  1842 300$:           clrl    -(r7)                           ; ensure the input buffer is empty
 57   0378'CF      DE   0625  1843                 moval   w^indir_cmd_fab, r7             ; address the "ei" fab
   58   50 A7   D0   062A  1844                 movl    fab$l_tecsts(r7), r8            ; save original "pre-fetched" char/flag
                         062E  1845                 bs      #fab$v_tecnxt, fab$l_tecsts(r7) ; guess at "pre-fetch" working
```

```
                50 A7    20    88  062E                    bisb    s^#1@fab$v_tecnxt, fab$l_tecsts(r7)
                   56    66    D0  0632  1846              movl    (r6), r6                ; get "ei" fab pointer
                         0C    13  0635  1847              beql    310$                    ; none?
                      0627     30  0637  1848              bsbw    getbyt                  ; get the next character
00000000'8F             50    D1  063A  1849              cmpl    r0, #rms$_normal        ; normal completion?
                         0A    13  0641  1850              beql    320$                    ; yep, character was "pre-fetched"
                               0643  1851 310$:            bc      #fab$v_tecnxt, fab$l_tecsts(r7) ; say nothing "pre-fetched"
                50 A7    20    8A  0643                    bicb    s^#1@fab$v_tecnxt, fab$l_tecsts(r7)
                      071F     30  0647  1852              bsbw    reset_indir             ; close the indirect file
                   00'AB     B6  064A  1853              incw    b^indir(r11)            ;  then mark as "funny"(1)
                53 A7    51    90  064D  1854 320$:       movb    r1, fab$l_tecsts+3(r7)  ; save (possible) "pre-fetch" character
                03 58    05    E0  0651  1855              bbs     #fab$v_tecnxt, r8, 330$ ; originally have "pre-fetched" data?
                      FF2E     31  0655  1856              brw     180$                    ; nope, go loop to try again...
                               0658  1857
6A   58    08    18    EF  0658  1858 330$:       extzv   #24, #8, r8, i_r0(r10)  ; set the "pre-fetched" character
                      FF34     31  065D  1859              brw     190$                    ;  and go check it out...
                               0660  1860
                               0660  1861 .disable lsb
```

```
                        0660  1863 .sbttl  Echoing, etc.
                        0660  1864
                   0FFC 0660  1865 .entry  tec$out_ascid, ^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11> ; output by desc
                        0662  1866
5B  00000000'EF  3E     0662  1867         movaw   r5set, r11              ; (re-)point to teco's read/write area
      57  04 BC  7D     0669  1868         movq    @4(ap), r7              ; get string's descriptor
          57 57  3C     066D  1869         movzwl  r7, r7                  ;  and get real string length
             08  13     0670  1870         beql    20$                     ; null string, just do the <CR><LF>
          56 88  9A     0672  1871 10$:     movzbl  (r8)+, r6               ; fetch next string character
             1D  10     0675  1872         bsbb    echo_buffer             ;  and output buffer it
            F8 57  F5   0677  1873         sobgtr  r7, 10$                 ;   then loop for more...
             0C  10     067A  1874 20$:     bsbb    echo_crlf               ; now go do the <CR><LF>
          50 01  D0     067C  1875         movl    #1, r0                  ; set the success return status
          00FD'CF  95   067F  1876         tstb    w^ctrlo_flag            ; is control/o in effect?
             02  13     0683  1877         beql    30$                     ; nope
             50  D4     0685  1878         clrl    r0                      ; yep, change return status to failure
                 04     0687  1879 30$:     ret                             ; return
                        0688  1880
                        0688  1881 echo_crlf:                             ; buffer and dump a <CR><LF>
          56 0D  9A     0688  1882         movzbl  #13, r6                 ; set a <CR>
             07  10     068B  1883         bsbb    echo_buffer             ;  and go output it
                        068D  1884 echo_lf:                              ; buffer and dump a <LF>
          56 0A  9A     068D  1885         movzbl  #10, r6                 ; set a <LF>
                        0690  1886 echo_char:                            ; buffer and dump a character
          07F0'CF  9F   0690  1887         pushab  w^echo_dump             ; dump buffer after buffering
                        0694  1888 echo_buffer:                          ; buffer a echo character
       25 56 07  E0     0694  1889         bbs     #7, r6, 20$             ; a "negative" character?
             20  56  91 0698  1890         cmpb    r6, #32                 ; a control character?
                 5F  1E 069B  1891         bgequ   echo_byte               ; nope
                 0D  56 91 069D  1892      cmpb    r6, #13                 ; carriage return or higher
                     52 1A 06A0  1893      bgtru   40$                     ; it's higher
                 07  56 91 06A2  1894      cmpb    r6, #7                  ; bell or lower?
                     55 1A 06A5  1895      bgtru   echo_byte               ; it's bs, tab, lf, vt, ff, or cr
                     02 1F 06A7  1896      blssu   10$                     ; lower than bell
                     51 10 06A9  1897      bsbb    echo_byte               ; it's bell, ring the bell first
                     56 DD 06AB  1898 10$: pushl   r6                      ; save character
                  56  5E 8F 9A   06AD  1899 movzbl  #^a/^/, r6             ; prefix with an "^"
                     49 10 06B1  1900      bsbb    echo_byte               ; go output the "^"
56  8E  00000040 8F  C9 06B3  1901         bisl3   #64, (sp)+, r6          ; restore character making it visible
                     3F 11 06BB  1902      brb     echo_byte               ;  and go output it
                        06BD  1903
                51  5B5D 8F  3C   06BD  1904 20$:  movzwl  #<^a/[/@8>+^a/]/, r1 ; guess at hex digits trailing/leading
   50  FFFFFF00'EF 46  32 06C2  1905      cvtwl   cnv8bt-256[r6], r0      ; get the conversion character pair
                     0A 19 06CA  1906      blss    30$                     ;  it's hex digits (<15> = 1)
             2B 00'AB  0C  E0   06CC  1907 bbs     #tec$v_et$8bt, b^etype(r11), echo_byte ; 8-bit terminal?
                51  3C3E 8F  3C   06D1  1908 movzwl  #<^a/<7@8>+^a/>/, r1   ; set compose sequence trailing/leading
                     7E 50 7D 06D6  1909 30$: movq    r0, -(sp)              ; save our special characters
                  56  05 AE 9A   06D9  1910 movzbl  5(sp), r6               ; get the leading signal
                     1D 10 06DD  1911      bsbb    echo_byte               ;  and go output it
                     56 6E 9A 06DF  1912   movzbl  (sp), r6                ; get the first of the character pair
                     18 10 06E2  1913      bsbb    echo_byte               ;  and go output it
   56  6E  07  08  EF   06E4  1914 extzv   #8, #7, (sp), r6                ; get the second of pair (w/ <7> = 0)
                     11 10 06E9  1915      bsbb    echo_byte               ;  and go output it
                  56  04 AE 9A   06EB  1916 movzbl  4(sp), r6               ; get the trailing signal
                     5E 08 C0 06EF  1917   addl    #8, sp                  ; pop our special characters from stack
                     08 11 06F2  1918      brb     echo_byte               ; go output trailing signal & exit
                        06F4  1919
```

```
                 1B   56   91  06F4  1920 40$:    cmpb    r6, #27                 ; escape?
                      B2   12  06F7  1921         bneq    10$                     ; nope, use an uparrow
                   56   24   9A  06F9  1922         movzbl  #^a/$/, r6              ; yep, use "$"
                             06FC  1923 echo_byte:                                ; buffer a echo byte
      0000'8F    00000000'EF   B1  06FC  1924         cmpw    ttoptr, #ttobuf         ; is the terminal output buffer empty?
                        2E   13  0705  1925         beql    30$                     ; yes, always o.k. to buffer, etc.
00000000'EF      00000000'EF   B1  0707  1926         cmpw    ctlofg, ttomod          ; mode correct?
                        0B   13  0712  1927         beql    10$                     ; it's totally correct
      00000000'EF   0000'8F   B1  0714  1928         cmpw    #io$m_canctrlo, ttomod  ; mode correct (control/o cancel)?
                        0B   12  071D  1929         bneq    20$                     ; nope, must be really changing modes
      0000'8F    00000000'EF   B1  071F  1930 10$:    cmpw    ttoptr, #ttobuf+ttobfl  ; any room to buffer a character?
                        0B   1F  0728  1931         blssu   30$                     ; room left, just go buffer character
                00000000'EF   B5  072A  1932 20$:    tstw    ttoint                  ; is output currently in progress?
                        32   18  0730  1933         bgeq    40$                     ; yes, we must wait for it to finish
                      FC3C   30  0732  1934         bsbw    tec$output              ; no, start it going...
                00000000'EF   B6  0735  1935 30$:    incw    ttoint                  ; interlock the terminal output buffer
00000000'EF     00000000'EF   A8  073B  1936         bisw    ctlofg, ttomod          ; .OR. any control/o cancel into mode
                00000000'EF   B4  0746  1937         clrw    ctlofg                  ; clear out control/o cancel request
                             074C  1938         bc      #tec$v_et$cco, b^etype(r11) ;  and say we did it
                   00'AB   10   8A  074C                  bicb    s^#1@tec$v_et$cco, b^etype(r11)
             00000000'FF   56   90  0750  1939         movb    r6, @ttoptr             ; store the character in the buffer
                00000000'EF   B6  0757  1940         incw    ttoptr                  ;  then bump the buffer pointer
                00000000'EF   B7  075D  1941         decw    ttoint                  ;   and take away the buffer interlock
                             05  0763  1942         rsb                             ; exit
                             0764  1943
                      FAD7   30  0764  1944 40$:    bsbw    tec$wait                ; go wait for output to finish
                        93   11  0767  1945         brb     echo_byte               ;  then try, try again...
```

```
                        0769  1947 .sbttl  Process line/character deletion echoing
                        0769  1948
                        0769  1949 .enable lsb
                        0769  1950
                        0769  1951 tec$delln:                              ; control/u handler
      1D 00'AB    01    E0  0769  1952          bbs     #tec$v_et$crt, b^etype(r11), 40$ ; scope mode?
           56     0D    9A  076E  1953 10$:     movzbl  #13, r6             ; do tty compatible, set <cr>
                FF20    30  0771  1954          bsbw    echo_buffer         ;  and output it
           56     0A    9A  0774  1955          movzbl  #10, r6             ; set <lf>
                FF16    31  0777  1956 20$:     brw     echo_char           ; go fully echo the character
                        077A  1957
        18 AA     02    C2  077A  1958 30$:     subl    #2, i_sp(r10)       ; 'push' the sp stack
     18 BA    20 AA    B0  077E  1959          movw    i_pc(r10), ai_sp(r10) ;  and move return onto it
20 AA    00000000'EF    3E  0783  1960          movaw   prtlin, i_pc(r10)   ; set line re-print as the new exit
   57    00000000'EF    3C  078B  1961 40$:     movzwl  crterl, r7          ; get erase line sequence base
                 49    10  0792  1962          bsbb    70$                 ;  and go do it
   57    00000000'EF    3C  0794  1963          movzwl  crtcup, r7          ; get cursor up sequence base
                 40    10  079B  1964          bsbb    70$                 ;  and go do that also
                 CF    11  079D  1965          brb     10$                 ; now go do a <cr><lf>
                        079F  1966
                        079F  1967 tec$delch:                              ; delete handler
           56    6A    9A  079F  1968          movzbl  i_r0(r10), r6       ; get the deleted character
      D0 00'AB    01    E1  07A2  1969          bbc     #tec$v_et$crt, b^etype(r11), 20$ ; scope mode?
           58    01    D0  07A7  1970          movl    #1, r8              ; set count for 1 erase sequence
        59 56    07    E0  07AA  1971          bbs     #7, r6, 100$        ; a "negative" character?
           20    56    91  07AE  1972          cmpb    r6, #32             ; normal graphic?
                 65    1E  07B1  1973          bgequ   120$                ; yes, 1 sequence
           1B    56    91  07B3  1974          cmpb    r6, #27             ; escape?
                 60    13  07B6  1975          beql    120$                ; yes, 1 sequence
                 58    D6  07B8  1976          incl    r8                  ; no, set count for 2 sequences
           0D    56    91  07BA  1977          cmpb    r6, #13             ; is it carriage return or higher?
                 59    1A  07BD  1978          bgtru   120$                ; higher, do 2 sequences
                 B9    13  07BF  1979          beql    30$                 ; it's carriage return, re-print line
           07    56    91  07C1  1980          cmpb    r6, #7              ; is it bell or lower?
                 52    1B  07C4  1981          blequ   120$                ; that it is, do 2 sequences
           0A    56    91  07C6  1982          cmpb    r6, #10             ; check around line feed
                 AF    1F  07C9  1983          blssu   30$                 ; it's backspace or tab, re-print line
                 09    13  07CB  1984          beql    60$                 ; it's line feed, do 1 cursor up
           0C    56    91  07CD  1985          cmpb    r6, #12             ; check around form feed
                 02    1F  07D0  1986          blssu   50$                 ; it's vertical tab, do 2 cursor ups
                 00    10  07D2  1987          bsbb    50$                 ; do 2 cursor ups, fall to do 2 more
                 00    10  07D4  1988 50$:     bsbb    60$                 ; do 1 cursor up, fall to do another
   57    00000000'EF    3C  07D6  1989 60$:     movzwl  crtcup, r7          ; get cursor up sequence base
    57    00'AB    A0  07DD  1990 70$:     addw    b^crtype(r11), r7   ; find correct sequence pointer
       57    67    3C  07E1  1991          movzwl  (r7), r7            ;  and get pointer to count, string
       58    87    9A  07E4  1992          movzbl  (r7)+, r8           ; pickup the count, correct pointer
       56    87    9A  07E7  1993 80$:     movzbl  (r7)+, r6           ; get a character
          FF0F    30  07EA  1994          bsbw    echo_byte           ;  and output it
        F7 58     F5  07ED  1995          sobgtr  r8, 80$             ;  then loop for more...
                        07F0  1996 echo_dump:                              ; dump the buffered echoing
      00000000'EF    B5  07F0  1997          tstw    ttoint              ; is terminal output buffer free?
                 0E    18  07F6  1998          bgeq    90$                 ; nope, go exit
0000'8F 00000000'EF    B1  07F8  1999          cmpw    ttoptr, #ttobuf     ; yep, anything buffered therein?
                 03    13  0801  2000          beql    90$                 ; nothing there, go exit
              FB6B    30  0803  2001          bsbw    tec$output          ; else start that output going
                        05  0806  2002 90$:     rsb                         ; exit
                        0807  2003
```

```
                FFFFFF00'EF46   B5  0807  2004 100$:   tstw    cnv8bt-256[r6]          ; check the conversion table entry
                           05   19  080E  2005          blss    110$                    ; it's hex digits (<15> = 1)
                03 00'AB   0C   E0  0810  2006          bbs     #tec$v_et$8bt, b^etype(r11), 120$ ; 8-bit terminal?
                        58 04   D0  0815  2007 110$:    movl    #4, r8                  ; we must do 4 erase sequences
                      FA23      30  0818  2008 120$:    bsbw    tec$wait                ; wait for all output to be queued
                                    081B  2009          $qiow_s -                       ; queue an i/o request with wait
                                    081B  2010                  chan = w^ter_o_chan, -  ;  on the terminal write channel
                                    081B  2011                  func = s^#io$_writevblk, - ;  with a write function code
                                    081B  2012                  iosb = w^ter_o_pos      ;  so we can get horizontal position
                           7E   7C  081B                       CLRQ    -(SP)
                           7E   7C  081D                       CLRQ    -(SP)
                           00   DD  081F                       PUSHL   #0
                           00   DD  0821                       PUSHL   #0
                           7E   7C  0823                       CLRQ    -(SP)
                    0010'CF     7F  0825                       PUSHAQ  w^ter_o_pos
                    7E   00'    3C  0829                       MOVZWL  s^#io$_writevblk,-(SP)
                 7E 00F6'CF     3C  082C                       MOVZWL  w^ter_o_chan,-(SP)
                           00   DD  0831                       PUSHL   #0
             00000000'GF   0C   FB  0833                       CALLS   #12,G^SYS$QIOW
                50 0016'CF     9A  083A  2013          movzbl  w^ter_o_pos+6, r0       ; extract the 1 based position
                           1A   13  083F  2014          beql    130$                    ; a zero, no position, skip it
                        58 50   D1  0841  2015          cmpl    r0, r8                  ; are we looking at a line wrap?
                           15   14  0844  2016          bgtr    130$                    ; nope
             57 00000000'EF     3C  0846  2017          movzwl  crterl, r7              ; get erase line sequence base
                           1E   10  084D  2018          bsbb    160$                    ;  and go to do it
             57 00000000'EF     3C  084F  2019          movzwl  crtcup, r7              ; get cursor up sequence base
                           15   10  0856  2020          bsbb    160$                    ;  and go to do that also
                      FF1F      31  0858  2021          brw     30$                     ; now go really do it...
                                    085B  2022
                     02 58      D1  085B  2023 130$:    cmpl    r8, #2                  ; how many sequences do we need?
                           06   19  085E  2024          blss    150$                    ; 1
                           02   13  0860  2025          beql    140$                    ; 2
                           00   10  0862  2026          bsbb    140$                    ; we need 4 sequences
                           00   10  0864  2027 140$:    bsbb    150$                    ; we need 2 sequences
                57 00000000'EF     3C  0866  2028 150$:    movzwl  crterc, r7          ; get erase character sequence base
                      FF6D      31  086D  2029 160$:    brw     70$                     ; go do the sequence, then exit
                                    0870  2030
                                    0870  2031 .disable lsb
```

```
                                0870  2033 .sbttl  Page backwards
                                0870  2034
                                0870  2035 .enable lsb
                                0870  2036
                                0870  2037 10$:    err     NFI, <"No file for input">
                          F8D2   30  0870                    bsbw    err
                           00000351' 0873                     .long $$$$$$
                                5879  0351                     .word  $$$$$$
72 6F 66 20 65 6C 69 66 20 6F 4E 00' 0353                     .ascic "No file for input"
                    74 75 70 6E 69 20  035F
                                11    0353
                                0877  2038
                                0877  2039 20$:    err     NFO, <"No file for output">
                          F8CB   30  0877                    bsbw    err
                           00000365' 087A                     .long $$$$$$
                                587F  0365                     .word  $$$$$$
72 6F 66 20 65 6C 69 66 20 6F 4E 00' 0367                     .ascic "No file for output"
                    74 75 70 74 75 6F  0373
                                12    0367
                                087E  2040
                                087E  2041 tec$bakup:                                   ; page backwards
                     56   00'AB   3C  087E  2042              movzwl  b^inpntr(r11), r6    ; get pointer to input file pointer
                        56   66   D0  0882  2043              movl    (r6), r6             ;  then get input file fab pointer
                             E9   13  0885  2044              beql    10$                  ; no file...
                             56   DD  0887  2045              pushl   r6                   ; save input file fab pointer
                     56   00'AB   3C  0889  2046              movzwl  b^oupntr(r11), r6    ; get pointer to output file pointer
                        56   66   D0  088D  2047              movl    (r6), r6             ;  then get output file fab pointer
                             E5   13  0890  2048              beql    20$                  ; no file...
                           5C B6   7F  0892  2049              pushaq  @fab$q_tecque+4(r6)  ; save current last buffer pointer
                             56   DD  0895  2050              pushl   r6                   ; save output file fab pointer
                           01AD   30  0897  2051              bsbw    tec$putbf            ; now go dump the text buffer
                        56   8E   D0  089A  2052              movl    (sp)+, r6            ; get back output file fab pointer
                  11 50 A6   02   E0  089D  2053              bbs     #fab$v_tecbuf, fab$l_tecsts(r6), 30$ ; already into buffering?
                                08A2  2054                    $rewind -                    ; rewind the output file
                                08A2  2055                            rab=@fab$l_tecrab(r6) ;  so we can re-write it later
                           54 B6   DF  08A2                    PUSHAL  @fab$l_tecrab(r6)
                00000000'GF   01   FB  08A5                    CALLS   #$$.TMP1,G^SYS$REWIND
                          F84F   30  08AC  2056                bsbw    success_or_err       ; check for success completion
                                08AF  2057                    bs      #fab$v_tecbuf, fab$l_tecsts(r6) ; indicate data is buffered
                        50 A6   04   88  08AF                    bisb    s^#1@fab$v_tecbuf, fab$l_tecsts(r6)
                        50   10 AA   AE  08B3  2058 30$:      mnegw   i_r4(r10), r0        ; get number of pages to back up
                           51   8E   7D  08B7  2059              movq    (sp)+, r1            ; get start pointer & input fab pointer
                        53   58 A6   7E  08BA  2060              movaq   fab$q_tecque(r6), r3 ; address the output queue's root
                           53   51   D1  08BE  2061 40$:       cmpl    r1, r3               ; have we backed up to the top?
                                08   13  08C1  2062              beql    50$                  ; yep, must exit the loop early
                        51   04 A1   D0  08C3  2063              movl    4(r1), r1            ; else link backwards to the next
                                50   B7  08C7  2064              decw    r0                   ; should we look at it?
                                F3   14  08C9  2065              bgtr    40$                  ; yes, so loop...
                        10 AA   50   AE  08CB  2066 50$:       mnegw   r0, i_r4(r10)        ; return any remaining count
                        5C A6   51   D1  08CF  2067 60$:       cmpl    r1, fab$q_tecque+4(r6) ; have we finished pruning?
                                0E   13  08D3  2068              beql    70$                  ; yes
                        50   5C B6   0F  08D5  2069              remque  @fab$q_tecque+4(r6), r0 ; no, remove a buffer from output tail
                        58 A2   60   0E  08D9  2070              insque  (r0), fab$q_tecque(r2) ;  and add it to input queue beginning
                                08DD  2071                    bs      #fab$v_tecbuf, fab$l_tecsts(r2) ; say using buffered data now
                        50 A2   04   88  08DD                    bisb    s^#1@fab$v_tecbuf, fab$l_tecsts(r2)
                                EC   11  08E1  2072              brb     60$                  ; loop for the next buffer...
                                08E3  2073
```

```
05  08E3  2074 70$:    rsb                         ; exit
    08E4  2075
    08E4  2076 .disable lsb
```

```
                        08E4  2078 .sbttl  Get input
                        08E4  2079
                        08E4  2080 .enable lsb
                        08E4  2081
                        08E4  2082 10$:     err     NFI, <''No file for input''>
                F85E     30 08E4                     bsbw    err
                 0000037A' 08E7                      .long $$$$$$
                    5879   037A                       .word  $$$$$$
72 6F 66 20 65 6C 69 66 20 6F 4E 00' 037C                .ascic "No file for input"
                74 75 70 6E 69 20  0388
                        11  037C
                        08EB  2083
              50 A6     D6  08EB  2084 20$:     incl    fab$l_tecsts(r6)        ; set internal end-of-file indicator
                        08EE  2085 .assume fab$m_teceof eq 1
                        08EE  2086          $close -                            ; close
                        08EE  2087                  fab=(r6)                    ;  the file
                    66  DF  08EE                    PUSHAL  (r6)
          00000000'GF   01  FB  08F0                CALLS   #$$.TMP1,G^SYS$CLOSE
           010C'CF   0C A6  D0  08F7  2088          movl    fab$l_stv(r6), w^err_msgvec+8 ; save the STV value
                F7FE     30  08FD  2089          bsbw    success_or_err          ; check for success completion
              00'AB   00  B2  0900  2090 30$:     mcomw   #0, b^eoflag(r11)       ; set end-of-file flag
                    05  0904  2091              rsb                             ;  and exit
                        0905  2092
           03 50 A6   09  E1  0905  2093 40$:     bbc     #fab$v_tecb2, fab$l_tecsts(r6), 50$ ; /b2 mode?
                 007B     30  090A  2094          bsbw    150$                    ; yes, go do trailing trims
              00'AB   00  B2  090D  2095 50$:     mcomw   #0, b^ffflag(r11)       ; set form feed flag
                    05  0911  2096              rsb                             ;  and exit
                        0912  2097
                        0912  2098 tec$getbf:                                    ; get input
                00'AB   B4  0912  2099          clrw    b^eoflag(r11)           ; preset not at end-of-file
                00'AB   B4  0915  2100          clrw    b^ffflag(r11)           ; preset no form feed at buffer's end
             56 00'AB   3C  0918  2101          movzwl  b^inpntr(r11), r6       ; get pointer to input file pointer
                56   66  D0  091C  2102          movl    (r6), r6                ;  then get input file fab pointer
                    C3  13  091F  2103          beql    10$                     ; no file
                57   6A  D0  0921  2104          movl    i_r0(r10), r7           ; get buffer pointer
             58   04 AA  D0  0924  2105          movl    i_r1(r10), r8           ;  and room counter
                    56  13  0928  2106          beql    120$                    ; no room at all, we'd better exit...
                    58  D7  092A  2107 60$:     decl    r8                      ; count down the room counter
          04 50 A6   02  E0  092C  2108 70$:     bbs     #fab$v_tecbuf, fab$l_tecsts(r6), 80$ ; never eof if buffering
             CB 50 A6   E8  0931  2109          blbs    fab$l_tecsts(r6), 30$   ; keep saying end-of-file if true
                        0935  2110 .assume fab$m_teceof eq 1
                 0329   30  0935  2111 80$:     bsbw    getbyt                  ; get the next character
          00000000'8F   50  D1  0938  2112          cmpl    r0, #rms$_eof           ; end-of-file?
                    AA  13  093F  2113          beql    20$                     ; yep, so quit
                 F7BA   30  0941  2114          bsbw    success_or_err          ; else check for success completion
          05 50 A6   09  E1  0944  2115          bbc     #fab$v_tecb2, fab$l_tecsts(r6), 90$ ; /b2 mode?
                0D  51  91  0949  2116          cmpb    r1, #13                 ; a <cr>?
                    DE  13  094C  2117          beql    70$                     ; yep, ignore the <cr> (for now...)
                0C  51  91  094E  2118 90$:     cmpb    r1, #12                 ; it is <ff>?
                    B2  13  0951  2119          beql    40$                     ; yep, we're done
                87  51  90  0953  2120          movb    r1, (r7)+               ; nope, store in the text buffer
                00'AB   B6  0956  2121          incw    b^zz(r11)               ;  and count it as stored
                0A  51  91  0959  2122          cmpb    r1, #10                 ; <lf>?
                    19  12  095C  2123          bneq    110$                    ; nope
          0E 50 A6   09  E1  095E  2124          bbc     #fab$v_tecb2, fab$l_tecsts(r6), 100$ ; /b2 mode?
                    1C  10  0963  2125          bsbb    130$                    ; yes, go do trailing trims
           87   0A0D 8F  B0  0965  2126          movw    #13+<10@8>, (r7)+       ; store closing <cr><lf>
```

```
                00'AB   02    A0  096A  2127              addw    #2, b^zz(r11)           ;  and count the added <cr><lf>
                   58   02    C2  096E  2128              subl    #2, r8                  ;   then decrease the free space
                08 AA   58    D1  0971  2129 100$:        cmpl    r8, i_r2(r10)           ; how's room doing?
                        09    19  0975  2130              blss    120$                    ; out of room, we must quit
            00000080 8F 58    D1  0977  2131 110$:        cmpl    r8, #128                ; down to very small??
                        AA    14  097E  2132              bgtr    60$                     ; nope, loop for more...
                              05  0980  2133 120$:        rsb                             ; exit
                                  0981  2134
                        57    D7  0981  2135 130$:        decl    r7                      ; backup the buffer pointer
                     00'AB    B7  0983  2136 140$:        decw    b^zz(r11)               ; remove a character
                        58    D6  0986  2137              incl    r8                      ;  and increase the free space
                  6A    57    D1  0988  2138 150$:        cmpl    r7, i_r0(r10)           ; backed up too far?
                        0F    1B  098B  2139              blequ   160$                    ; yep, we must stop
                  26    77    91  098D  2140              cmpb    -(r7), #^a/&/           ; trailing "&"?
                        F1    13  0990  2141              beql    140$                    ; yes, remove it
                  20    67    91  0992  2142              cmpb    (r7), #32               ; trailing space?
                        EC    13  0995  2143              beql    140$                    ; yes, remove it also
                  09    87    91  0997  2144              cmpb    (r7)+, #9               ; trailing <tab>?
                        E5    13  099A  2145              beql    130$                    ; yes, that gets removed too
                              05  099C  2146 160$:        rsb                             ; exit
                                  099D  2147
                                  099D  2148 .disable lsb
```

```
                        099D  2150 .sbttl  Put output
                        099D  2151
                        099D  2152 dump_data:                              ; dump buffered data
                  03FC  099D  2153          .word   ^m<r2,r3,r4,r5,r6,r7,r8,r9>
      7E    58 A6     7D  099F  2154          movq    fab$q_tecque(r6), -(sp) ; replicate the queue root
      50       6E     7D  09A3  2155          movq    (sp), r0                ; address head (R0) and tail (R1)
      04 A0    6E     7E  09A6  2156          movaq   (sp), 4(r0)             ; re-link head's back ptr to our copy
      61       6E     7E  09AA  2157          movaq   (sp), (r1)              ; re-link tail's forw ptr to our copy
58 A6    58 A6     7E  09AD  2158          movaq   fab$q_tecque(r6), fab$q_tecque(r6) ; reset the real
5C A6    58 A6     7E  09B2  2159          movaq   fab$q_tecque(r6), fab$q_tecque+4(r6) ;  queue root
      50    F8 BD    0F  09B7  2160 10$:     remque  @-8(fp), r0             ; remove next item from the queue
               19    1D  09BB  2161          bvs     20$                     ; nothing more...
               50    DD  09BD  2162          pushl   r0                      ; else save pointer to removed hunk
      53    10 A0    D0  09BF  2163          movl    8+4+4(r0), r3           ; address the data
      54    0C A0    D0  09C3  2164          movl    8+4(r0), r4             ;  and get its count
               59    D4  09C7  2165          clrl    r9                      ; don't add any <ff>
                0086 30  09C9  2166          bsbw    put_buffer              ; now go put it out
            50     8E    D0  09CC  2167          movl    (sp)+, r0               ; get back pointer to removed hunk
0D9F'CF     00    FB  09CF  2168          calls   #0, w^free_data         ;  and free it up
               E1    11  09D4  2169          brb     10$                     ;   then loop for the next...
                        09D6  2170
                  04    09D6  2171 20$:     ret                             ; return
                        09D7  2172
                        09D7  2173 save_data:                              ; save the put buffer data
                  003C  09D7  2174          .word   ^m<r2,r3,r4,r5>
               7E    7C  09D9  2175          clrq    -(sp)                   ; make room for LIB$GET_VM args
63    54       0C    3A  09DB  2176 10$:     locc    #12, r4, (r3)           ; is there an embedded <ff>?
               10    13  09DF  2177          beql    20$                     ; nope
52    54       50    C3  09E1  2178          subl3   r0, r4, r2              ; yep, find this hunk's size
               52    D6  09E5  2179          incl    r2                      ;  including the <ff>
               22    10  09E7  2180          bsbb    60$                     ; go allocate and load a buffer
            53    52    C0  09E9  2181          addl    r2, r3                  ; bump pointer over this hunk
            54    52    C2  09EC  2182          subl    r2, r4                  ;  and skip it in the count
               EA    12  09EF  2183          bneq    10$                     ; loop if there's more to look at...
               59    D5  09F1  2184 20$:     tstl    r9                      ; a trailing <ff> to add?
               02    13  09F3  2185          beql    30$                     ; nope
               54    D6  09F5  2186          incl    r4                      ; yep, so count it in the count
            52    54    D0  09F7  2187 30$:     movl    r4, r2                  ; now use the remaining count
               0B    13  09FA  2188          beql    40$                     ; nothing remaining...
               0D    10  09FC  2189          bsbb    60$                     ; go allocate and load the buffer
               59    D5  09FE  2190          tstl    r9                      ; are we adding a <ff>?
               05    13  0A00  2191          beql    40$                     ; nope
   13 A142     0C    90  0A02  2192          movb    #12, 8+4+8-1(r1)[r2]    ; yep, load in the <ff>
            50    01    D0  0A07  2193 40$:     movl    #1, r0                  ; set success
               04    0A0A  2194 50$:     ret                             ; exit
                        0A0B  2195
F8 AD       52    14    C1  0A0B  2196 60$:     addl3   #8+4+8, r2, -8(fp)      ; set hunk size w/ overhead included
               FC AD    DF  0A10  2197          pushal  -4(fp)                  ; stack address of (returned) address
               F8 AD    DF  0A13  2198          pushal  -8(fp)                  ; stack address of size
00000000'GF    02    FB  0A16  2199          calls   #2, g^lib$get_vm        ; go allocate virtual memory
            EA 50    E9  0A1D  2200          blbc    r0, 50$                 ; exit if any error...
         51    FC AD    D0  0A20  2201          movl    -4(fp), r1              ; address the allocated new memory
               3F    BB  0A24  2202          pushr   #^m<r0,r1,r2,r3,r4,r5>  ; save the MOVC clobbered registers
      5C B6    61    0E  0A26  2203          insque  (r1), @fab$q_tecque+4(r6) ; insert new hunk onto queue's tail
   08 A1    F8 AD    D0  0A2A  2204          movl    -8(fp), 8(r1)           ; load total size of hunk into hunk
      0C A1    52    D0  0A2F  2205          movl    r2, 8+4(r1)             ; load data size of hunk into hunk
   10 A1    14 A1    9E  0A33  2206          movab   8+4+8(r1), 8+4+4(r1)    ; load data address of hunk into hunk
```

```
                14 A1   63   52   28  0A38  2207          movc    r2, (r3), 8+4+8(r1)     ; load data itself into hunk
                            3F   BA  0A3D  2208          popr    #^m<r0,r1,r2,r3,r4,r5>  ; restore the MOVC clobbered registers
                            05   0A3F  2209          rsb                             ; exit
                                 0A40  2210
                                 0A40  2211 .enable lsb
                                 0A40  2212
                                 0A40  2213 10$:     err     NFO, <"No file for output">
                       F702      30   0A40                     bsbw    err
                        0000038E'      0A43                      .long $$$$$$
                             587F      038E                       .word  $$$$$$
72 6F 66 20 65 6C 69 66 20 6F 4E 00'   0390                       .ascic "No file for output"
                74 75 70 74 75 6F 20   039C
                            12   0390
                                 0A47  2214
                                 0A47  2215 tec$putbf:                               ; put output
                    53   6A      D0   0A47  2216          movl    i_r0(r10), r3           ; get text buffer pointer
                54    04 AA      D0   0A4A  2217          movl    i_r1(r10), r4           ;  and character count
                59    08 AA      D0   0A4E  2218          movl    i_r2(r10), r9           ; put emit <ff> flag here
                                 0A52  2219 put_buffer:                              ; internal put output
                56   00'AB      3C   0A52  2220          movzwl  b^oupntr(r11), r6       ; get pointer to output file pointer
                   56    66      D0   0A56  2221          movl    (r6), r6                ;  then get output file fab pointer
                         E5      13   0A59  2222          beql    10$                     ; no file
                57    54 A6      D0   0A5B  2223          movl    fab$l_tecrab(r6), r7    ; file, get the rab pointer
             FF73 CF    00      FB   0A5F  2224          calls   #0, save_data           ; go save the put buffer data
                       F697      30   0A64  2225          bsbw    success_or_err          ; check for success completion
             01 50 A6    02      E1   0A67  2226          bbc     #fab$v_tecbuf, fab$l_tecsts(r6), 20$ ; proceed if not buffering
                                 05   0A6C  2227          rsb                             ; else exit...
                                 0A6D  2228
                    51   53      D0   0A6D  2229 20$:     movl    r3, r1                  ; save starting position
                         52      D4   0A70  2230          clrl    r2                      ;  and reset count
                         54      D7   0A72  2231 30$:     decl    r4                      ; more to look at?
                         7E      19   0A74  2232          blss    130$                    ; nope
                         52      D6   0A76  2233          incl    r2                      ; yep, count another character
                      0C    83      91   0A78  2234          cmpb    (r3)+, #12              ; <ff> or higher?
                            F5      1A   0A7B  2235          bgtru   30$                     ; higher, keep looking...
                   0A    FF A3      91   0A7D  2236          cmpb    -1(r3), #10             ; <lf> or lower?
                            EF      1F   0A81  2237          blssu   30$                     ; lower, keep looking...
                            67      12   0A83  2238          bneq    110$                    ; higher, it's <vt> or <ff>, do record
                1E A6       03      93   0A85  2239          bitb    #fab$m_cr!fab$m_ftn, fab$b_rat(r6) ; lf/cr and/or ftn cc?
                            61      13   0A89  2240          beql    110$                    ; no, so no additions, etc., do record
                      01    52      D1   0A8B  2241          cmpl    r2, #1                  ; anything before the <lf>?
                            5C      15   0A8E  2242          bleq    110$                    ; nope, go do a record
                   0D    FE A3      91   0A90  2243          cmpb    -2(r3), #13             ; yep, it it a <cr>?
                            56      12   0A94  2244          bneq    110$                    ; no, do a record also
                      02    52      D1   0A96  2245          cmpl    r2, #2                  ; anything before the <cr><lf>?
                            06      15   0A99  2246          bleq    40$                     ; nope
                1B    FD A3      91   0A9B  2247          cmpb    -3(r3), #27             ; yep, is it <esc><cr><lf>?
                            4B      13   0A9F  2248          beql    110$                    ; it is, don't remove <cr><lf>, etc.
                   52       02      C2   0AA1  2249 40$:     subl    #2, r2                  ; take away the <cr><lf>
             43 50 A6       09      E1   0AA4  2250          bbc     #fab$v_tecb2, fab$l_tecsts(r6), 110$ ; /b2 mode?
                            52      D5   0AA9  2251          tstl    r2                      ; anything in the record?
                            06      13   0AAB  2252          beql    50$                     ; nope
                26    FD A3      91   0AAD  2253          cmpb    -3(r3), #^a/&/          ; yep, is it <&><cr><lf>?
                            39      13   0AB1  2254          beql    110$                    ; that it is, go output as is
                            54      D5   0AB3  2255 50$:     tstl    r4                      ; more to come?
                            31      15   0AB5  2256          bleq    100$                    ; nope, but check for <ff> coming
                   30       63      91   0AB7  2257          cmpb    (r3), #^a/0/            ; is next a digit?
```

```
                05   1F   0ABA  2258           blssu   60$                     ; non-digit, add "&" to record
           39   63   91   0ABC  2259           cmpb    (r3), #^a/9/            ; really a digit?
                2B   1B   0ABF  2260           blequ   110$                    ; yes, a digit next, go do the record
           7E   73   B0   0AC1  2261  60$:     movw    -(r3), -(sp)            ; save the <cr><lf> from text buffer
      63   2609 8F   B0   0AC4  2262           movw    #9+<^a/&/a8>, (r3)      ; guess at adding <tab><&> to record
           07   52   D1   0AC9  2263           cmpl    r2, #7                  ; already across the first tab stop?
                0D   14   0ACC  2264           bgtr    80$                     ; yep, change that to <space>
           55   51   D0   0ACE  2265           movl    r1, r5                  ; else copy pointer to record
           53   55   D1   0AD1  2266  70$:     cmpl    r5, r3                  ; are we up to record's end?
                08   1E   0AD4  2267           bgequ   90$                     ; yes, no <tab> in record
           09   85   91   0AD6  2268           cmpb    (r5)+, #9               ; is this a <tab>?
                F6   12   0AD9  2269           bneq    70$                     ; not a <tab>, keep looking
           63   20   90   0ADB  2270  80$:     movb    #32, (r3)               ; change to adding <space><&>
           52   02   C0   0ADE  2271  90$:     addl    #2, r2                  ; we added 2 characters to record
                2D   10   0AE1  2272           bsbb    150$                    ; go put the new record
           83   8E   B0   0AE3  2273           movw    (sp)+, (r3)+            ; restore <cr><lf> into text buffer
                06   11   0AE6  2274           brb     120$                    ;  and go check for normal completion
                          0AE8  2275
                59   D5   0AE8  2276  100$:    tstl    r9                      ; a <ff> coming?
                D5   12   0AEA  2277           bneq    60$                     ; yes, go add "&" to record
                22   10   0AEC  2278  110$:    bsbb    150$                    ; go put that record
           F60D      30   0AEE  2279  120$:    bsbw    success_or_err          ; check for success completion
           FF79      31   0AF1  2280           brw     20$                     ;  then loop...
                          0AF4  2281
                59   D5   0AF4  2282  130$:    tstl    r9                      ; add a <ff>?
                10   13   0AF6  2283           beql    140$                    ; nope
           7E   63   90   0AF8  2284           movb    (r3), -(sp)             ; yep, save the next byte
           63   0C   90   0AFB  2285           movb    #12, (r3)               ;  then make it a <ff>
                52   D6   0AFE  2286           incl    r2                      ;   and count that <ff>
                0E   10   0B00  2287           bsbb    150$                    ; go put the final record
           63   8E   90   0B02  2288           movb    (sp)+, (r3)             ; restore the next byte
           F5F6      31   0B05  2289           brw     success_or_err          ; check for success completion & exit
                          0B08  2290
                52   D5   0B08  2291  140$:    tstl    r2                      ; any size?
                24   13   0B0A  2292           beql    170$                    ; nope
           F5EE CF   9F   0B0C  2293           pushab  w^success_or_err        ; check for success completion on exit
      28 A7     51   D0   0B10  2294  150$:    movl    r1, rab$l_rbf(r7)       ; set starting address of record
      22 A7     52   B0   0B14  2295           movw    r2, rab$w_rsz(r7)       ;  and record's size
                58   D4   0B18  2296           clrl    r8                      ; clear our "over quota" flag
                          0B1A  2297  160$:    $put -                          ; put a record
                          0B1A  2298                 rab=(r7)                 ;  to the file
                67   DF   0B1A                 PUSHAL  (r7)
 00000000'GF      01   FB   0B1C                 CALLS   #$$.TMP1,G^SYS$PUT
  010C'CF    0C A7   D0   0B23  2299           movl    rab$l_stv(r7), w^err_msgvec+8 ; save the STV value
        04 50   E8   0B29  2300           blbs    r0, 170$                ; all done if no error
      01 58     00   E3   0B2C  2301           bbcs    #0, r8, 180$            ; really an error if second time
                05   0B30  2302  170$:    rsb                             ; exit
                          0B31  2303
00000000'8F   0C A7   D1   0B31  2304  180$:    cmpl    rab$l_stv(r7), #ss$_exdiskquota ; is it the quota error?
                F5   12   0B39  2305           bneq    170$                    ; nope, a real true error
                          0B3B  2306           bs      s^#io$v_canctrlo, ctlofg ; do a control/o cancel
00 00000000'EF    00'  E3   0B3B                 bbcs    s^#io$v_canctrlo, ctlofg, 30014$
                          0B43         30014$:
        0323'CF   7F   0B43  2307           pushaq  w^quota_msg_desc        ; set the quota exceeded message
      FB14 CF     01   FB   0B47  2308           calls   #1, w^tec$out_ascid     ;  and go output it
                CC   11   0B4C  2309           brb     160$                    ; now go try, try again...
                          0B4E  2310
```

```
0B4E  2311 .disable lsb
```

```
                        0B4E  2313 .sbttl  Get an input byte
                        0B4E  2314
                        0B4E  2315 .enable lsb
                        0B4E  2316
                        0B4E  2317 getbyt_emit_ctl:                        ; emit a control character
51   FFFFFFA0 8F   CA   0B4E  2318          bicl    #^c<<1@6>!31>, r1       ; trim to 8-bit flag & control char
      0E 51    06   E5  0B55  2319          bbcc    #6, r1, 10$             ; use 7-bit set control if <6>=0
                        0B59  2320          bs      #7, r1                  ; else use 8-bit set control
      00 51    07   E3  0B59                 bbcs    #7, r1, 30015$
                        0B5D                30015$:
               08   11  0B5D  2321          brb     10$                     ;  and go emit it
                        0B5F  2322
                        0B5F  2323 getbyt_emit_cr:                         ; emit a <cr>
          51   0D   9A  0B5F  2324          movzbl  #13, r1                 ; set the <cr>
               03   11  0B62  2325          brb     10$                     ;  then go emit it
                        0B64  2326
                        0B64  2327 getbyt_emit_lf:                         ; emit a <lf>
          51   0A   9A  0B64  2328          movzbl  #10, r1                 ; set the <lf>
      60 A6    8E   D0  0B67  2329 10$:     movl    (sp)+, fab$l_tecdsp(r6) ; set the next dispatch address
                        0B6B  2330          rsb                             ; exit
                        0B6C  2331
   1D 1E A6    02   E1  0B6C  2332 20$:     bbc     #fab$v_prn, fab$b_rat(r6), 40$ ; br if not print file format
00000000'8F    50   D1  0B71  2333          cmpl    r0, #rms$_eof           ; end-of-file?
               14   12  0B78  2334          bneq    40$                     ; nope, other error
   06 50 A6    03   E0  0B7A  2335          bbs     #fab$v_tecicr, fab$l_tecsts(r6), 30$ ; just ignored a <cr>?
      0D   64 A6    91  0B7F  2336          cmpb    fab$l_tecctl(r6), #13   ; was last character a <cr>?
               09   12  0B83  2337          bneq    40$                     ; not a <cr>, nothing more needed...
               DD   10  0B85  2338 30$:     bsbb    getbyt_emit_lf          ; ignored or last <cr>, go emit a <lf>
50   00000000'8F    D0  0B87  2339          movl    #rms$_eof, r0           ; now restore the end-of-file code
               8E   D5  0B8E  2340 40$:     tstl    (sp)+                   ; pop the return exit
               05   10  0B90  2341          rsb                             ;  and exit with error code
                        0B91  2342
                        0B91  2343 getbyt_first:                           ; initial dispatch entry point
   60 A6    FD AF   9E  0B91  2344          movab   b^getbyt_first, fab$l_tecdsp(r6) ; set new record dispatch
      55    54 A6   D0  0B96  2345          movl    fab$l_tecrab(r6), r5    ; get the rab pointer
                        0B9A  2346          $get -                          ; get
                        0B9A  2347                  rab=(r5)                ;  the next record
                  65    DF  0B9A                  PUSHAL  (r5)
00000000'GF       01    FB  0B9C                  CALLS   #$$.TMP1,G^SYS$GET
 010C'CF       0C   A5  D0  0BA3  2348          movl    rab$l_stv(r5), w^err_msgvec+8 ; save the STV value
00000000'8F       50   D1  0BA9  2349          cmpl    r0, #rms$_normal        ; normal completion?
                  BA   12  0BB0  2350          bneq    20$                     ; nope, exit with error code
   26 1E A6       02   E1  0BB2  2351          bbc     #fab$v_prn, fab$b_rat(r6), 90$ ; print file format?
   66 A6    2C   B5  B0  0BB7  2352          movw    @rab$l_rhb(r5), fab$l_tecctl+2(r6) ; copy vfc bytes
      51    66 A6   98  0BBC  2353          cvtbl   fab$l_tecctl+2(r6), r1  ; get the 'prefix' byte
                  15   13  0BC0  2354          beql    80$                     ; none
                  10   19  0BC2  2355          blss    70$                     ; a control
   05 50 A6       01   E2  0BC4  2356          bbss    #fab$v_tecnolst, fab$l_tecsts(r6), 60$ ; first time?
            66 A6    97  0BC9  2357 50$:     decb    fab$l_tecctl+2(r6)      ; another 'prefix' <cr><lf> to do?
                  20   15  0BCC  2358          bleq    100$                    ; nope, go do the real data
                  8F   10  0BCE  2359 60$:     bsbb    getbyt_emit_cr          ; yep, emit a <cr>
                  92   10  0BD0  2360          bsbb    getbyt_emit_lf          ; emit a <lf>
                  F5   11  0BD2  2361          brb     50$                     ; now loop...
                        0BD4  2362
               FF77   30  0BD4  2363 70$:     bsbw    getbyt_emit_ctl         ; emit a control character
                        0BD7  2364 80$:     bs      #fab$v_tecnolst, fab$l_tecsts(r6) ; not first time anymore
      50 A6    02   88  0BD7                 bisb    s^#1@fab$v_tecnolst, fab$l_tecsts(r6)
```

```
                11   11  0BDB  2365          brb     100$                    ;  then continue
                                 0BDD  2366
             67 A6   94  0BDD  2367 90$:     clrb    fab$l_tecctl+3(r6)      ; guess at no carriage control
          1E A6 03   93  0BE0  2368          bitb    #fab$m_cr!fab$m_ftn, fab$b_rat(r6) ; implied lf/cr or ftn ccl?
                08   13  0BE4  2369          beql    100$                    ; neither
             67 A6   96  0BE6  2370          incb    fab$l_tecctl+3(r6)      ; one or the other, do cr/lf at end
             22 A5   B5  0BE9  2371          tstw    rab$w_rsz(r5)           ; a null record?
                36   13  0BEC  2372          beql    130$                    ; yep, no need to process it...
       60 A6 F3'AF   9E  0BEE  2373 100$:    movab   b^110$, fab$l_tecdsp(r6) ; set dispatch for record data
          55 54 A6   D0  0BF3  2374 110$:    movl    fab$l_tecrab(r6), r5    ; get the rab pointer
             22 A5   B7  0BF7  2375          decw    rab$w_rsz(r5)           ; more in the record?
                08   19  0BFA  2376          blss    120$                    ; nope
          51 28 B5   9A  0BFC  2377          movzbl  @rab$l_rbf(r5), r1      ; yep, so get a byte
             28 A5   D6  0C00  2378          incl    rab$l_rbf(r5)           ;  and bump the record pointer
                05   0C03  2379          rsb                             ;   then exit
                                 0C04  2380
          51 67 A6   98  0C04  2381 120$:    cvtbl   fab$l_tecctl+3(r6), r1  ; get the 'postfix' byte
                25   13  0C08  2382          beql    140$                    ; none
                26   19  0C0A  2383          blss    150$                    ; a control
       13 1E A6 02   E0  0C0C  2384          bbs     #fab$v_prn, fab$b_rat(r6), 130$ ; print file format?
          51 64 A6   9A  0C11  2385          movzbl  fab$l_tecctl(r6), r1    ; get last record data byte
             1B 51   91  0C15  2386          cmpb    r1, #27                 ; was it escape?
                15   13  0C18  2387          beql    140$                    ; it was, no additions
             0C 51   91  0C1A  2388          cmpb    r1, #12                 ; was it <ff> or higher?
                05   1A  0C1D  2389          bgtru   130$                    ; higher, go add a <cr><lf>
             0A 51   91  0C1F  2390          cmpb    r1, #10                 ; was it <lf>, <vt>, or <ff>?
                0B   1E  0C22  2391          bgequ   140$                    ; one of the above, no additions
              FF38   30  0C24  2392 130$:    bsbw    getbyt_emit_cr          ; emit a <cr>
              FF3A   30  0C27  2393          bsbw    getbyt_emit_lf          ; emit a <lf>
             67 A6   97  0C2A  2394          decb    fab$l_tecctl+3(r6)      ; another 'postfix' <lf> to do?
                F5   14  0C2D  2395          bgtr    130$                    ; yep, so do another
              FF5F   31  0C2F  2396 140$:    brw     getbyt_first            ; now is the time for a new record...
                                 0C32  2397
              FF19   30  0C32  2398 150$:    bsbw    getbyt_emit_ctl         ; emit a control character
                F8   11  0C35  2399          brb     140$                    ;  then go to the next record
                                 0C37  2400
                                 0C37  2401 .disable lsb
                                 0C37  2402
                                 0C37  2403 .enable lsb
                                 0C37  2404
          50 58 A6   D0  0C37  2405 10$:     movl    fab$q_tecque(r6), r0    ; get queued data buffer pointer
          51 10 B0   9A  0C3B  2406          movzbl  @8+4+4(r0), r1          ; get character from buffer
             10 A0   D6  0C3F  2407          incl    8+4+4(r0)               ; bump the buffer pointer
             0C A0   D7  0C42  2408          decl    8+4(r0)                 ; count down the count
                46   12  0C45  2409          bneq    60$                     ; more remains, go exit
          50 58 B6   0F  0C47  2410          remque  @fab$q_tecque(r6), r0   ; no more, remove buffer from queue
                04   12  0C4B  2411          bneq    20$                     ; another buffer remains in the queue
                                 0C4D  2412          bc      #fab$v_tecbuf, fab$l_tecsts(r6) ; else turn off buffering
          50 A6 04   8A  0C4D                  bicb    s^#1@fab$v_tecbuf, fab$l_tecsts(r6)
    0D9F'CF 00   FB  0C51  2413 20$:     calls   #0, w^free_data         ; go free up the buffer
                35   11  0C56  2414          brb     60$                     ; now go exit
                                 0C58  2415
             65 A6   95  0C58  2416 30$:     tstb    fab$l_tecctl+1(r6)      ; have <cr>, already at left margin?
                49   12  0C5B  2417          bneq    90$                     ; nope, we need this <cr>
                                 0C5D  2418          bs      #fab$v_tecicr, fab$l_tecsts(r6) ; yep, say <cr> ignored...
          50 A6 08   88  0C5D                  bisb    s^#1@fab$v_tecicr, fab$l_tecsts(r6)
                                 0C61  2419 getbyt:                             ; get an input byte
```

```
                D1 50 A6    02    E0  0C61  2420          bbs     #fab$v_tecbuf, fab$l_tecsts(r6), 10$ ; is there buffered data?
                38 50 A6    04    E4  0C66  2421          bbsc    #fab$v_tececr, fab$l_tecsts(r6), 80$ ; do <lf> if extra <cr>
                      60 B6     16  E8  0C6B  2422          jsb     @fab$l_tecdsp(r6)       ; else dispatch to proper routine
                16 1E A6    02    E1  0C6E  2423          bbc     #fab$v_prn, fab$b_rat(r6), 50$ ; br if not print file format
                                      0C73  2424          bc      #fab$v_tecicr, fab$l_tecsts(r6) ; undo <cr> ignored flag
                   50 A6    08    8A  0C73                 bicb    s^#1@fab$v_tecicr, fab$l_tecsts(r6)
                      0D    51    91  0C77  2425          cmpb    r1, #13                 ; <cr> or greater?
                            09    1A  0C7A  2426          bgtru   40$                     ; greater, normal & no left margin
                            DA    13  0C7C  2427          beql    30$                     ; it's <cr>, so left margin
                         0A   51    91  0C7E  2428          cmpb    r1, #10                 ; is it <lf>, <vt>, or <ff>?
                            06    1A  0C81  2429          bgtru   50$                     ; it's <vt> or <ff>
                            10    13  0C83  2430          beql    70$                     ; it's <lf>, check for needing <cr>
                   65 A6    01    90  0C85  2431  40$:    movb    #1, fab$l_tecctl+1(r6)  ; set non-zero for not at left margin
                   64 A6    51    90  0C89  2432  50$:    movb    r1, fab$l_tecctl(r6)    ; remember last character returned
             50    00000000'8F    D0  0C8D  2433  60$:    movl    #rms$_normal, r0        ; set normal completion
                                   05  0C94  2434          rsb                             ;  and exit
                                      0C95  2435
                   64 A6    51    B1  0C95  2436  70$:    cmpw    r1, fab$l_tecctl(r6)    ; 2 <lf>'s at left margin?
                            EE    12  0C99  2437          bneq    50$                     ; nope
                      51    0D    9A  0C9B  2438          movzbl  #13, r1                 ; yep, set the missing <cr>
                EA 50 A6    04    E3  0C9E  2439          bbcs    #fab$v_tececr, fab$l_tecsts(r6), 60$ ; emit the extra <cr>
                      51    0A    9A  0CA3  2440  80$:    movzbl  #10, r1                 ; now (re-)set the <lf>
                         65 A6    94  0CA6  2441  90$:    clrb    fab$l_tecctl+1(r6)      ; indicate left margin
                            DE    11  0CA9  2442          brb     50$                     ;  and go exit with <cr> or <lf>
                                      0CAB  2443
                                      0CAB  2444 .disable lsb
```

```
                        0CAB  2446 .sbttl  Switch to alternate output
                        0CAB  2447
                        0CAB  2448 .enable lsb
                        0CAB  2449
                        0CAB  2450 tec$outsv:                                  ; switch output file
00'AB   0000'8F   B0    0CAB  2451         movw    #oupalt, b^oupntr(r11)      ; do the pointer switch
                        0CB1  2452 set_outputname:                             ; set output file name, etc.
     57   00'AB   3C    0CB1  2453         movzwl  b^oupntr(r11), r7           ; get pointer to output file pointer
             0D   11    0CB5  2454         brb     10$                         ; go set file name, etc.
                        0CB7  2455
                        0CB7  2456 .sbttl  Switch to alternate input
                        0CB7  2457
                        0CB7  2458 tec$inpsv:                                  ; switch input file
00'AB   0000'8F   B0    0CB7  2459         movw    #inpalt, b^inpntr(r11)      ; do the pointer switch
                        0CBD  2460 set_inputname:                              ; set input file name, etc.
         00'AB   B4    0CBD  2461         clrw    b^eoflag(r11)               ; guess at not at end-of-file
     57  00'AB   3C    0CC0  2462         movzwl  b^inpntr(r11), r7           ; get pointer to input file pointer
  00000000'EF   94    0CC4  2463 10$:    clrb    filsrt                      ; guess at file closed (no name)
        56   67   D0    0CCA  2464         movl    (r7), r6                    ; get file's fab pointer
             43   13    0CCD  2465         beql    50$                         ; file is closed
  08 50 A6    02   E0    0CCF  2466         bbs     #fab$v_tecbuf, fab$l_tecsts(r6), 20$ ; never eof if buffering
     04 50 A6   E9    0CD4  2467         blbc    fab$l_tecsts(r6), 20$       ; branch if not at end-of-file
                        0CD8  2468 .assume fab$m_teceof eq 1
     00'AB   00   B2    0CD8  2469         mcomw   #0, b^eoflag(r11)           ; eof, indicate such
                        0CDC  2470 set_filename:                               ; set a file name, etc.
     50   28 A6   D0    0CDC  2471 20$:    movl    fab$l_nam(r6), r0           ; get pointer to nam from fab
     51   04 A0   D0    0CE0  2472         movl    nam$l_rsa(r0), r1           ;  and pointer to filename
     52   03 A0   9A    0CE4  2473         movzbl  nam$b_rsl(r0), r2           ;   and get filename's length
             08   13    0CE8  2474         beql    30$                         ; no length?
             61   95    0CEA  2475         tstb    (r1)                        ; a starting null?
             04   12    0CEC  2476         bneq    30$                         ; nope
             51   D6    0CEE  2477         incl    r1                          ; yep, skip it
             52   D7    0CF0  2478         decl    r2                          ;  also skip in count
        FB'8F    52   91    0CF2  2479 30$:    cmpb    r2, #filsiz-3-1-1           ; will the whole specification fit?
             04   1B    0CF6  2480         blequ   40$                         ; yep
     52   FB'8F   9A    0CF8  2481         movzbl  #filsiz-3-1-1, r2           ; nope, we must truncate it
00000000'EF   61   52   28    0CFC  2482 40$:    movc    r2, (r1), filsrt            ; move in the file specification
             63   94    0D04  2483         clrb    (r3)                        ;  and make result asciz
     07 50 A6    09   E1    0D06  2484         bbc     #fab$v_tecb2, fab$l_tecsts(r6), 50$ ; /b2?
 63   0032422F   8F   D0    0D0B  2485         movl    #^a"/B2", (r3)              ; yep, add the switch and asciz again
                  05    0D12  2486 50$:    rsb                                 ; exit
                        0D13  2487
                        0D13  2488 .disable lsb
```

```
                        0D13  2490 .sbttl  Close input & output files
                        0D13  2491
                        0D13  2492 .enable lsb
                        0D13  2493
                        0D13  2494 tec$clsfl:                              ; close input & output files
                 3E  10 0D13  2495          bsbb    close_input            ; close the input file first
                        0D15  2496
                        0D15  2497 .sbttl  Close output file
                        0D15  2498
                        0D15  2499 tec$clsof:                              ; close output file
              57  00'AB  3C  0D15  2500          movzwl  b^oupntr(r11), r7  ; get pointer to output file pointer
                 56  67  D0  0D19  2501 10$:     movl    (r7), r6           ; get file's fab pointer
                     47  13  0D1C  2502          beql    40$                ; no file
           0A 04 A6  0F  E5  0D1E  2503          bbcc    #fab$v_dlt, fab$l_fop(r6), 20$ ; a file, don't delete it
           05 50 A6  02  E5  0D23  2504          bbcc    #fab$v_tecbuf, fab$l_tecsts(r6), 20$ ; need data buffer dump?
            FC70 CF  00  FB  0D28  2505          calls   #0, w^dump_data    ; yes, so go do so
                     67  D4  0D2D  2506 20$:     clrl    (r7)               ; file is open no longer
                 32 50 A6  E8  0D2F  2507          blbs    fab$l_tecsts(r6), 40$ ; already closed if at end-of-file
                        0D33  2508 .assume fab$m_teceof eq 1
                       0062  30  0D33  2509          bsbw    70$                ; release any and all data lines...
                        0D36  2510          $close -                   ; close
                        0D36  2511                  fab=(r6)           ;  the file
                     66  DF  0D36                  PUSHAL  (r6)
          00000000'GF  01  FB  0D38                  CALLS   #$$.TMP1,G^SYS$CLOSE
           010C'CF  0C A6  D0  0D3F  2512          movl    fab$l_stv(r6), w^err_msgvec+8 ; save the STV value
          00000000'8F  50  D1  0D45  2513          cmpl    r0, #rms$_eof      ; did we get end-of-file (why?)?
                     02  12  0D4C  2514          bneq    30$                ; nope
                     50  D6  0D4E  2515          incl    r0                 ; yep, fudge for success (RMS bug?)
                   F3AB  31  0D50  2516 30$:     brw     success_or_err     ; check for success completion & exit
                        0D53  2517
                        0D53  2518 close_input:                            ; close input file
                  00'AB  B4  0D53  2519          clrw    b^eoflag(r11)      ; never eof if input file closed
                57  00'AB  3C  0D56  2520          movzwl  b^inpntr(r11), r7  ; get pointer to input file pointer
                     BD  11  0D5A  2521          brb     10$                ;  then go close it & exit
                        0D5C  2522
                        0D5C  2523 .sbttl  Kill output file
                        0D5C  2524
                        0D5C  2525 tec$kilfl:                              ; delete output file
                57  00'AB  3C  0D5C  2526          movzwl  b^oupntr(r11), r7  ; get pointer to output file pointer
                   56  67  D0  0D60  2527          movl    (r7), r6           ;  then get output file fab pointer
                     C8  12  0D63  2528          bneq    20$                ; a file, go close & delete it
                         05  0D65  2529 40$:     rsb                        ; else just exit
                        0D66  2530
                        0D66  2531 .sbttl  Close indirect command file
                        0D66  2532
                        0D66  2533 close_indir:                            ; close indirect w/ error checking
                     E7 AF  9F  0D66  2534          pushab  b^30$              ; error check upon exit
                        0D69  2535 reset_indir:                            ; close indirect command file
                  00'AB  B4  0D69  2536          clrw    b^indir(r11)       ; ensure indirect file looks closed
          50  00000000'8F  D0  0D6C  2537          movl    #rms$_normal, r0   ; pre-set o.k. if file's not open
          56  00000000'EF  D0  0D73  2538          movl    cmdprm, r6         ; get indirect file fab pointer
                     17  13  0D7A  2539          beql    50$                ; no file
                     1A  10  0D7C  2540          bsbb    70$                ; release any and all data lines...
                        0D7E  2541          $close -                   ; close
                        0D7E  2542                  fab=(r6)           ;  the file
                     66  DF  0D7E                  PUSHAL  (r6)
          00000000'GF  01  FB  0D80                  CALLS   #$$.TMP1,G^SYS$CLOSE
```

```
                00000000'EF   D4   0D87  2543          clrl    cmdprm                          ;   and say we did it
          010C'CF    0C A6    D0   0D8D  2544          movl    fab$l_stv(r6), w^err_msgvec+8 ; save the STV value
                              05   0D93  2545 50$:      rsb                                     ; exit
                                   0D94  2546
                    9F'AF     00   FB   0D94  2547 60$:      calls   #0, b^free_data         ; go free up the buffer
                    50     58 B6   0F   0D98  2548 70$:      remque  @fab$q_tecque(r6), r0   ; dequeue the next buffer
                              F6   1C   0D9C  2549          bvc     60$                     ; there's one...
                              05   0D9E  2550          rsb                                     ; no more, exit
                                   0D9F  2551
                                   0D9F  2552 .disable lsb
                                   0D9F  2553
                                   0D9F  2554 free_data:                                      ; free up a data buffer
                         0002      0D9F  2555          .word   ^m<r1>
                            50     DD   0DA1  2556          pushl   r0                      ; stack the buffer's address
                         08 A0     DD   0DA3  2557          pushl   8(r0)                   ; stack the buffer's size
                         FC AD     DF   0DA6  2558          pushal  -4(fp)                  ; stack address of buffer's address
                         F8 AD     DF   0DA9  2559          pushal  -8(fp)                  ; stack address of buffer's size
              00000000'GF    02   FB   0DAC  2560          calls   #2, g^lib$free_vm       ; go free up virtual memory
                              04   0DB3  2561          ret                                     ; exit
                                   0DB4  2562
                                   0DB4  2563 .sbttl  Error message finish up
                                   0DB4  2564
                                   0DB4  2565 tec$aller:                                      ; error message finish up
                         0496      30   0DB4  2566          bsbw    not_exiting             ; not exiting if error
          03    00'AB    02    00   ED   0DB7  2567          cmpzv   #0, #2, b^ehelp(r11), #3 ; want long form error message?
                              AA   12   0DBD  2568          bneq    reset_indir             ; nope, so don't give it
                         50   00   D2   0DBF  2569          mcoml   #0, r0                  ; signal error message printing (-1)
                            F23B'  30   0DC2  2570          bsbw    tecoexelbr              ;  and go do it
                              A2   11   0DC5  2571          brb     reset_indir             ; go ensure no indirect and exit
                                   0DC7  2572
                                   0DC7  2573          tmporg  tecoexelbr
                                   0000  2574
                         50        7C   0000  2575          clrq    r0                      ; say nothing here...
                                   05   0002  2576          rsb                             ;  and exit
                                        0003  2577
                                        0003  2578          unorg
```

```
                        0DC7  2580 .sbttl  Get files opened, etc.
                        0DC7  2581
                        0DC7  2582 tec$getfl:                                   ; get files (EB, EI, EN, ER, EW)
             02B0   30  0DC7  2583          bsbw    fetch_filbuf                ; go fetch the filename buffer
               26   12  0DCA  2584          bneq    non_null                    ; go off to process non-null...
          08 AA   B5  0DCC  2585          tstw    i_r2(r10)                   ; null, but what is it for?
               18   14  0DCF  2586          bgtr    20$                         ; it's EW
               0D   13  0DD1  2587          beql    10$                         ; it's ER
FFF7 8F   08 AA   B1  0DD3  2588          cmpw    i_r2(r10),#^a/I/-^a/R/      ; come on now, what is it really?
               8B   13  0DD9  2589          beql    close_indir                 ; it's EI, go close indirect
               15   19  0DDB  2590          blss    non_null                    ; it's EB, we'll die in parse...
             0334   31  0DDD  2591          brw     en_next                     ; it's EN, go get next occurance
                        0DE0  2592
00'AB   0000'8F   B0  0DE0  2593 10$:     movw    #inpnor, b^inpntr(r11)      ; do the pointer switch
             FED4   31  0DE6  2594          brw     set_inputname               ;  and go set input file name, etc.
                        0DE9  2595
00'AB   0000'8F   B0  0DE9  2596 20$:     movw    #oupnor, b^oupntr(r11)      ; do the pointer switch
             FEBF   31  0DEF  2597          brw     set_outputname              ;  and go set output file name, etc.
                        0DF2  2598
                        0DF2  2599 non_null:                                    ; non-null file specification
   51    08ED'CF   9E  0DF2  2600          movab   w^file_spec_buf, r1         ; get pointer to filespec buffer
         52   51   D0  0DF7  2601          movl    r1, r2                      ;  in two registers
         08EC'CF   94  0DFA  2602          clrb    w^file_spec_len             ; reset the filespec's length
         08E0'CF   D4  0DFE  2603          clrl    w^file_spec_opt             ;  and options
              53   D4  0E02  2604          clrl    r3                          ; say outside of quotes initially
             02BB   30  0E04  2605 10$:     bsbw    get_file_char               ; get a character
               48   13  0E07  2606          beql    70$                         ; null, the end
            05 53   E8  0E09  2607          blbs    r3, 20$                     ; branch if inside quotes
         2F   50   91  0E0C  2608          cmpb    r0, #^a"/"                  ; else check for a switch
               09   13  0E0F  2609          beql    30$                         ; found one, switches start here...
         81   50   90  0E11  2610 20$:     movb    r0, (r1)+                   ; (re-)store into filespec's buffer
         08EC'CF   96  0E14  2611          incb    w^file_spec_len             ;  and (re-)count it in the length
               EA   11  0E18  2612          brb     10$                         ;   then loop for more...
                        0E1A  2613
   51    08E4'CF   DE  0E1A  2614 30$:     moval   w^file_spec_swt, r1         ; point to the switch buffer
               61   D4  0E1F  2615          clrl    (r1)                        ;  and clear it
             029E   30  0E21  2616 40$:     bsbw    get_file_char               ; get a switch character
               10   13  0E24  2617          beql    50$                         ; null, end of this switch & spec too
         2F   50   91  0E26  2618          cmpb    r0, #^a"/"                  ; start of another switch?
               0B   13  0E29  2619          beql    50$                         ; yes, go end this one first
         81   50   90  0E2B  2620          movb    r0, (r1)+                   ; else store a switch character
         08E7'CF   95  0E2E  2621          tstb    w^file_spec_swt+3           ; did we store too many?
               ED   13  0E32  2622          beql    40$                         ; nope, continue
               21   11  0E34  2623          brb     80$                         ; yep, go give an error...
                        0E36  2624
   51    024C'CF   7E  0E36  2625 50$:     movaq   w^switch_list-4, r1         ; get the switch list
               81   D5  0E3B  2626 60$:     tstl    (r1)+                       ; skip the bit pattern
               61   95  0E3D  2627          tstb    (r1)                        ; more to check?
               16   13  0E3F  2628          beql    80$                         ; nope, go give an error
   81    08E4'CF   D1  0E41  2629          cmpl    w^file_spec_swt, (r1)+      ; a match?
               F3   12  0E46  2630          bneq    60$                         ; nope, keep checking...
   08E0'CF    61   C8  0E48  2631          bisl    (r1), w^file_spec_opt       ; yep, set the correct bit(s)
               50   95  0E4D  2632          tstb    r0                          ; is there more to come?
               C9   12  0E4F  2633          bneq    30$                         ; yes, go get it...
         08EC'CF   95  0E51  2634 70$:     tstb    w^file_spec_len             ; did we get any length at all?
               0D   12  0E55  2635          bneq    do_non_null                 ; yep
50   00000000'8F   D0  0E57  2636 80$:     movl    #rms$_syn, r0               ; say that's illegal
```

```
                                0E5E  2637
                                0E5E  2638 .enable lsb
                                0E5E  2639
                     F29D   31  0E5E  2640 10$:    brw     success_or_err          ; go die with the error
                                0E61  2641
                     028B   31  0E61  2642 20$:    brw     en_preset               ; go do preset for "en"
                                0E64  2643
                                0E64  2644 do_non_null:                            ; do real opens, etc.
                   08 AA    B5  0E64  2645          tstw    i_r2(r10)               ; what is it for?
                      0C    14  0E67  2646          bgtr    30$                     ; it's EW
                      42    13  0E69  2647          beql    70$                     ; it's ER
           FFF7 8F    08 AA    B1  0E6B  2648          cmpw    i_r2(r10),#^a/I/-^a/R/  ; come on now, what is it really?
                      15    13  0E71  2649          beql    40$                     ; it's EI
                      EC    14  0E73  2650          bgtr    20$                     ; it's EN
               57   00'AB    3C  0E75  2651 30$:    movzwl  b^oupntr(r11), r7       ; EB/EW, point to output file pointer
                  56   67    D0  0E79  2652          movl    (r7), r6                ;  then get output file fab pointer
                      27    13  0E7C  2653          beql    60$                     ; closed is o.k.
                     FE30   30  0E7E  2654          bsbw    set_outputname          ; already open, set open file's name
                                0E81  2655          err     OFO, <"Output file already open">
                     F2C1   30  0E81                  bsbw    err
                      000003A3' 0E84                  .long $$$$$$
                          5EBF  03A3                  .word  $$$$$$
65 6C 69 66 20 74 75 70 74 75 4F 00' 03A5                  .ascic "Output file already open"
65 70 6F 20 79 64 61 65 72 6C 61 20  03B1
                                6E    03BD
                                18    03A5
                                0E88  2656
                     FEDB   30  0E88  2657 40$:    bsbw    close_indir             ; close the current indirect file
                56   0378'CF    DE  0E8B  2658          moval   w^indir_cmd_fab, r6     ; get the indirect command fab
            57   00000000'EF    DE  0E90  2659          moval   cmdprm, r7              ;  and where to store fab pointer
                      28    11  0E97  2660          brb     80$                     ;   then join common open code
                                0E99  2661
                   00'AB    B5  0E99  2662 50$:    tstw    b^nflg(r11)             ; are we returning a value?
                      C0    18  0E9C  2663          bgeq    10$                     ; no, just die with the error
                                0E9E  2664          err     FNF, <"File not found">
                     F2A4   30  0E9E                  bsbw    err
                      000003BE' 0EA1                  .long $$$$$$
                          27B6  03BE                  .word  $$$$$$
6F 66 20 74 6F 6E 20 65 6C 69 46 00' 03C0                  .ascic "File not found"
                             64 6E 75  03CC
                                0E    03C0
                                0EA5  2665
                   08 AA    B5  0EA5  2666 60$:    tstw    i_r2(r10)               ; is it EB or EW?
                      03    15  0EA8  2667          bleq    70$                     ; it's EB
                     00D8   31  0EAA  2668          brw     170$                    ; it's EW
                                0EAD  2669
                     FEA3   30  0EAD  2670 70$:    bsbw    close_input             ; close the current input file
                56   0110'CF    DE  0EB0  2671          moval   w^input_nor_fab, r6     ; guess at normal input
                0000'8F    57  B1  0EB5  2672          cmpw    r7, #inpnor             ; good guess?
                      05    13  0EBA  2673          beql    80$                     ; yep
                56   0484'CF    DE  0EBC  2674          moval   w^input_alt_fab, r6     ; nope, alternate input
             50 A6   08E0'CF    D0  0EC1  2675 80$:    movl    w^file_spec_opt, fab$l_tecsts(r6) ; set file spec options
             60 A6   FCC6 CF    9E  0EC7  2676          movab   w^getbyt_first, fab$l_tecdsp(r6) ; reset the get byte dispatch
                64 A6    0A    9A  0ECD  2677          movzbl  #10, fab$l_tecctl(r6)   ;  and the control bytes
          04 A6   00000040 8F    D0  0ED1  2678          movl    #fab$m_sqo, fab$l_fop(r6) ; reset file options
             1E A6   0202 8F    B0  0ED9  2679          movw    #<fab$c_var@8>!fab$m_cr, fab$b_rat(r6) ; reset record fmt/attr
                                0EDF  2680 .assume fab$b_rfm eq fab$b_rat+1
```

```
                17 A6    02    90   0EDF  2681          movb    #fab$m_get, fab$b_shr(r6) ; set sharing to only other gets
             05 50 A6    07    E1   0EE3  2682          bbc     #fab$v_tecrw, fab$l_tecsts(r6), 90$ ; rw?
                                  0EE8  2683          bs      #fab$v_rwo, fab$l_fop(r6) ; set rewind before open
             00 04 A6    07    E3   0EE8                bbcs    #fab$v_rwo, fab$l_fop(r6), 30020$
                                  0EED        30020$:
             05 50 A6    08    E1   0EED  2684  90$:    bbc     #fab$v_tecsh, fab$l_tecsts(r6), 100$ ; /sh?
                17 A6    43 8F   90   0EF2  2685        movb    #fab$m_get!fab$m_put!fab$m_upi, fab$b_shr(r6) ; set sharing
          34 A6    08EC'CF    90   0EF7  2686  100$:   movb    w^file_spec_len, fab$b_fns(r6) ; set file spec's length
                                  0EFD  2687          $open -                         ; open
                                  0EFD  2688                      fab=(r6)            ;  the input file
                         66    DF   0EFD                PUSHAL  (r6)
          00000000'GF    01    FB   0EFF                CALLS   #$$.TMP1,G^SYS$OPEN
           010C'CF    0C A6    D0   0F06  2689          movl    fab$l_stv(r6), w^err_msgvec+8 ; save the STV value
                      8A 50    E9   0F0C  2690          blbc    r0, 50$                 ; branch if failure of any type
                03    1F A6    91   0F0F  2691          cmpb    fab$b_rfm(r6), #fab$c_vfc ; vfc record format?
                         0D    12   0F13  2692          bneq    110$                    ; nope
          50    00000000'8F    D0   0F15  2693          movl    #rms$_fsz, r0           ; pre-set bad vfc size error
                0C    3F A6    91   0F1C  2694          cmpb    fab$b_fsz(r6), #input_vfc_siz ; vfc size correct?
                         14    1A   0F20  2695          bgtru   130$                    ; nope, go die with an error
                04    1F A6    91   0F22  2696  110$:   cmpb    fab$b_rfm(r6), #fab$c_stm ; stream record format?
                         04    12   0F26  2697          bneq    120$                    ; nope
                1E A6    02    88   0F28  2698          bisb    #fab$m_cr, fab$b_rat(r6) ; always say implied lf/cr for stream
                                  0F2C  2699  120$:   $connect -                        ; connect
                                  0F2C  2700                      rab=@fab$l_tecrab(r6) ;  the correct rab
                      54 B6    DF   0F2C                PUSHAL  @fab$l_tecrab(r6)
          00000000'GF    01    FB   0F2F                CALLS   #$$.TMP1,G^SYS$CONNECT
                   F1AB       30   0F36  2701  130$:   bsbw    success_or_cls          ; check for success completion
             0A 50 A6    06    E1   0F39  2702          bbc     #fab$v_tecfmt, fab$l_tecsts(r6), 140$ ; any format options?
             1E A6    F7 8F    8A   0F3E  2703          bicb    #^c<fab$m_blk>, fab$b_rat(r6) ; yes, clear all but this
             1E A6    52 A6    88   0F43  2704          bisb    fab$l_tecsts+2(r6), fab$b_rat(r6) ;  then apply options
             58 A6    58 A6    7E   0F48  2705  140$:   movaq   fab$q_tecque(r6), fab$q_tecque(r6) ; initialize the
             5C A6    58 A6    7E   0F4D  2706          movaq   fab$q_tecque(r6), fab$q_tecque+4(r6) ;  data buffer queue
                   67    56    D0   0F52  2707          movl    r6, (r7)                ; set file as open
                FFF7 8F    08 AA    B1   0F55  2708          cmpw    i_r2(r10), #^a/I/-^a/R/ ; what is it?
                                  0A   19   0F5B  2709          blss    160$                    ; it's EB
                                  04   14   0F5D  2710          bgtr    150$                    ; it's ER
                      00'AB    57   B0   0F5F  2711          movw    r7, b^indir(r11)        ; it's EI, set indirect as active
                         FD76      30   0F63  2712  150$:   bsbw    set_filename            ; set the file's name, etc.
                                  05   0F66  2713          rsb                             ;  and exit
                                  0F67  2714
                      50    28 A6    D0   0F67  2715  160$:   movl    fab$l_nam(r6), r0       ; get pointer to nam from fab
                      51    03 A0    9A   0F6B  2716          movzbl  nam$b_rsl(r0), r1       ;  and get resultant filename's length
                   08EC'CF    51    90   0F6F  2717          movb    r1, w^file_spec_len     ; set a new file spec length
          00FF 8F    00    04 B0    51   2C   0F74  2718          movc5   r1, @nam$l_rsa(r0), #0, - ; move input's resultant filename
                         08ED'CF          0F7C
                                  0F7F  2719                      #nam$c_maxrss, w^file_spec_buf ;  into file spec buffer
                                  0F7F  2720          bs      #fab$v_tecnv, w^file_spec_opt ; ensure maximized version
                   00 08E0'CF    0A   E3   0F7F                bbcs    #fab$v_tecnv, w^file_spec_opt, 30021$
                                  0F85        30021$:
                      57    00'AB    3C   0F85  2721  170$:   movzwl  b^oupntr(r11), r7       ; get place to store fab pointer
                   56    0274'CF    DE   0F89  2722          moval   w^output_nor_fab, r6    ; guess at normal output
                   0000'8F    57    B1   0F8E  2723          cmpw    r7, #oupnor             ; good guess?
                                  05   13   0F93  2724          beql    180$                    ; yep
                   56    05E8'CF    DE   0F95  2725          moval   w^output_alt_fab, r6    ; nope, alternate output
                50 A6    08E0'CF    D0   0F9A  2726  180$:   movl    w^file_spec_opt, fab$l_tecsts(r6) ; set file spec options
             04 A6    10000044 8F    D0   0FA0  2727          movl    #fab$m_sqo!fab$m_sup!fab$m_tef, fab$l_fop(r6) ; reset options
                         36 A6    B4   0FA8  2728          clrw    fab$w_mrs(r6)           ; maximum record size = 0 for variable
```

```
                1F A6    02    90   0FAB   2729           movb    #fab$c_var, fab$b_rfm(r6) ; guess at variable record format
                1E A6    02    90   0FAF   2730           movb    #fab$m_cr, fab$b_rat(r6) ; guess at implied lf/cr records
                      24 A6    D4   0FB3   2731           clrl    fab$l_xab(r6)             ; guess at no specific protection code
          17 A6       42 8F    90   0FB6   2732           movb    #fab$m_get!fab$m_upi, fab$b_shr(r6) ; set sharing to gets
          05 50 A6       07    E1   0FBB   2733           bbc     #fab$v_tecrw, fab$l_tecsts(r6), 190$ ; /rw?
                                     0FC0   2734           bs      #fab$v_rwo, fab$l_fop(r6) ; set rewind before open
          00 04 A6       07    E3   0FC0                  bbcs    #fab$v_rwo, fab$l_fop(r6), 30022$
                                     0FC5          30022$:
          05 50 A6       08    E1   0FC5   2735  190$:    bbc     #fab$v_tecsh, fab$l_tecsts(r6), 200$ ; /sh?
          17 A6       43 8F    90   0FCA   2736           movb    #fab$m_get!fab$m_put!fab$m_upi, fab$b_shr(r6) ; set sharing
          04 50 A6       0A    E1   0FCF   2737  200$:    bbc     #fab$v_tecnv, fab$l_tecsts(r6), 210$ ; /nv?
                                     0FD4   2738           bs      #fab$v_mxv, fab$l_fop(r6) ; set maximized version numbers
                04 A6    02    88   0FD4                  bisb    s^#1@fab$v_mxv, fab$l_fop(r6)
                50    00'AB    3C   0FD8   2739  210$:    movzwl  b^inpntr(r11), r0         ; get input file's pointer
                   50    60    D0   0FDC   2740           movl    (r0), r0                  ;  to get its fab pointer
                         22    13   0FDF   2741           beql    230$                      ; no input file
          04 1E A0       00    E1   0FE1   2742           bbc     #fab$v_ftn, fab$b_rat(r0), 220$ ; really want fortran ccl?
                1E A6    01    90   0FE6   2743           movb    #fab$m_ftn, fab$b_rat(r6) ; yes, so set it
                   08 AA    B5   0FEA   2744  220$:    tstw    i_r2(r10)                 ; is it EB or EW?
                         14    14   0FED   2745           bgtr    230$                      ; it's EW
                51    24 A0    D0   0FEF   2746           movl    fab$l_xab(r0), r1         ; it's EB, get input's protection xab
    08 A1          00F0 8F    AA   0FF3   2747           bicw    #<xab$m_noread! -         ;  clear no read
                                     0FF9   2748                   xab$m_nowrite! -          ;   and no write
                                     0FF9   2749                   xab$m_noexe! -            ;    and no execute
                                     0FF9   2750                   xab$m_nodel>@xab$v_own, - ;     and no delete @ owner spot
                                     0FF9   2751                         xab$w_pro(r1)       ; to ensure owner stuff is o.k.
                24 A6    51    D0   0FF9   2752           movl    r1, fab$l_xab(r6)         ; now use that protection code
                04    1F A0    91   0FFD   2753           cmpb    fab$b_rfm(r0), #fab$c_stm ; input in stream record format?
                         05    13   1001   2754           beql    240$                      ; yep, go default output to stream
             08 50 A6       0B    E1   1003   2755  230$:    bbc     #fab$v_tecstm, fab$l_tecsts(r6), 250$ ; stream format option?
                1F A6       04    90   1008   2756  240$:    movb    #fab$c_stm, fab$b_rfm(r6) ; set output for stream format
                1E A6       02    90   100C   2757           movb    #fab$m_cr, fab$b_rat(r6) ;  with implied lf/cr records
             04 50 A6       0C    E1   1010   2758  250$:    bbc     #fab$v_tecvar, fab$l_tecsts(r6), 260$ ; variable format option?
                1F A6       02    90   1015   2759           movb    #fab$c_var, fab$b_rfm(r6) ; set output for variable format
             05 50 A6       06    E1   1019   2760  260$:    bbc     #fab$v_tecfmt, fab$l_tecsts(r6), 270$ ; any format options?
          1E A6    52 A6    90   101E   2761           movb    fab$l_tecsts+2(r6), fab$b_rat(r6) ; yes, apply options
       34 A6    08EC'CF    90   1023   2762  270$:    movb    w^file_spec_len, fab$b_fns(r6) ; set file spec's length
                                     1029   2763           $create -                 ; create
                                     1029   2764                   fab=(r6)          ;  the output file
                         66    DF   1029                  PUSHAL  (r6)
    00000000'GF          01    FB   102B                  CALLS   #$$.TMP1,G^SYS$CREATE
     010C'CF       0C A6    D0   1032   2765           movl    fab$l_stv(r6), w^err_msgvec+8 ; save the STV value
           52      24 A6    D0   1038   2766           movl    fab$l_xab(r6), r2         ; save protection xab use indicator
                   24 A6    D4   103C   2767           clrl    fab$l_xab(r6)             ;  then remove the protection xab
                      52    D5   103F   2768           tstl    r2                        ; did we use a protection xab?
                         09    13   1041   2769           beql    280$                      ; nope
    00000000'8F          50    D1   1043   2770           cmpl    r0, #rms$_prv             ; yep, a privilege violation error?
                         D7    13   104A   2771           beql    270$                      ; that we did, go try, try again...
                                     104C   2772  280$:    bs      #fab$v_dlt, fab$l_fop(r6) ; (pre-)mark file for deletion
             00 04 A6    0F    E3   104C                   bbcs    #fab$v_dlt, fab$l_fop(r6), 30024$
                                     1051          30024$:
    00000000'8F          50    D1   1051   2773           cmpl    r0, #rms$_supersede       ; did we supersede something?
                         03    13   1058   2774           beql    290$                      ; yes, that's o.k.
                   F0A1    30   105A   2775           bsbw    success_or_err            ; else check for success completion
                                     105D   2776  290$:    $connect -                ; connect
                                     105D   2777                   rab=@fab$l_tecrab(r6) ;  the correct rab
                   54 B6    DF   105D                  PUSHAL  @fab$l_tecrab(r6)
```

```
                00000000'GF    01   FB  1060                    CALLS   #$$.TMP1,G^SYS$CONNECT
                          FO7A 30   1067  2778                    bsbw    success_or_cls          ; check for success completion
                   58 A6   58 A6  7E   106A  2779                    movaq   fab$q_tecque(r6), fab$q_tecque(r6) ; initialize the
                   5C A6   58 A6  7E   106F  2780                    movaq   fab$q_tecque(r6), fab$q_tecque+4(r6) ;  data buffer queue
                        67   56  D0   1074  2781                    movl    r6, (r7)                ; set file as open
                             FC37  31   1077  2782                    brw     set_outputname          ; set the file's name, etc. & exit
                                   107A  2783
                                   107A  2784 .disable lsb
                                   107A  2785
                                   107A  2786 fetch_filbuf:                                       ; fetch (converted) filename buffer
    00   00000000'EF    0000'8F    2F   107A  2787                    movtuc  #filsiz, filsrt, -      ; move translated from TECO's buffer
08ED'CF    00FF 8F    0100'CF         1084
                                   108D  2788                    #0, w^file_spec_table, - ;  'til 0(end), 128(spec), 255(end?)
                                   108D  2789                    #nam$c_maxrss, w^file_spec_buf ;  into our file spec buffer
                             29    1C   108D  2790                    bvc     30$                     ; terminator not seen??
                        80 8F    81   91   108F  2791 10$:    cmpb    (r1)+, #128             ; is it the special (128)?
                             23    12   1093  2792                    bneq    30$                     ; nope, other, call it the end
                             50    D7   1095  2793                    decl    r0                      ; yep, remove the special from count
                             1F    13   1097  2794                    beql    30$                     ; nothing left??
                   65   81    80 8F    81   1099  2795                    addb3   #128, (r1)+, (r5)       ; convert next to (almost) proper code
                        09 85         05   E5   109E  2796                    bbcc    #5, (r5)+, 20$          ; now do an (almost) final correction
                           05 FF      A5   E9   10A2  2797                    blbc    -1(r5), 20$             ; have we (finally) done it?
                      FF A5   FF 8F    90   10A6  2798                    movb    #255, -1(r5)            ; nope, this is the last fix!
                             50    D7   10AB  2799 20$:    decl    r0                      ; remove modified code from count
                             54    D7   10AD  2800                    decl    r4                      ; say we've store another character
65   54   63   00   61         50    2F   10AF  2801                    movtuc  r0, (r1), #0, (r3), r4, (r5) ; translated move rest of data
                             D7    1D   10B6  2802                    bvs     10$                     ;  and loop if another hit...
                             65    94   10B8  2803 30$:    clrb    (r5)                    ; ensure .asciz format
              08EC'CF    FF 8F    54   83   10BA  2804                    subb3   r4, #nam$c_maxrss, w^file_spec_len ; form file spec length
                             05    10C1  2805                    rsb                             ; exit w/ Z=1 if zero length
                                   10C2  2806
                                   10C2  2807 get_file_char:                                      ; get file specification character
                   50    82    9A   10C2  2808 10$:    movzbl  (r2)+, r0               ; get next byte from filename buffer
                         27    13   10C5  2809                    beql    40$                     ; null, the end, exit 'beql'
                      1A 53    E8   10C7  2810                    blbs    r3, 20$                 ; branch if inside of quotes
                   20    50    91   10CA  2811                    cmpb    r0, #32                 ; is this a junky character?
                         F3    1B   10CD  2812                    blequ   10$                     ; yes, ignore it
                7F 8F    50    91   10CF  2813                    cmpb    r0, #127                ; other kind of junky character?
                         ED    13   10D3  2814                    beql    10$                     ; yes, ignore it
                61 8F    50    91   10D5  2815                    cmpb    r0, #^a/A/+32           ; is it lower case?
                         09    1F   10D9  2816                    blssu   20$                     ; not lower case
                7A 8F    50    91   10DB  2817                    cmpb    r0, #^a/Z/+32           ; might be
                         03    1A   10DF  2818                    bgtru   20$                     ; but it isn't...
                   50    20    8A   10E1  2819                    bicb    #32, r0                 ; lower case, make into upper case
                   22    50    91   10E4  2820 20$:    cmpb    r0, #^a/"/              ; changing quote mode?
                         03    12   10E7  2821                    bneq    30$                     ; nope
                   53    01    CC   10E9  2822                    xorl    #1, r3                  ; yep, so flip the flop
                         50    95   10EC  2823 30$:    tstb    r0                      ; ensure 'bneq' exit
                               05   10EE  2824 40$:    rsb                             ; exit
```

```
                        10EF  2826 .sbttl  Do "en" processing
                        10EF  2827
                        10EF  2828 .enable lsb
                        10EF  2829
                        10EF  2830 en_preset:                              ; preset the "en" specification
    56   06EC'CF    DE  10EF  2831          moval   w^en_fab, r6           ; get the "en" fab pointer
 34 A6   08EC'CF    90  10F4  2832          movb    w^file_spec_len, fab$b_fns(r6) ; set file spec's length
                        10FA  2833          $parse -                       ; parse the specification
                        10FA  2834                  fab=(r6)               ;  from the "en" fab
              66    DF  10FA                PUSHAL  (r6)
00000000'GF   01    FB  10FC                CALLS   #$$.TMP1,G^SYS$PARSE
 010C'CF   0C A6    D0  1103  2835          movl    fab$l_stv(r6), w^err_msgvec+8 ; save the STV value
      50 A6   01    D0  1109  2836 10$:     movl    #fab$m_teceof, fab$l_tecsts(r6) ; reset private status (fnf)
          EFEE      30  110D  2837          bsbw    success_or_err         ; check for success completion
        50 A6       D4  1110  2838          clrl    fab$l_tecsts(r6)       ; made it, mark as active
                    05  1113  2839          rsb                            ;  then exit
                        1114  2840
                        1114  2841 en_next:                               ; get next "en" occurance
    56   06EC'CF    DE  1114  2842          moval   w^en_fab, r6           ; get the "en" fab pointer
      34 50 A6      E8  1119  2843          blbs    fab$l_tecsts(r6), 50$  ; keep saying 'fnf' if we need to
                        111D  2844 .assume fab$m_teceof eq 1
                        111D  2845 20$:     $search -                      ; else search for next occurance
                        111D  2846                  fab=(r6)               ;  using the "en" fab
              66    DF  111D                PUSHAL  (r6)
00000000'GF   01    FB  111F                CALLS   #$$.TMP1,G^SYS$SEARCH
 010C'CF   0C A6    D0  1126  2847          movl    fab$l_stv(r6), w^err_msgvec+8 ; save the STV value
          03 50     E9  112C  2848          blbc    r0, 30$                ; branch if any type of error
           FBAA     31  112F  2849          brw     set_filename           ; set the resultant stuff & exit
                        1132  2850
00000000'8F   50    D1  1132  2851 30$:     cmpl    r0, #rms$_fnf          ; not found?
                13  13  1139  2852          beql    40$                    ; yes, so say so
00000000'8F   50    D1  113B  2853          cmpl    r0, #rms$_nmf          ; no more files?
                0A  13  1142  2854          beql    40$                    ; yes, so say so
            0C A6   D5  1144  2855          tstl    fab$l_stv(r6)          ; fatal type error?
              C0    12  1147  2856          bneq    10$                    ; yep, so die with it
           EF76     30  1149  2857          bsbw    success_or_announce    ; else just announce the error
              CF    11  114C  2858          brb     20$                    ;  and go try, try again...
                        114E  2859
            50 A6   D6  114E  2860 40$:     incl    fab$l_tecsts(r6)       ; keep saying 'fnf' from now on
                        1151  2861 .assume fab$m_teceof eq 1
                        1151  2862 50$:     err     FNF, <"File not found">
             EFF1   30  1151                 bsbw    err
              000003CF' 1154                 .long $$$$$$
                  27B6  03CF                 .word  $$$$$$
6F 66 20 74 6F 6E 20 65 6C 69 46 00' 03D1                .ascic "File not found"
                  64 6E 75  03DD
                     0E     03D1
                        1158  2863
                        1158  2864 .disable lsb
```

```
                        1158  2866 .sbttl  Handle the EJ flag
                        1158  2867
                        1158  2868 tec$ejflg:                                    ; get ej flag information
          56  0060'CF   9E  1158  2869          movab   w^tmp_string_buf+8, r6    ; point into temporary area
              66   7C   115D  2870          clrq    (r6)                       ; no return len(+8) & end-of-list(+12)
          57  00A0'CF   9E  115F  2871          movab   w^tmp_string2_buf, r7     ; point to result area
              76   57   D0  1164  2872          movl    r7, -(r6)                  ; set pointer to result buffer(+4)
          76  0319 8F   B0  1167  2873          movw    #jpi$_pid, -(r6)           ; guess at wanting process id(+2)
              58   05   D0  116C  2874          movl    #5, r8                     ; guess at -1ej (5 => vms)
                   6A   B5  116F  2875          tstw    i_r0(r10)                  ; what does the user really want?
                   37   19  1171  2876          blss    20$                        ; it is -1ej, return 5 for vms
                   0F   13  1173  2877          beql    10$                        ; it is 0ej, go fetch process id
          58  00FA'CF   3C  1175  2878          movzwl  w^ter_o_unit, r8           ; guess at 1ej, terminal unit
              02   6A   B1  117A  2879          cmpw    i_r0(r10), #2              ; what is it really?
                   2B   19  117D  2880          blss    20$                        ; it is 1ej, return terminal unit
          66  0304 8F   B0  117F  2881          movw    #jpi$_uic, (r6)            ; it is 2ej, fetch uic(+2)
              76   3F   B0  1184  2882 10$:     movw    #tmp_string2_siz, -(r6)    ; set size of result buffer(+0)
                        1187  2883          $getjpi_s -                        ; get job information
                        1187  2884                  itmlst=(r6)                ;  item list...
                   7E   7C  1187                  CLRQ    -(SP)
                   00   DD  1189                  PUSHL   #0
                   66   DF  118B                  PUSHAL  (r6)
                   00   DD  118D                  PUSHL   #0
                   00   DD  118F                  PUSHL   #0
                   00   DD  1191                  PUSHL   #0
    00000000'GF    07   FB  1193                  CALLS   #7,G^SYS$GETJPI
            EF3A   30  119A  2885          bsbw    success_or_abrt            ; check for success completion
              58   67   D0  119D  2886          movl    (r7), r8                   ; fetch result
                   6A   B5  11A0  2887          tstw    i_r0(r10)                  ; what was it?
                   06   13  11A2  2888          beql    20$                        ; process id (0ej)
58  08  08   02 A7   F0  11A4  2889          insv    2(r7), #8, #8, r8          ; uic, make into 2 bytes
              6A   58   3C  11AA  2890 20$:     movzwl  r8, i_r0(r10)              ; set the desired result
                   05   11AD  2891          rsb                                ; exit
```

```
                        11AE  2893 .sbttl  Handle line truncation mode changes
                        11AE  2894
                        11AE  2895 .enable lsb
                        11AE  2896
                        11AE  2897 tec$truln:                                   ; change truncate lines mode
            52   08  D0  11AE  2898         movl    #tec$v_et$tru, r2          ; set the ET bit to check
            53   09  D0  11B1  2899         movl    #tt$v_wrap, r3             ;  and corresponding characteristic
            50   6A  D2  11B4  2900         mcoml   i_r0(r10), r0              ; get the new bit(s) to check against
                 09  11  11B7  2901         brb     10$                        ; go join common code
                        11B9  2902
                        11B9  2903 .sbttl  Handle 8-bit terminal mode changes
                        11B9  2904
                        11B9  2905 tec$eight:                                   ; change 8-bit terminal mode
            52   0C  D0  11B9  2906         movl    #tec$v_et$8bt, r2          ; set the ET bit to check
            53   0F  D0  11BC  2907         movl    #tt$v_eightbit, r3         ;  and corresponding characteristic
            50   6A  D0  11BF  2908         movl    i_r0(r10), r0              ; get the new bit(s) to check against
        00F6'CF  B5  11C2  2909 10$:    tstw    w^ter_o_chan               ; is there a terminal output channel?
            57   13  11C6  2910         beql    40$                        ; nope, just go exit
        0004'CF  DD  11C8  2911         pushl   w^devdepend                ; save the original characteristics
  00 0004'CF     53  E5  11CC  2912         bbcc    r3, w^devdepend, 20$       ; ensure bit off in characteristics
      06 50      52  E1  11D2  2913 20$:     bbc     r2, r0, 30$                ; shound we set the characteristic?
  00 0004'CF     53  E2  11D6  2914         bbss    r3, w^devdepend, 30$       ; set bit on in characteristics
            15   10  11DC  2915 30$:     bsbb    tec$setmode                ; go set the terminal mode
    0004'CF     8E  D0  11DE  2916         movl    (sp)+, w^devdepend         ; restore original characteristics
          EF18  31  11E3  2917         brw     success_or_err             ; check for success completion & exit
                        11E6  2918
                        11E6  2919 .sbttl  Handle new terminal width
                        11E6  2920
                        11E6  2921 tec$width:                                   ; handle new terminal width
        00F6'CF  B5  11E6  2922         tstw    w^ter_o_chan               ; is there a terminal output channel?
            33   13  11EA  2923         beql    40$                        ; nope, just go exit
0002'CF 0000'CB  B0  11EC  2924         movw    htsize+rwsize(r11), w^devbufsiz ; set the new terminal width
                        11F3  2925 tec$setmode:                                 ; do a IO$_SETMODE on terminal output
          F048  30  11F3  2926         bsbw    tec$wait                   ; wait for all output to be queued
                        11F6  2927         $qiow_s -                          ; queue an I/O request
                        11F6  2928                 chan = w^ter_o_chan, -     ;  on the terminal output channel
                        11F6  2929                 func = s^#io$_setmode, - ;  doing a setmode
                        11F6  2930                 iosb = w^ter_o_pos, -      ;  using this as the IOSB
                        11F6  2931                 p1 = w^getdvi_info, -      ;  set the information in this buffer
                        11F6  2932                 p2 = #getdvi_info_len      ;  which is this long
            7E   7C  11F6                       CLRQ    -(SP)
            7E   7C  11F8                       CLRQ    -(SP)
            0C   DD  11FA                       PUSHL   #getdvi_info_len
        0000'CF  DF  11FC                       PUSHAL  w^getdvi_info
            7E   7C  1200                       CLRQ    -(SP)
        0010'CF  7F  1202                       PUSHAQ  w^ter_o_pos
        7E  00'  3C  1206                       MOVZWL  s^#io$_setmode,-(SP)
    7E  00F6'CF  3C  1209                       MOVZWL  w^ter_o_chan,-(SP)
            00   DD  120E                       PUSHL   #0
00000000'GF  0C  FB  1210                       CALLS   #12,G^SYS$QIOW
      05 50      E9  1217  2933         blbc    r0, 40$                    ; just exit if any error
    50  0010'CF  3C  121A  2934         movzwl  w^ter_o_pos, r0            ; else return the completion status
                 05  121F  2935 40$:    rsb                                ; exit
                        1220  2936
                        1220  2937 .disable lsb
```

```
                        1220  2939 .sbttl  Stop terminal hacks
                        1220  2940
                        1220  2941 .enable lsb
                        1220  2942
                        1220  2943 tec$xitnw:                                ; stop terminal hacks
    0100'CF    01    90 1220  2944          movb    #1, w^exiting_flag       ; exiting flag <- on
                        1225  2945 enable_ctrlt:                            ; (re-)enable CTRL/T actions
50   00000000'GF    9E 1225  2946          movab   g^lib$enable_ctrl, r0    ; address enable oob cli service
   51    08E8'CF    DE 122C  2947          moval   w^ter_oob_msk, r1        ; enable mask is saved old mask(s)
         00F4'CF    B5 1231  2948 10$:     tstw    w^ter_i_chan             ; do we have a terminal for input?
                15    13 1235  2949          beql    30$                      ; nope
                7E    DF 1237  2950          pushal  -(sp)                    ; arg #2 -> returned old mask
                61    DF 1239  2951          pushal  (r1)                     ; arg #1 -> enable/disable mask
             60    02    FB 123B  2952          calls   #2, (r0)                 ; call the enable/disable cli service
               05 50    E9 123E  2953          blbc    r0, 20$                  ; skip .OR. if an error occured...
    08E8'CF    6E    C8 1241  2954          bisl    (sp), w^ter_oob_msk      ; .OR. together old mask(s)
          5E    04    C0 1246  2955 20$:     addl    #4, sp                   ; clean up the stack
             EE76    30 1249  2956          bsbw    success_or_announce      ; announce any failure...
                05    124C  2957 30$:     rsb                              ; exit
                        124D  2958
                        124D  2959 not_exiting:                             ; say we're not really exiting
 F9 0100'CF    00    E5 124D  2960          bbcc    #0, w^exiting_flag, 30$  ; exiting flag <- off, exit if was off
50   00000000'GF    9E 1253  2961          movab   g^lib$disable_ctrl, r0   ; address disable oob cli service
51   00100000 8F    DE 125A  2962          moval   #lib$m_cli_ctrlt, r1     ; disable mask is for CTRL/T
                CE    11 1261  2963          brb     10$                      ; go do the CTRL/T disable
                        1263  2964
                        1263  2965 .disable lsb
```

```
                        1263  2967 .sbttl  Process special functions
                        1263  2968
                        1263  2969 .enable lsb
                        1263  2970
          50    87  9A  1263  2971 10$:    movzbl  (r7)+, r0               ; get a byte from the buffer
                1B  13  1266  2972         beql    40$                     ; end of buffer, exit Z=1
    61 8F       50  91  1268  2973         cmpb    r0, #^a/A/+32           ; should we convert case?
                09  1F  126C  2974         blssu   20$                     ; nope
    7A 8F       50  91  126E  2975         cmpb    r0, #^a/Z/+32           ; really?
                03  1A  1272  2976         bgtru   20$                     ; nope again
          50    20  8A  1274  2977         bicb    #32, r0                 ; yep, force upper case
          20    50  91  1277  2978 20$:    cmpb    r0, #32                 ; is it a space?
                07  12  127A  2979         bneq    40$                     ; nope, exit Z=0
          50    87  91  127C  2980 30$:    cmpb    (r7)+, r0               ; yep, is next also a space?
                FB  13  127F  2981         beql    30$                     ; multiple spaces, collapse them
                57  D7  1281  2982         decl    r7                      ; else back up the pre-fetch (& Z=0)
                05  1283  2983 40$:    rsb                             ; exit, Z=1 => end-of-buffer
                    1284  2984
      66    FF  8F  9A  1284  2985 50$:    movzbl  #nam$c_maxrss, (r6)     ; set maximum length of value
    04  A6    57  D0  1288  2986         movl    r7, 4(r6)               ; set address part of value desc
67      66    00  3A  128C  2987         locc    #0, (r6), (r7)          ; locate the terminating null
          66    50  C2  1290  2988         subl    r0, (r6)                ;  and find the true length
                66  7F  1293  2989         pushaq  (r6)                    ; arg #2 -> symbol's value
                68  7F  1295  2990         pushaq  (r8)                    ; arg #1 -> symbol's name
00000000'GF     02  FB  1297  2991         calls   #2, g^lib$set_symbol    ; call the set symbol cli service
            EE36    30  129E  2992 53$:    bsbw    success_or_abrt         ; check for success completion
            00E3    31  12A1  2993         brw     130$                    ; go move value to filename buffer
                    12A4  2994
                68  B5  12A4  2995 57$:    tstw    (r8)                    ; zero length symbol name?
                6A  13  12A6  2996         beql    90$                     ; yep, error, go return a value of 0
                66  3F  12A8  2997         pushaw  (r6)                    ; arg #3 -> symbol's value length
                66  7F  12AA  2998         pushaq  (r6)                    ; arg #2 -> symbol's value
                68  7F  12AC  2999         pushaq  (r8)                    ; arg #1 -> symbol's name
00000000'GF     03  FB  12AE  3000         calls   #3, g^lib$get_symbol    ; call the get symbol cli service
00000000'8F     50  D1  12B5  3001         cmpl    r0, #lib$_nosuchsym     ; did the symbol exist?
                E0  12  12BC  3002         bneq    53$                     ; yes, or other error...
            00C4    31  12BE  3003         brw     120$                    ; go move null to filename buffer
                    12C1  3004
                A0  10  12C1  3005 60$:    bsbb    10$                     ; get the next character
          20    50  91  12C3  3006         cmpb    r0, #32                 ; is it a space?
                4A  12  12C6  3007         bneq    90$                     ; nope, error, go return a value of 0
                    12C8  3008         getdesc tmp_string2, r8         ; reset & get desc for temp string #2
    58  00A0'CF  9E  12C8                  movab   w^tmp_string2_buf, r8
          78    58  D0  12CD                  movl    r8, -(r8)
          78    3F  3C  12D0                  movzwl  #tmp_string2_siz, -(r8)
          51    88  D0  12D3  3009         movl    (r8)+, r1               ; get length of symbol's name buffer
          52    68  D0  12D6  3010         movl    (r8), r2                ;  and get a pointer to it's storage
                78  D4  12D9  3011         clrl    -(r8)                   ; reset length of symbol's name desc
                51  D7  12DB  3012         decl    r1                      ; we store before check; fix length...
                84  10  12DD  3013 70$:    bsbb    10$                     ; get next character of symbol's name
                C3  13  12DF  3014         beql    57$                     ; end, must be reading the symbol
          20    50  91  12E1  3015         cmpb    r0, #32                 ; is it a space?
                9E  13  12E4  3016         beql    50$                     ; yep, that's the name's end
          82    50  90  12E6  3017         movb    r0, (r2)+               ; store the symbol's name character
FFEE 68   01    51  F1  12E9  3018         acbl    r1, #1, (r8), 70$       ; bump length and check for too far
                21  11  12EF  3019         brb     90$                     ; it's too far, go return a value of 0
                    12F1  3020
```

```
                        12F1  3021 tec$gexit:                          ; process special functions
          FD86    30    12F1  3022          bsbw    fetch_filbuf       ; go fetch the filename buffer
57  08ED'CF     9E    12F4  3023          movab   w^file_spec_buf, r7  ; address the fetched data
            FF67  30    12F9  3024          bsbw    10$                ; get the first command byte
              14  13    12FC  3025          beql    90$                ; the end, go exit n=0
56  02AD'CF     9E    12FE  3026          movab   w^colon_eg_list-3, r6 ; get (biased) list of functions
        56  03  C0    1303  3027 80$:      addl    #3, r6             ; skip uncompared characters
        58  86  D0    1306  3028          movl    (r6)+, r8          ; pick up pointer to logical name desc
        86  50  91    1309  3029          cmpb    r0, (r6)+          ; could it be this function?
            08  13    130C  3030          beql    100$               ; might be...
            58  D5    130E  3031          tstl    r8                 ; more to check?
            F1  12    1310  3032          bneq    80$                ; yep, loop...
          00'AB B4    1312  3033 90$:     clrw    b^n(r11)           ; set return value of 0
                05    1315  3034          rsb                        ; exit
                      1316  3035
            FF4A  30  1316  3036 100$:    bsbw    10$                ; get next character of name
              F7  13  1319  3037          beql    90$                ; no more, go return value of 0
        86  50  91    131B  3038          cmpb    r0, (r6)+          ; is the character correct?
            F2  12    131E  3039          bneq    90$                ; nope, go return value of 0
            66  95    1320  3040          tstb    (r6)               ; yep, more to check?
            F2  12    1322  3041          bneq    100$               ; there's more, loop...
                      1324  3042          getdesc tmp_string, r6     ; reset & get desc for temp string
56  0058'CF     9E    1324                 movab   w^tmp_string_buf, r6
        76  56  D0    1329                 movl    r6, -(r6)
        76  3F  3C    132C                 movzwl  #tmp_string_siz, -(r6)
            58  D5    132F  3043          tstl    r8                 ; is it DCL symbol manipulation?
            8E  13    1331  3044          beql    60$                ; yes
              FF2D  30  1333  3045          bsbw    10$                ; get the next character
                30  13  1336  3046          beql    110$               ; no more, go read the logical
          20    50  91  1338  3047          cmpb    r0, #32            ; another, is it a space?
                D5  12  133B  3048          bneq    90$                ; nope, error, go return a value of 0
          86    3F  3C  133D  3049          movzwl  #63, (r6)+         ; set maximum string size
          66    57  D0  1340  3050          movl    r7, (r6)           ;  and pointer to string
      67  3F    00  3A  1343  3051          locc    #0, #63, (r7)      ; locate the terminating null
          76    50  C2  1347  3052          subl    r0, -(r6)          ;  then correct the size
                4B  13  134A  3053          beql    150$               ; size is zero, go delete the logical
              87'AF 9F  134C  3054          pushab  b^130$             ; return new value string upon exit
          7E    68  7D  134F  3055          movq    (r8), -(sp)        ; set logical name string desc
          7E    66  7D  1352  3056          movq    (r6), -(sp)        ; set value string desc
          6E    7F  1355  3057          pushaq  (sp)               ; arg #2 -> value string
            0C  AE  7F  1357  3058          pushaq  4+8(sp)            ; arg #1 -> logical name
00000000'GF     02  FB  135A  3059          calls   #2, g^lib$set_logical ; call the set logical cli service
    52  0000'8F     3C  1361  3060          movzwl  #ss$_supersede, r2 ; we ignore supersede of an old string
                45  11  1366  3061          brb     160$               ; now go check it
                        1368  3062
                        1368  3063 110$:    $trnlog_s -                ; translate
                        1368  3064                    lognam=(r8), -     ;  teco's logical
                        1368  3065                    rsllen=(r6), -     ;  setting result length here
                        1368  3066                    rslbuf=(r6)        ;  which is the result buffer
                00  DD  1368                         PUSHL   #0
                00  DD  136A                         PUSHL   #0
                00  DD  136C                         PUSHL   #0
                66  7F  136E                         PUSHAQ  (r6)
                66  3F  1370                         PUSHAW  (r6)
                68  7F  1372                         PUSHAQ  (r8)
00000000'GF     06  FB  1374                         CALLS   #6,G^SYS$TRNLOG
              ED59  30  137B  3067          bsbw    success_or_abrt    ; check for success completion
```

```
                0000'8F    50   B1  137E  3068            cmpw    r0, #ss$_notran         ; did any translation happen?
                           02   12  1383  3069            bneq    130$                    ; yep, return the result string
                           66   D4  1385  3070 120$:      clrl    (r6)                    ; set result string to null
    FFFE'8F    00   04 B6    66   2C  1387  3071 130$:      movc5   (r6), @4(r6), #0, #filsiz-1-1, filsrt ; move to filename buffer
                     00000000'EF      138F
                           63   94  1394  3072            clrb    (r3)                    ; ensure answer is .asciz
                           05   1396  3073 140$:      rsb                             ; exit with n=-1 for success
                                1397  3074
                      EB   AF   9F  1397  3075 150$:      pushab  120$                    ; return null value string upon exit
                           7E   7C  139A  3076            clrq    -(sp)                   ; dummies to be removed below...
                      7E   68   7D  139C  3077            movq    (r8), -(sp)             ; set logical name string desc
                           6E   7F  139F  3078            pushaq  (sp)                    ; arg #1 -> logical name
                00000000'GF   01   FB  13A1  3079            calls   #1, g^lib$delete_logical ; call the delete logical cli service
                     52   0000'8F   3C  13A8  3080            movzwl  #ss$_nolognam, r2       ; we ignore no logical to delete
                          5E   10   C0  13AD  3081 160$:      addl    #2*8, sp                ; remove the stacked descriptors
                          52   50   D1  13B0  3082            cmpl    r0, r2                  ; is this the error we ignore?
                               E1   13  13B3  3083            beql    140$                    ; yep, just go exit
                          ED1F      31  13B5  3084            brw     success_or_abrt         ; check for success completion & exit
                                    13B8  3085
                                    13B8  3086 .disable lsb
```

```
                                13B8  3088 .sbttl  Get additional memory
                                13B8  3089
                                13B8  3090 .enable lsb
                                13B8  3091
                     24 AA   D6 13B8  3092 10$:    incl    i_ps(r10)               ; indicate failure with c=1
                             05 13BB  3093         rsb                             ;  and exit with failure
                                13BC  3094
                                13BC  3095 tec$sizer:                              ; get additional memory
     51   04 AA    000001FF 8F  C1 13BC  3096         addl3   #512-1, i_r1(r10), r1  ; get the request amount
               51    000001FF 8F  CA 13C5  3097         bicl    #512-1, r1             ;  rounded up to a page multiple
                  00E8'CF    51  D1 13CC  3098         cmpl    r1, w^still_free       ; is there room left?
                             E5  1A 13D1  3099         bgtru   10$                    ; nope, we must fail
                  00E8'CF    51  C2 13D3  3100         subl    r1, w^still_free       ; move the requested amount
                    00'AB    51  A0 13D8  3101         addw    r1, b^curfre(r11)      ;  to teco's free space
                 6B 00'AB    07  E0 13DC  3102         bbs     #tec$v_et$xit, b^etype(r11), 40$ ; doing announcements?
               0000'8F    00'AB  B1 13E1  3103         cmpw    b^qcmnd(r11), #cmdqrg  ; are we within a macro?
                             63  12 13E7  3104         bneq    40$                    ; skip announcements if so
                  5E 0100'CF     E8 13E9  3105         blbs    w^exiting_flag, 40$    ; also skip announcements if exiting
                       00'AB     B5 13EE  3106         tstw    b^indir(r11)           ; from within indirect command file?
                             59  12 13F1  3107         bneq    40$                    ; skip announcements if true
                                13F3  3108         bs      s^#io$v_canctrlo, ctlofg ; do a control/o cancel
          00 00000000'EF    00'  E3 13F3                bbcs    s^#io$v_canctrlo, ctlofg, 30025$
                                13FB  3109 30025$:
                  56    5B 8F  9A 13FB  3110         movzbl  #^a/[/, r6             ; start with "["
                       F292  30 13FF  3111         bsbw    echo_buffer            ;  in the message
           56    00'AB    00'AB  A1 1402  3112         addw3   b^qrstor(r11), b^qmax(r11), r6 ; find end of
                    56    00'AB  A0 1408  3113         addw    b^curfre(r11), r6      ;  total space in use
                          56  5B  A2 140C  3114         subw    r11, r6                ; now find total byte size in use
                          56  56  3C 140F  3115         movzwl  r6, r6                 ; now convert that to a longword
           56    000001FF 8F  C0 1412  3116         addl    #511, r6               ; round and
           56    00000200 8F  C6 1419  3117         divl    #512, r6               ;  find that size in pages
                             57  D4 1420  3118         clrl    r7                     ; clear the high order
     57    56    56    0A   7B 1422  3119         ediv    #10, r6, r6, r7        ;  and divide to tens/units
                             11  13 1427  3120         beql    30$                    ; no tens
           58    56    0A   C3 1429  3121         subl3   #10, r6, r8            ; are tens really up to hundreds?
                             09  19 142D  3122         blss    20$                    ; nope
                          56  31  9A 142F  3123         movzbl  #^a/1/, r6             ; yep, set a "1" for the hundreds
                       F25F  30 1432  3124         bsbw    echo_buffer            ;  and go output it
                          56  58  D0 1435  3125         movl    r8, r6                 ; restore corrected tens
                             13  10 1438  3126 20$:    bsbb    50$                    ; output result (tens)
                          56  57  90 143A  3127 30$:    movb    r7, r6                 ; copy the remainder (units)
                             0E  10 143D  3128         bsbb    50$                    ;  and go output it
                    0342'CF    7F 143F  3129         pushaq  w^sizing_msg_desc      ; set the rest of the sizing message
                F218 CF    01  FB 1443  3130         calls   #1, w^tec$out_ascid    ;  and go output it
                  00'AB    01  A8 1448  3131         bisw    #1, b^outdne(r11)      ; say memory message output done
                             05  144C  3132 40$:    rsb                            ; exit
                                144D  3133
                          56  30  80 144D  3134 50$:    addb    #^a/0/, r6             ; make character into a digit
                             F241  31 1450  3135         brw     echo_buffer            ;  and, then, go buffer it
                                1453  3136
                                1453  3137 .disable lsb
```

```
                        1453  3138 .sbttl  Get date and time
                        1453  3139
                        1453  3140 .enable lsb
                        1453  3141
                        1453  3142 tec$date:                                  ; get date
                  2A 10 1453  3143          bsbb    20$                       ; go get the information
57  86  076C  8F  A3  1455  3144          subw3   #1900, (r6)+, r7          ; get year-1900
          57  10  A4  145B  3145          mulw    #16, r7                   ;  then multiply it by 16
          57  86  A0  145E  3146          addw    (r6)+, r7                 ; add in the month
          57  20  A4  1461  3147          mulw    #32, r7                   ;  then multiply by 32
          57  86  A0  1464  3148 10$:     addw    (r6)+, r7                 ; finally, add in the day or seconds/2
          6A  57  3C  1467  3149          movzwl  r7, i_r0(r10)             ; return the result
              05  146A  3150          rsb                               ;  and exit
                  146B  3151
                  146B  3152 tec$time:                                    ; get time
              12  10  146B  3153          bsbb    20$                       ; go get the information
          56  06  C0  146D  3154          addl    #6, r6                    ; skip over year, month, day
      57  86  3C  A5  1470  3155          mulw3   #60, (r6)+, r7            ; get hour*60
          57  86  A0  1474  3156          addw    (r6)+, r7                 ; add in minutes
          57  1E  A4  1477  3157          mulw    #60/2, r7                 ;  then multiply by 30
          66  02  A6  147A  3158          divw    #2, (r6)                  ; form seconds/2
              E5  11  147D  3159          brb     10$                       ; go add in seconds/2 and exit
                  147F  3160
      56  0058'CF  9E  147F  3161 20$:     movab   w^tmp_string_buf, r6      ; get pointer to temporary area
                  1484  3162          $numtim_s -                           ; get numeric date/time
                  1484  3163                  timbuf=(r6)                   ;  into temporary area
                00  DD  1484                  PUSHL   #0
                66  DF  1486                  PUSHAL  (r6)
00000000'GF     02  FB  1488                  CALLS   #2,G^SYS$NUMTIM
            EC45  31  148F  3164          brw     success_or_abrt           ; check for success completion
                  1492  3165
                  1492  3166 .disable lsb
```

```
                        1492  3168 .sbttl  Exit from TECO
                        1492  3169
                        1492  3170 .enable lsb
                        1492  3171
                 1B  13 1492  3172 10$:    beql    20$                         ; no file
                        1494  3173          $close -                          ; else close
                        1494  3174                  fab=(r6)                  ;  the file
                 66  DF 1494             PUSHAL  (r6)
00000000'GF      01  FB 1496             CALLS   #$$.TMP1,G^SYS$CLOSE
  010C'CF    0C  A6  D0 149D  3175       movl    fab$l_sts(r6), w^err_msgvec+8 ; save the STV value
00000000'8F      50  D1 14A3  3176       cmpl    r0, #rms$_ifi               ; was file already closed (bad IFI)?
                 03  13 14AA  3177       beql    20$                         ; yes, don't call it an error...
               EC13  30 14AC  3178       bsbw    success_or_announce         ; announce any failure...
                 05  14AF  3179 20$:     rsb                                 ; exit
                     14B0  3180
                     14B0  3181 tec$texit:                                   ; exit from teco
               F33D  30 14B0  3182       bsbw    echo_dump                   ; dump out any partial terminal output
56   00000000'EF      D0 14B3  3183       movl    cmdprm, r6                  ; get indirect file fab pointer
                 D6  10 14BA  3184       bsbb    10$                         ;  and close it if necessary
56   00000000'EF      D0 14BC  3185       movl    inpnor, r6                  ; get normal input fab pointer
                 CD  10 14C3  3186       bsbb    10$                         ;  and close it if necessary
56   00000000'EF      D0 14C5  3187       movl    inpalt, r6                  ; get alternate input fab pointer
                 C4  10 14CC  3188       bsbb    10$                         ;  and close it if necessary
56   00000000'EF      D0 14CE  3189       movl    oupnor, r6                  ; get normal output fab pointer
                 BB  10 14D5  3190       bsbb    10$                         ;  and close it if necessary
56   00000000'EF      D0 14D7  3191       movl    oupalt, r6                  ; get alternate output fab pointer
                 B2  10 14DE  3192       bsbb    10$                         ;  and close it if necessary
     50   00F8'CF     3C 14E0  3193       movzwl  w^ter_c_chan, r0            ; get terminal control/c ast channel
                 0D  13 14E5  3194       beql    30$                         ; none
                        14E7  3195       $dassgn_s -                         ; deassign channel
                        14E7  3196                chan=r0                    ;  from control/c ast channel
            7E   50  3C 14E7             MOVZWL  r0,-(SP)
00000000'GF      01  FB 14EA             CALLS   #1,G^SYS$DASSGN
               EBCE  30 14F1  3197       bsbw    success_or_announce         ; announce any failure...
     50   00F4'CF     3C 14F4  3198 30$:  movzwl  w^ter_i_chan, r0            ; get terminal input channel
                 12  13 14F9  3199       beql    40$                         ; none
                     14FB  3200          $dassgn_s -                         ; deassign channel
                     14FB  3201                  chan=r0                   ;  from terminal input
            7E   50  3C 14FB             MOVZWL  r0,-(SP)
00000000'GF      01  FB 14FE             CALLS   #1,G^SYS$DASSGN
               EBBA  30 1505  3202       bsbw    success_or_announce         ; announce any failure...
               FD1A  30 1508  3203       bsbw    enable_ctrlt                ; go (re-)enable CTRL/T actions
                 0E  11 150B  3204       brb     50$                         ; continue
                     150D  3205
                     150D  3206 40$:     $close -                            ; close
                     150D  3207                  fab=w^input_sys_fab       ;  sys$input
         07A0'CF     DF 150D             PUSHAL  w^input_sys_fab
00000000'GF      01  FB 1511             CALLS   #$$.TMP1,G^SYS$CLOSE
               EBA7  30 1518  3208       bsbw    success_or_announce         ; announce any failure...
         00F6'CF     B5 151B  3209 50$:  tstw    w^ter_o_chan                ; a terminal output channel?
                 1A  13 151F  3210       beql    60$                         ; none
               ECEE  30 1521  3211       bsbw    tec$wait_done               ; wait for terminal output to complete
               FCCC  30 1524  3212       bsbw    tec$setmode                 ; go (re-)set correct terminal modes
               EB98  30 1527  3213       bsbw    success_or_announce         ; announce any failure...
                     152A  3214          $dassgn_s -                         ; deassign channel
                     152A  3215                  chan=w^ter_o_chan         ;  from terminal output
      7E   00F6'CF     3C 152A             MOVZWL  w^ter_o_chan,-(SP)
```

```
  00000000'GF    01   FB  152F                    CALLS   #1,G^SYS$DASSGN
           EB89  30   1536  3216          bsbw    success_or_announce     ; announce any failure...
             3A  11   1539  3217          brb     90$                     ; continue
                      153B  3218
        08BE'CF  B5   153B  3219 60$:     tstw    w^output_sys_rab+rab$w_rsz ; any remaining record length?
             11  12   153F  3220          bneq    70$                     ; yes, so dump the record
     01  00F2'CF  B1   1541  3221          cmpw    w^output_sys_vfc, #1    ; any remaining print control?
             1F  1B   1546  3222          blequ   80$                     ; none(0) or final NL(1), forget it
        00F2'CF  97   1548  3223          decb    w^output_sys_vfc        ; remove any final NL request
             04  18   154C  3224          bgeq    70$                     ; all is still o.k.
        00F2'CF  94   154E  3225          clrb    w^output_sys_vfc        ; else say no prefixing at all
08C4'CF  1E00'CF  9E   1552  3226 70$:     movab   w^output_sys_buf, w^output_sys_rab+rab$l_rbf ; reset pointer
                      1559  3227          $put -                          ; put a record
                      1559  3228                  rab=w^output_sys_rab    ;  to sys$output
        089C'CF  DF   1559                PUSHAL  w^output_sys_rab
  00000000'GF    01   FB  155D                CALLS   #$$.TMP1,G^SYS$PUT
           EB5B  30   1564  3229          bsbw    success_or_announce     ; announce any failure...
                      1567  3230 80$:     $close -                        ; close
                      1567  3231                  fab=w^output_sys_fab    ;  sys$output
        084C'CF  DF   1567                PUSHAL  w^output_sys_fab
  00000000'GF    01   FB  156B                CALLS   #$$.TMP1,G^SYS$CLOSE
           EB4D  30   1572  3232          bsbw    success_or_announce     ; announce any failure...
           FB02  30   1575  3233 90$:     bsbw    fetch_filbuf            ; go fetch the filename buffer
             18  13   1578  3234          beql    100$                    ; null string, just exit
        08ED'CF  9F   157A  3235          pushab  w^file_spec_buf         ; stack descriptor
     7E  08EC'CF  9A   157E  3236          movzbl  w^file_spec_len, -(sp)  ;  of command string
             6E  7F   1583  3237          pushaq  (sp)                    ; arg #1 -> command string
  00000000'GF    01   FB  1585  3238          calls   #1, g^lib$do_command    ; call the do command cli service
         5E  08  C0   158C  3239          addl    #8, sp                  ; remove stacked descriptor
           EB30  30   158F  3240          bsbw    success_or_announce     ; announce any failure...
 50   00000000'EF  3C   1592  3241 100$:    movzwl  xitsts, r0              ; get the exit status
         01  50  D1   1599  3242          cmpl    r0, #1                  ; what kind of an exit is this?
             0C  13   159C  3243          beql    120$                    ; =1, a normal exit
             07  1F   159E  3244          blssu   110$                    ; =0, an EG exit
 50   0FFFFFF7'8F  D0   15A0  3245          movl    #<ss$_abort!sts$m_inhib_msg>-9, r0 ; >1, an abort exit
         50  09  C0   15A7  3246 110$:    addl    #9, r0                  ; make EG exits into a 9
                      15AA  3247 120$:    $exit_s -                       ; finally done, really exit
                      15AA  3248                  code = r0               ;  with this reason code
             50  DD   15AA                PUSHL   r0
  00000000'GF    01   FB  15AC                CALLS   #1,G^SYS$EXIT
                      15B3  3249
                      15B3  3250 .disable lsb
                      15B3  3251
                      15B3  3252 .end     tec$teco
```

Symbol table

```
$$$$$$                          = 000027B6
$$.TAB                          = 0000089C R      06
$$.TABEND                       = 000008E0 R      06
$$.TMP                          = 00000400
$$.TMP1                         = 00000001
$$.TMP2                         = 000000CF
$$T1                            = 00000001
$ALLER                          = 00000011   G
$BAKUP                          = 0000000C   G
$CLI.                           = 00000050 R      07
$CLI..                          = 0000006C R      07
$CLSFL                          = 0000000F   G
$CLSOF                          = 00000010   G
$DATE                           = 00000007   G
$DELCH                          = 00000014   G
$DELLN                          = 00000013   G
$EIGHT                          = 00000002   G
$EJFLG                          = 00000004   G
$GETBF                          = 0000000D   G
$GETFL                          = 00000009   G
$GEXIT                          = 00000005   G
$INPSV                          = 0000000A   G
$KILFL                          = 00000012   G
$OUTSV                          = 0000000B   G
$PUTBF                          = 0000000E   G
$SIZER                          = 00000006   G
$TEXIT                          = 00000016   G
$TIME                           = 00000008   G
$TRULN                          = 00000003   G
$WIDTH                          = 00000001   G
$XITNW                          = 00000015   G
..ABS..                           00000000
ABORT_EXIT                        00000465 R      0A
CHECK_ESC_CSI                     00000255 R      08
CLEAN_UP_AND_START                00000480 R      0A
CLI$A_RQADDR                    = 0000000C
CLI$B_RQSTAT                    = 00000003
CLI$B_RQTYPE                    = 00000000
CLI$C_REQDESC                   = 0000001C
CLI$GET_VALUE                     ********     X    0A
CLI$K_GETCMD                    = 00000001
CLI$K_VERB_EDIT                   ********     X    0A
CLI$PRESENT                       ********     X    0A
CLI$W_RQSIZE                    = 00000008
CLI_COMMAND_LINE                  00000074 R      07
CLI_DOLLAR                        00000100 R      04
CLI_EQUALS                        000000F7 R      04
CLI_INIT                          00000109 R      04
CLI_INSPECT                       000000D6 R      04
CLI_NO_CREATE                     000000B4 R      04
CLI_NO_INI                        000000A6 R      04
CLI_NO_MEMORY                     000000C5 R      04
CLI_NULL                          000000E6 R      04
CLI_PARM_P1                       00000153 R      04
CLI_QUAL_COMMAND                  00000119 R      04
CLI_QUAL_CREATE                   00000128 R      04
CLI_QUAL_EXECUTE                  00000144 R      04
CLI_QUAL_MEMORY                   00000136 R      04
CLI_QUAL_OUTPUT                   0000015D R      04
CLI_QUAL_READ_ONLY                0000C16B R      04
CLI_REQ_GETCMD                    00000050 R      07
CLI_RESULT                        0000006C R      07
CLI_SPACE                         000000EE R      04
CLI_VERB_TECO                     0000009A R      04
CLOSE_INDIR                       00000D66 R      08
CLOSE_INPUT                       00000D53 R      08
CMDPRM                            ********       X  0A
CMDQRG                            ********       X  08
CNV8BT                            ********       X  0A
COLON_EG_LIST                     000002B0 R        03
CRTCUP                            ********       X  08
CRTERC                            ********       X  08
CRTERL                            ********       X  08
CRTYPE                            ********       X  08
CTLOFG                            ********       X  08
CTRLC_FLAG                        000000FC R        06
CTRLO_FLAG                        000000FD R        06
CTRLZ_CNT                         000000EC R        06
CURFRE                            ********       X  08
DC$_TERM                          ********       X  0A
DEVBUFSIZ                         00000002 R        07
DEVCLASS                          00000000 R        07
DEVDEPEND                         00000004 R        07
DEVDEPEND2                        00000008 R        07
DEVNAM                            0000000C R        07
DEVTYPE                           00000001 R        07
DO_NON_NULL                       00000E64 R        08
DSC$K_CLASS_D                   = 00000002
DSC$K_DTYPE_T                   = 0000000E
DUMP_DATA                         0000099D R        08
DVI$_DEVBUFSIZ                  = 00000008
DVI$_DEVCLASS                   = 00000004
DVI$_DEVDEPEND                  = 0000000A
DVI$_DEVDEPEND2                 = 0000001C
DVI$_DEVNAM                     = 00000020
DVI$_DEVTYPE                    = 00000006
DVI$_UNIT                       = 0000000C
ECHO_BUFFER                       00000694 R        08
ECHO_BYTE                         000006FC R        08
ECHO_CHAR                         00000690 R        08
ECHO_CRLF                         00000688 R        08
ECHO_DUMP                         000007F0 R        08
ECHO_LF                           0000068D R        08
EHELP                             ********     X    08
ENABLE_CTRLCAST                   000001CC R        08
ENABLE_CTRLT                      00001225 R        08
EN_FAB                            000006EC R        06
EN_NAM                            00000740 R        06
EN_NEXT                           00001114 R        08
EN_OCCUR                          00000FE7 R        06
EN_PRESET                         000010EF R        08
EN_SPEC                           00000EE8 R        06
EOFLAG                            ********     X    08
ERR                               00000145 R        08
```

```
ERRBFL                  ********    X   08
ERRBUF                  ********    X   0A
ERR_MSGVEC              00000104 R      06
ETYPE                   ********    X   0A
EUFLAG                  ********    X   0A
EXITING_FLAG            00000100 R      06
FAB$B_FNS               =  00000034
FAB$B_FSZ               =  0000003F
FAB$B_RAT               =  0000001E
FAB$B_RFM               =  0000001F
FAB$B_SHR               =  00000017
FAB$C_BID               =  00000003
FAB$C_BLN               =  00000050
FAB$C_SEQ               =  00000000
FAB$C_STM               =  00000004
FAB$C_VAR               =  00000002
FAB$C_VFC               =  00000003
FAB$L_ALQ               =  00000010
FAB$L_FNA               =  0000002C
FAB$L_FOP               =  00000004
FAB$L_NAM               =  00000028
FAB$L_STV               =  0000000C
FAB$L_TECCTL               00000064
FAB$L_TECDSP               00000060
FAB$L_TECRAB               00000054
FAB$L_TECSTS               00000050
FAB$L_XAB               =  00000024
FAB$M_BLK               =  00000008
FAB$M_CR                =  00000002
FAB$M_FTN               =  00000001
FAB$M_GET               =  00000002
FAB$M_PUT               =  00000001
FAB$M_SQO               =  00000040
FAB$M_SUP               =  00000004
FAB$M_TECB2                00000200
FAB$M_TECBUF               00000004
FAB$M_TECECR               00000010
FAB$M_TECEOF               00000001
FAB$M_TECFMT               00000040
FAB$M_TECICR               00000008
FAB$M_TECNO1ST             00000002
FAB$M_TECNV                00000400
FAB$M_TECNXT               00000020
FAB$M_TECRW                00000080
FAB$M_TECSH                00000100
FAB$M_TECSTM               00000800
FAB$M_TECVAR               00001000
FAB$M_TEF               =  10000000
FAB$M_UPI               =  00000040
FAB$Q_TECQUE               00000058
FAB$V_CHAN_MODE         =  00000002
FAB$V_CR                =  00000001
FAB$V_DLT               =  0000000F
FAB$V_FILE_MODE         =  00000004
FAB$V_FTN               =  00000000
FAB$V_GET               =  00000001
FAB$V_LNM_MODE          =  00000000
FAB$V_MXV               =  00000001
FAB$V_PRN               =  00000002
FAB$V_PUT               =  00000000
FAB$V_RWO               =  00000007
FAB$V_SQO               =  00000006
FAB$V_TECB2                00000009
FAB$V_TECBUF               00000002
FAB$V_TECECR               00000004
FAB$V_TECEOF               00000000
FAB$V_TECFMT               00000006
FAB$V_TECICR               00000003
FAB$V_TECNO1ST             00000001
FAB$V_TECNV                0000000A
FAB$V_TECNXT               00000005
FAB$V_TECRW                00000007
FAB$V_TECSH                00000008
FAB$V_TECSTM               0000000B
FAB$V_TECVAR               0000000C
FAB$V_TRN               =  00000004
FAB$W_GBC               =  00000048
FAB$W_MRS               =  00000036
FETCH_FILBUF               0000107A R      08
FFFLAG                     ********    X   08
FILBUF                     ********    X   0A
FILE_SPEC_BUF              000008ED R      06
FILE_SPEC_LEN              000008EC R      06
FILE_SPEC_OPT              000008E0 R      06
FILE_SPEC_SWT              000008E4 R      06
FILE_SPEC_TABLE            00000100 R      03
FILSIZ                     ********    X   08
FILSRT                     ********    X   0A
FREE_DATA                  00000D9F R      08
GETBYT                     00000C61 R      08
GETBYT_EMIT_CR             00000B5F R      08
GETBYT_EMIT_CTL            00000B4E R      08
GETBYT_EMIT_LF             00000B64 R      08
GETBYT_FIRST               00000B91 R      08
GETDVI_INFO                00000000 R      07
GETDVI_INFO_LEN         =  0000000C
GETDVI_ITMLST              00000000 R      04
GETDVI_TER_C               0000021C R      0A
GETDVI_TER_I               000000C4 R      0A
GETDVI_TER_O               00000158 R      0A
GET_FILE_CHAR              000010C2 R      08
HTSIZE                     ********    X   08
INDIR                      ********    X   0A
INDIR_CMD_BUF              00000E00 R      05
INDIR_CMD_DNA              0000031F R      03
INDIR_CMD_DNS              00000004
INDIR_CMD_FAB              00000378 R      06
INDIR_CMD_NAM              000003E0 R      06
INDIR_CMD_RAB              00000440 R      06
INDIR_CMD_SIZ              00000800
INDIR_CMD_SPEC             00000BEB R      06
INDIR_CMD_VFC              000010F2 R      06
INITIAL_SIZ                00001388
INI_DCD_FNS             =  0000000E
```

```
INI_DCD_LOGNAM                  0000008E R      04
INPALT                          ********   X    08
INPNOR                          ********   X    0A
INPNTR                          ********   X    0A
INPUT_ALT_BUF                   00001600 R      05
INPUT_ALT_FAB                   00000484 R      06
INPUT_ALT_NAM                   000004EC R      06
INPUT_ALT_RAB                   000005A4 R      06
INPUT_ALT_SIZ                   00000800
INPUT_ALT_SPEC                  00000CEA R      06
INPUT_ALT_VFC                   000010FE R      06
INPUT_ALT_XAB                   0000054C R      06
INPUT_NOR_BUF                   00000600 R      05
INPUT_NOR_FAB                   00000110 R      06
INPUT_NOR_NAM                   00000178 R      06
INPUT_NOR_RAB                   00000230 R      06
INPUT_NOR_SIZ                   00000800
INPUT_NOR_SPEC                  000009ED R      06
INPUT_NOR_VFC                   000010E6 R      06
INPUT_NOR_XAB                   000001D8 R      06
INPUT_SYS_FAB                   000007A0 R      06
INPUT_SYS_RAB                   00000808 R      06
INPUT_SYS_VFC                   0000110A R      06
INPUT_VFC_SIZ                   0000000C
IO$BIN                          ********   X    0A
IO$CCO                          ********   X    0A
IO$M_CANCTRLO                   ********   X    0A
IO$M_CTRLCAST                   ********   X    08
IO$M_DSABLMBX                   ********   X    08
IO$M_NOFILTR                    ********   X    08
IO$M_NOFORMAT                   ********   X    0A
IO$M_TRMNOECHO                  ********   X    08
IO$V_CANCTRLO                   ********   X    08
IO$V_CVTLOW                     ********   X    08
IO$V_NOECHO                     ********   X    08
IO$V_TIMED                      ********   X    08
IO$_READVBLK                    ********   X    08
IO$_SETMODE                     ********   X    08
IO$_WRITEVBLK                   ********   X    08
IOERR                           ********   X    08
IOERRS                          ********   X    08
I_BIAS                          00000020
I_CODE                          0000001C
I_PC                            00000020
I_PS                            00000024
I_R0                            00000000
I_R1                            00000004
I_R2                            00000008
I_R3                            0000000C
I_R4                            00000010
I_R5                            00000014
I_SP                            00000018
JPI$_PID                      = 00000319
JPI$_UIC                      = 00000304
LIB$DELETE_LOGICAL              ********   X    08
LIB$DISABLE_CTRL                ********   X    08
LIB$DO_COMMAND                  ********   X    08
LIB$ENABLE_CTRL                 ********   X    08
LIB$FREE_VM                     ********   X    08
LIB$GET_SYMBOL                  ********   X    08
LIB$GET_VM                      ********   X    08
LIB$K_CLI_GLOBAL_SYM          = 00000002
LIB$K_CLI_LOCAL_SYM           = 00000001
LIB$M_CLI_CTRLT               = 00100000
LIB$M_CLI_CTRLY               = 02000000
LIB$SET_LOGICAL                 ********   X    08
LIB$SET_SYMBOL                  ********   X    08
LIB$_NOSUCHSYM                  ********   X    08
N                               ********   X    08
NAM$B_ESS                     = 0000000A
NAM$B_NOP                     = 00000008
NAM$B_RSL                     = 00000003
NAM$B_RSS                     = 00000002
NAM$C_BID                     = 00000002
NAM$C_BLN                     = 00000060
NAM$C_MAXRSS                  = 000000FF
NAM$L_ESA                     = 0000000C
NAM$L_RSA                     = 00000004
NFLG                            ********   X    08
NON_NULL                        00000DF2 R      08
NOT_EXITING                     0000124D R      08
OUPALT                          ********     X  08
OUPNOR                          ********     X  0A
OUPNTR                          ********     X  0A
OUTDNE                          ********     X  0A
OUTPUT_ALT_FAB                  000005E8 R      06
OUTPUT_ALT_NAM                  00000648 R      06
OUTPUT_ALT_RAB                  000006A8 R      06
OUTPUT_ALT_SPEC                 00000DE9 R      06
OUTPUT_NOR_FAB                  00000274 R      06
OUTPUT_NOR_NAM                  000002D4 R      06
OUTPUT_NOR_RAB                  00000334 R      06
OUTPUT_NOR_SPEC                 00000AEC R      06
OUTPUT_SYS_BUF                  00001E00 R      05
OUTPUT_SYS_FAB                  0000084C R      06
OUTPUT_SYS_RAB                  0000089C R      06
OUTPUT_SYS_SIZ                  00000200
OUTPUT_SYS_VFC                  000000F2 R      06
PDL                             ********     X  0A
PDLSRT                          ********     X  0A
PRE_LOAD_Q_REGS                 0000025B R      0A
PRTCIN                          ********     X  08
PSL$C_USER                    = 00000003
PSL$M_CM                      = 80000000
PSL$V_CURMOD                  = 00000018
PSL$V_PRVMOD                  = 00000016
PUT_BUFFER                      00000A52 R      08
QARRAY                          ********     X  0A
QCMND                           ********     X  08
QMAX                            ********     X  0A
QRSTOR                          ********     X  0A
QUOTA_MSG_DESC                  00000323 R      03
QZ                              ********     X  0A
R5SET                           ********     X  0A
```

Symbol table

```
RAB$B_RAC                       =  0000001E
RAB$C_BID                       =  00000001
RAB$C_BLN                       =  00000044
RAB$C_SEQ                       =  00000000
RAB$L_CTX                       =  00000018
RAB$L_RBF                       =  00000028
RAB$L_RHB                       =  0000002C
RAB$L_ROP                       =  00000004
RAB$L_STV                       =  0C00000C
RAB$V_LOC                       =  00000010
RAB$V_RAH                       =  00000009
RAB$V_TPT                       =  00000001
RAB$V_WBH                       =  0000000A
RAB$W_RSZ                       =  00000022
RESET_INDIR                       00000D69 R      08
RMS$_EOF                          ********   X    08
RMS$_FNF                          ********   X    08
RMS$_FSZ                          ********   X    0A
RMS$_IFI                          ********   X    08
RMS$_NMF                          ********   X    08
RMS$_NORMAL                       ********   X    08
RMS$_PRV                          ********   X    08
RMS$_SUPERSEDE                    ********   X    08
RMS$_SYN                          ********   X    08
RWSIZE                            ********   X    0A
SAVED_SP                          000000E4 R      06
SAVE_DATA                         000009D7 R      08
SCHBUF                            ********   X    0A
SCHSRT                            ********   X    0A
SET_FILENAME                      00000CDC R      08
SET_INPUTNAME                     00000CBD R      08
SET_OUTPUTNAME                    00000CB1 R      08
SIZING_MSG_DESC                   00000342 R      03
SPSET                             ********   X    0A
SS$_ABORT                         ********   X    0A
SS$_CONTROLC                      ********   X    08
SS$_EXDISKQUOTA                   ********   X    08
SS$_NOLOGNAM                      ********   X    08
SS$_NOTRAN                        ********   X    0A
SS$_SUPERSEDE                     ********   X    08
SS$_TIMEOUT                       ********   X    08
START_TECO                        00000077 R      08
STILL_FREE                        000000E8 R      06
STR$APPEND                        ********   X    0A
STR$CONCAT                        ********   X    0A
STS$K_SEVERE                    =  00000004
STS$M_INHIB_MSG                 =  10000000
STS$M_SEVERITY                  =  00000007
SUCCESS_ELSE_DIE                  000000A7 R      08
SUCCESS_OR_ABRT                   000000D7 R      08
SUCCESS_OR_ANNOUNCE               000000C2 R      08
SUCCESS_OR_CLS                    000000E4 R      08
SUCCESS_OR_ERR                    000000FE R      08
SWITCH_LIST                       00000250 R      03
SYMSPC                            ********   X    0A
SYS$ASSIGN                        ********  GX    0A
SYS$CANCEL                        ********  GX    08
SYS$CLI                           ********   X    0A
SYS$CLOSE                         ********  GX    0A
SYS$CLREF                         ********  GX    08
SYS$CONNECT                       ********  GX    0A
SYS$CREATE                        ********  GX    0A
SYS$CRELOG                        ********  GX    0A
SYS$DASSGN                        ********  GX    08
SYS$DCLCMH                        ********  GX    0A
SYS$EXIT                          ********  GX    08
SYS$GET                           ********  GX    08
SYS$GETDVI                        ********  GX    0A
SYS$GETJPI                        ********  GX    08
SYS$NUMTIM                        ********  GX    08
SYS$OPEN                          ********  GX    0A
SYS$PARSE                         ********  GX    08
SYS$PURGWS                        ********  GX    08
SYS$PUT                           ********  GX    08
SYS$PUTMSG                        ********  GX    08
SYS$QIO                           ********  GX    08
SYS$QIOW                          ********  GX    08
SYS$REWIND                        ********  GX    08
SYS$SEARCH                        ********  GX    08
SYS$TRNLOG                        ********  GX    0A
SYS$WAITFR                        ********  GX    08
T$ANSI                            00000100  G
T$DEC                             00002000  G
T$EDIT                            00001000  G
T$VT05                            00000001  G
TAGBUF                            ********   X    0A
TAGSRT                            ********   X    0A
TEC$ALLER                         00000DB4 R      08
TEC$BAKUP                         0000087E R      08
TEC$CLSFL                         00000D13 R      08
TEC$CLSOF                         00000D15 R      08
TEC$CMTRAP                        00000000 R      08
TEC$CTRLCAST                      00000194 R      08
TEC$DATE                          00001453 R      08
TEC$DELCH                         0000079F R      08
TEC$DELLN                         00000769 R      08
TEC$EIGHT                         000011B9 R      08
TEC$EJFLG                         00001158 R      08
TEC$GETBF                         00000912 R      08
TEC$GETFL                         00000DC7 R      08
TEC$GEXIT                         000012F1 R      08
TEC$INPSV                         00000CB7 R      08
TEC$INPUT                         00000576 R      08
TEC$KILFL                         00000D5C R      08
TEC$M_ED$CTL                      00000001
TEC$M_ED$EXP                      00000004
TEC$M_ED$IMD                      00000020
TEC$M_ED$INC                      00000040
TEC$M_ED$SRH                      00000010
TEC$M_ED$WCH                      00000080
TEC$M_ED$YNK                      00000002
TEC$M_ET$8BT                      00001000
TEC$M_ET$BIN                      00000001
TEC$M_ET$CC                       00008000
```

Symbol table

```
TEC$M_ET$CCO                    00000010
TEC$M_ET$CKE                    00000020
TEC$M_ET$CRT                    00000002
TEC$M_ET$DET                    00000040
TEC$M_ET$GRV                    00002000
TEC$M_ET$IAS                    00000200
TEC$M_ET$LC                     00000004
TEC$M_ET$NCH                    00000008
TEC$M_ET$RFS                    00000400
TEC$M_ET$TRU                    00000100
TEC$M_ET$XIT                    00000080
TEC$OUTPUT                      00000371 R      08
TEC$OUTPUT_AST                  000002BE R      08
TEC$OUTPUT_MORE                 00000375 R      08
TEC$OUTSV                       00000CAB R      08
TEC$OUT_ASCID                   00000660 RG     08
TEC$PUTBF                       00000A47 R      08
TEC$SETMODE                     000011F3 R      08
TEC$SIZER                       000013BC R      08
TEC$TECO                        00000000 RG     0A
TEC$TEXIT                       000014B0 R      08
TEC$TIME                        0000146B R      08
TEC$TRULN                       000011AE R      08
TEC$V_ED$CTL                    00000000
TEC$V_ED$EXP                    00000002
TEC$V_ED$IMD                    00000005
TEC$V_ED$INC                    00000006
TEC$V_ED$SRH                    00000004
TEC$V_ED$WCH                    00000007
TEC$V_ED$YNK                    00000001
TEC$V_ET$8BT                    0000000C
TEC$V_ET$BIN                    00000000
TEC$V_ET$CC                     0000000F
TEC$V_ET$CCO                    00000004
TEC$V_ET$CKE                    00000005
TEC$V_ET$CRT                    00000001
TEC$V_ET$DET                    00000006
TEC$V_ET$GRV                    0000000D
TEC$V_ET$IAS                    00000009
TEC$V_ET$LC                     00000002
TEC$V_ET$NCH                    00000003
TEC$V_ET$RFS                    0000000A
TEC$V_ET$TRU                    00000008
TEC$V_ET$XIT                    00000007
TEC$WAIT                        0000023E R      08
TEC$WAIT_DONE                   00000212 R      08
TEC$WAIT_DONE_LF                000001ED R      08
TEC$WIDTH                       000011E6 R      08
TEC$XITNW                       00001220 R      08
TECO                            ********   X    08
TECOBUF                         00000000 R      05
TECOCH                          ********   X    0A
TECOCMD                         ********   X    0A
TECOCMD_SIZ                     ********   X    0A
TECOCR                          ********   X    08
TECOCTL                         00000000 R      06
TECOCTLINI                      00000000 R      07
TECODAT                         00000000 R      03
TECODATINI                      00000000 R      04
TECOEXE                         00000000 R      08
TECOEXEINI                      00000000 R      0A
TECOEXELBR                      00000000 R      09
TECOJP                          ********   X    0A
TECOPD                          ********   X    0A
TECOSP                          ********   X    0A
TECOST                          ********   X    0A
TER_C_CHAN                      000000F8 R      06
TER_C_DEVNAM                    0000007B R      04
TER_I                           00000020 R      06
TER_I_ANY_TRM                   00000228 R      03
TER_I_BUF                       00000400 R      05
TER_I_BUF_PRE                   00000003
TER_I_CHAN                      000000F4 R      06
TER_I_DEVNAM                    00000061 R      04
TER_I_DEVNAM_FNA                00000058 R      04
TER_I_DEVNAM_FNS              = 00000009
TER_I_NOR7_TRM                  00000200 R      03
TER_I_NOR8_TRM                  00000028 R      06
TER_I_NOR_TRM_PTR               000000E0 R      06
TER_I_SIZ                       00000200
TER_I_STATUS                    00000018 R      06
TER_OOB_MSK                     000008E8 R      06
TER_O_BUF1                      00000000 R      05
TER_O_BUF2                      00000200 R      05
TER_O_CC                        000000F0 R      06
TER_O_CHAN                      000000F6 R      06
TER_O_DEVNAM                    00000073 R      04
TER_O_DEVNAM_FNA                00000069 R      04
TER_O_DEVNAM_FNS              = 0000000A
TER_O_FORCE                     000000FE R      06
TER_O_PEND                      000000FF R      06
TER_O_POS                       00000010 R      06
TER_O_SIZ                       00000200
TER_O_STATUS1                   00000000 R      06
TER_O_STATUS2                   00000008 R      06
TER_O_TABLE                     00000000 R      03
TER_O_UNIT                      000000FA R      06
TEXIT                           ********   X    08
TFLG                            ********   X    08
TMP_STRING                      00000050 R      06
TMP_STRING2                     00000098 R      06
TMP_STRING2_BUF               = 000000A0 R      06
TMP_STRING2_SIZ               = 0000003F
TMP_STRING_BUF                = 00000058 R      06
TMP_STRING_SIZ                = 0000003F
TT$M_ESCAPE                   = 00000008
TT$M_HALFDUP                  = 00100000
TT$V_EIGHTBIT                 = 0000000F
TT$V_LOWER                    = 00000007
TT$V_SCOPE                    = 0000000C
TT$V_WRAP                     = 00000009
TT$_VT05                      = 00000001
TT2$M_ANSICRT                 = 01000000
TT2$M_DECCRT                  = 20000000
```

```
TT2$M_EDIT                      = 10000000
TTOBFC                            ********    X   08
TTOBUF                            ********    X   08
TTOINT                            ********    X   08
TTOMOD                            ********    X   08
TTOPTR                            ********    X   08
TXSTOR                            ********    X   0A
UNIT                              0000004C  R     07
XAB$B_MTACC                     = 0000000A
XAB$B_PROT_MODE                 = 00000010
XAB$B_PROT_OPT                  = 0000000B
XAB$C_PRO                       = 00000013
XAB$C_PROLEN                    = 00000058
XAB$L_ACLBUF                    = 00000018
XAB$L_ACLCTX                    = 00000020
XAB$L_NXT                       = 00000004
XAB$M_NODEL                     = 00000008
XAB$M_NOEXE                     = 00000004
XAB$M_NOREAD                    = 00000001
XAB$M_NOWRITE                   = 00000002
XAB$V_OWN                       = 00000004
XAB$W_ACLSIZ                    = 0000001C
XAB$W_GRP                       = 0000000E
XAB$W_MBM                       = 0000000C
XAB$W_PRO                       = 00000008
XITSTS                            ********    X   08
ZMAX                              ********    X   0A
ZZ                                ********    X   08
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
..ABS..	00010000 (65536.)	02 (2.)	NOPIC USR OVR ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
TECODAT	000003E0 (992.)	03 (3.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC PAGE
TECODATINI	0000017C (380.)	04 (4.)	NOPIC USR CON REL LCL SHR NOEXE RD NOWRT NOVEC PAGE
TECOBUF	00002000 (8192.)	05 (5.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC PAGE
TECOCTL	00001116 (4374.)	06 (6.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC PAGE
TECOCTLINI	0000007C (124.)	07 (7.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC PAGE
TECOEXE	000015B3 (5555.)	08 (8.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC PAGE
TECOEXELBR	00000003 (3.)	09 (9.)	NOPIC USR OVR REL GBL SHR EXE RD NOWRT NOVEC PAGE
TECOEXEINI	000004CE (1230.)	0A (10.)	NOPIC USR CON REL LCL SHR EXE RD NOWRT NOVEC PAGE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.10	00:00:00.92
Command processing	123	00:00:00.55	00:00:03.41
Pass 1	1309	00:00:48.14	00:01:33.77
Symbol table sort	0	00:00:03.25	00:00:06.65

Pass 2	1607	00:00:12.77	00:00:28.04
Symbol table output	3	00:00:00.44	00:00:00.93
Psect synopsis output	3	00:00:00.05	00:00:00.06
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	3077	00:01:05.32	00:02:13.78

The working set limit was 2000 pages.
415521 bytes (812 pages) of virtual memory were used to buffer the intermediate code.
There were 110 pages of symbol table space allocated to hold 1850 non-local and 353 local symbols.
3252 source lines were read in Pass 1, producing 72 object records in Pass 2.
88 pages of virtual memory were used to define 79 macros.

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

Macro library name	Macros defined
_255DUA28:[SYSLIB]STARLET.MLB;2	59

2009 GETS were required to define 59 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:TECONAT/OBJ=OBJ$:TECONAT MSRC$:TECONAT/UPDATE=(ENH$:TECONAT)
```

UTILKEY.LIS

TECOLBR.LIS

TECOCMD.LIS

SCSLOA.LIS

TECO

TECO.MAP

TECONAT.LIS

TRACEMSG
MDL

TBKLIB
REQ

TRACE

TRACE
MAP

STRUCDEF
REQ

TBKPROLOG
REQ

TBKDST
REQ

STRUCDEF
LIS