

[illegible]

TECC
TECC
TECC
TECC
TECC
TECC
TECC
TECC
TECC
TECC
TEMP
TEXT
TFLC
TIME
TLIS
TOPL
TTOE
TTOE
TTOI
TTON
TTOF
TXSI
TYPE
TYPE
TYPE
TYPE
TYPE
UPPE
USRF
USRS
VERS
VTSI
VTSI
WATC
XITN
XITS
ZE-0
ZMAX
ZZ

```

TTTTTTTTT1  EEEEEEEEEEE  CCCCCCCCC  000000  NN  NN  AAAAAA  TTTTTTTTTT
TTTTTTTTTTT  EEEEEEEEEEE  CCCCCCCCC  000000  NN  NN  AAAAAA  TTTTTTTTTT
      TT      EE          CC      00      00  NN  NN  AA      AA  TT
      TT      EE          CC      00      00  NN  NN  AA      AA  TT
      TT      EE          CC      00      00  NNNN  NN  AA      AA  TT
      TT      EE          CC      00      00  NNNN  NN  AA      AA  TT
      TT      EEEEEEEEE  CC      00      00  NN  NN  NN  AA      AA  TT
      TT      EEEEEEEEE  CC      00      00  NN  NN  NN  AA      AA  TT
      TT      EE          CC      00      00  NN  NN  NN  AA      AA  TT
      TT      EE          CC      00      00  NN  NN  NN  AA      AA  TT
      TT      EE          CC      00      00  NN  NN  NN  AA      AA  TT
      TT      EE          CC      00      00  NN  NN  NN  AA      AA  TT
      TT      EEEEEEEEE  CC      00      00  NN  NN  NN  AA      AA  TT
      TT      EEEEEEEEE  CC      00      00  NN  NN  NN  AA      AA  TT
      TT      EEEEEEEEE  CCCCCCCCC  000000  NN  NN  AA      AA  TT
      TT      EEEEEEEEE  CCCCCCCCC  000000  NN  NN  AA      AA  TT

```

```

LL          IIIII
LL          IIIII
LL          III
LL          III
LL          III
LL          III
LL          III
LL          III
LL          III
LL          III
LL          III
LL          III
LL          III
LL          III
LLLLLLLLLL IIIII
LLLLLLLLLL IIIII
SSSSSSSSS
SSSSSSSSS
SS
SS
SS
SS
SSSSSS
SSSSSS
SS
SS
SS
SS
SSSSSSSSS
SSSSSSSSS

```

(1)	28	VAX-11 TECO
(2)	32	General macros
(3)	162	System definitions
(4)	219	Internal macros
(5)	241	'ET' (edit typeout) bits
(5)	262	'ED' (edit mode) bits
(6)	283	Internal definitions
(7)	337	.PSECT definitions
(8)	358	Pure data
(9)	554	Impure data
(10)	970	Permanent I/O buffers
(11)	999	Main startup entry point
(11)	1005	Initialization code
(17)	1317	Compatibility mode trap handler
(18)	1381	Initial start up
(19)	1393	Error processing, etc.
(20)	1482	Control/c ASTs
(21)	1505	Terminal output waits
(22)	1542	Terminal output
(23)	1694	Terminal input
(24)	1863	Echoing, etc.
(25)	1947	Process line/character deletion echoing
(26)	2033	Page backwards
(27)	2078	Get input
(28)	2150	Put output
(29)	2313	Get an input byte
(30)	2446	Switch to alternate output
(30)	2456	Switch to alternate input
(31)	2490	Close input & output files
(31)	2497	Close output file
(31)	2523	Kill output file
(31)	2531	Close indirect command file
(31)	2563	Error message finish up
(32)	2580	Get files opened, etc.
(33)	2826	Do 'en' processing
(34)	2866	Handle the EJ flag
(35)	2893	Handle line truncation mode changes
(35)	2903	Handle 8-bit terminal mode changes
(35)	2919	Handle new terminal width
(36)	2939	Stop terminal hacks
(37)	2967	Process special functions
(38)	3088	Get additional memory
(39)	3138	Get date and time
(40)	3168	Exit from TECO

```
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 .macro tmporg sect, <offset>
0000 44 .endm org
0000 45
0000 46 .macro deforg sect
0000 47 .macro 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 <1a<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'$v' suffix'$v' sym: .blkb
0000 85 .bsect prefix'$v' suffix'$v' sym
0000 86 prefix'$m' suffix'$v' sym: .blkb
0000 87 .dsect prefix'$v' suffix'$v' 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 <12*%extract(1,3000,bit)>-63
0000 136 bicb s^#12*%extract(1,3000,bit), dst
0000 137 .mexit
0000 138 .endc ; le <12*%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      <1a%extract(1,3000,bit)>-63
0000 147         bisb    s^#1a%extract(1,3000,bit), dst
0000 148 .mexit
0000 149 .endc    ; le      <1a%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
```

```
0000 162 .sbtll System definitions
0000 163
0000 164 .noshow meb
0000 165
0000 166 $clidef ; cli defs
0000 167 $dscdef ; string descriptor defs
0000 168 $dvidef ; get device information defs
0000 169 $jpiddef ; get job/process information defs
0000 170 $libclidef ; cli defs
0000 171 $psldef ; processor status longword defs
0000 172 $stsdef ; status code definitions
0000 173 $ttdef ; terminal modes/characteristics defs
0000 174 $tt2def ; terminal modes/characteristics defs
0000 175
0000 176 .show meb
0000 177
0000 178 ;+
0000 179 ; Entry to compatibility mode exception handler with:
0000 180 ;
0000 181 ; -32(r0) = saved r0 (ctl$al_cmctx)
0000 182 ; -28(r0) = saved r1
0000 183 ; -24(r0) = saved r2
0000 184 ; -20(r0) = saved r3
0000 185 ; -16(r0) = saved r4
0000 186 ; -12(r0) = saved r5
0000 187 ; -08(r0) = saved r6 (compatibility sp)
0000 188 ; -04(r0) = exception code
0000 189 ; 00(r0) = saved exception pc
0000 190 ; 04(r0) = saved exception psl (compatibility ps)
0000 191 ;
0000 192 ; Exception codes are:
0000 193 ;
0000 194 ; 0 = reserved instruction
0000 195 ; 1 = bpt
0000 196 ; 2 = iot
0000 197 ; 3 = emt
0000 198 ; 4 = trap
0000 199 ; 5 = illegal instruction
0000 200 ; 6 = odd address
0000 201 ; 7 = t-bit
0000 202 ; -
0000 203
0000 204 .dsect ; ctl$al_cmctx data storage offsets
0000 205
00000004 0000 206 i_r0: .blkl ; saved r0
00000008 0004 207 i_r1: .blkl ; saved r1
0000000C 0008 208 i_r2: .blkl ; saved r2
00000010 000C 209 i_r3: .blkl ; saved r3
00000014 0010 210 i_r4: .blkl ; saved r4
00000018 0014 211 i_r5: .blkl ; saved r5
0000001C 0018 212 i_sp: .blkl ; saved r6 (compatibility sp)
00000020 001C 213 i_code: .blkl ; saved exception code
00000024 0020 214 i_pc: .blkl ; saved pc
00000028 0024 215 i_ps: .blkl ; saved psl (compatibility ps)
00000028 0028 216
00000028 0020 217 .equate i_bias, i_pc-i_r0 ; bias from ctl$al_cmctx to entry r0
00000028 i_bias:
```

```
0020 219 .sbtll 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 .ascii 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 .mndef bin ; +1., output in binary (image) mode
00000001 0000 tec$V_et$bin: .blkb
00000002 0001 tec$m_et$bin: .blkb
0001 246 .mndef 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 .mndef lc ; +4., accept lower case input
00000003 0002 tec$V_et$lc: .blkb
00000008 0004 tec$m_et$lc: .blkb
0003 248 .mndef 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 .mndef cco ; +16., cancel control/o on output
00000005 0004 tec$V_et$cco: .blkb
00000020 0010 tec$m_et$cco: .blkb
0005 250 .mndef 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 .mndef det ; +64., detach and detached flag
00000007 0006 tec$V_et$det: .blkb
00000080 0040 tec$m_et$det: .blkb
0007 252 .mndef xit ; +128., "no prompt yet" flag
00000008 0007 tec$V_et$xit: .blkb
00000100 0080 tec$m_et$xit: .blkb
0008 253 .mndef tru ; +256., truncate long output lines
00000009 0008 tec$V_et$tru: .blkb
00000200 0100 tec$m_et$tru: .blkb
0009 254 .mndef ias ; +512., interactive scope available for "watch"
0000000A 0009 tec$V_et$ias: .blkb
00000400 0200 tec$m_et$ias: .blkb
000A 255 .mndef rfs ; +1024., refresh scope available for "watch"
0000000B 000A tec$V_et$rfs: .blkb
00000800 0400 tec$m_et$rfs: .blkb
000B 256 .mndef ; +2048., reserved by TECO-8
0000000C 000B .blkb
000C 257 .mndef 8bt ; +4096., terminal is an 8-bit terminal
0000000D 000C tec$V_et$8bt: .blkb
00002000 1000 tec$m_et$8bt: .blkb
000D 258 .mndef grv ; +8192., accept "" as escape during command input
0000000E 000D tec$V_et$grv: .blkb
00004000 2000 tec$m_et$grv: .blkb
000E 259 .mndef ; +16384., unused
0000000F 000E .blkb
000F 260 .mndef cc ; -32768., allow program to trap control/c
00000010 000F tec$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 .mndef ctl ; +1., don't allow "" as meaning control character
00000001 0000 tec$V_ed$ctl: .blkb
```

00000002	0001	tec\$med\$ctl:	.blkb	:	
	0001	.mvdef ynk		:	+2., allow yanks, etc. to clobber text buffer
00000002	0001	tec\$ved\$ynk:	.blkb	:	
00000004	0002	tec\$med\$ynk:	.blkb	:	
	0002	.mvdef exp		:	+4., don't allow arbitrary expansion(s)
00000003	0002	tec\$ved\$exp:	.blkb	:	
00000008	0004	tec\$med\$exp:	.blkb	:	
	0003	.mvdef		:	+8., reserved by TECO-8
00000004	0003		.blkb	:	
	0004	.mvdef srh		:	+16., don't reset 'dot' on search failure
00000005	0004	tec\$ved\$srh:	.blkb	:	
00000020	0010	tec\$med\$srh:	.blkb	:	
	0005	.mvdef imd		:	+32., allow immediate mode commands
00000006	0005	tec\$ved\$imd:	.blkb	:	
00000040	0020	tec\$med\$imd:	.blkb	:	
	0006	.mvdef inc		:	+64., only move 'dot' by one on iterative search failure
00000007	0006	tec\$ved\$inc:	.blkb	:	
00000080	0040	tec\$med\$inc:	.blkb	:	
	0007	.mvdef wch		:	+128., don't do automatic 'w' command before prompt
00000008	0007	tec\$ved\$wch:	.blkb	:	
00000100	0080	tec\$med\$wch:	.blkb	:	
	0008	.mvdef		:	+256., unused
00000009	0008		.blkb	:	
	0009	.mvdef		:	+512., unused
0000000A	0009		.blkb	:	
	000A	.mvdef		:	+1024., unused
0000000B	000A		.blkb	:	
	000B	.mvdef		:	+2048., unused
0000000C	000B		.blkb	:	
	000C	.mvdef		:	+4096., unused
0000000D	000C		.blkb	:	
	000D	.mvdef		:	+8192., unused
0000000E	000D		.blkb	:	
	000E	.mvdef		:	+16384., unused
0000000F	000E		.blkb	:	
	000F	.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$m_edita-16 ; define 'edit' functions...
1000 289 t$edit:: ; ... for compatibility mode
1000 290 .assume t$edit ne 0
1000 291
1000 292 .dsect tt2$m_deccrta-16 ; define 'dec crt'...
2000 293 t$dec:: ; ... for compatibility mode
2000 294 .assume t$dec ne 0
2000 295
2000 296 .dsect tt2$m_ansicrta-16 ; define 'ansi crt'...
0100 297 t$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$m_teceof: .blkb
0001 319 .mvdef nolst ; not first time through
00000002 0001 fab$v_tecnolst: .blkb
00000004 0002 fab$m_tecnolst: .blkb
0002 320 .mvdef buf ; use buffered data instead of file
00000003 0002 fab$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	C_10		fab\$m_tecocr:	.blkb	.	
	0005	323	.mvdef	nxt		; pre-fetched character exists (@ +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

```
0010 337 .sbttl .PSECT definitions
0010 338
00000000 339 .psect tecodat, page,nopic,usr,con,rel,lcl, shr,noexe,rd,nowrt
0000 340 deforg tecodat
00000000 341 .psect tecodatini, page,nopic,usr,con,rel,lcl, shr,noexe,rd,nowrt
0000 342 deforg tecodatini
0000 343
00000000 344 .psect tecobuf, page,nopic,usr,con,rel,lcl,noshr,noexe,rd, wrt
0000 345 deforg tecobuf
00000000 346 .psect tecocctl, page,nopic,usr,con,rel,lcl,roshr,noexe,rd, wrt
0000 347 deforg tecocctl
00000000 348 .psect tecocctlini, page,nopic,usr,con,rel,lcl,noshr,noexe,rd, wrt
0000 349 deforg tecocctlini
0000 350
00000000 351 .psect tecoexe, page,nopic,usr,con,rel,lcl, shr, exe,rd,nowrt
0000 352 deforg tecoexe
00000000 353 .psect tecoexelbr, page,nopic,usr,ovr,rel,gbt, shr, exe,rd,nowrt
0000 354 deforg tecoexelbr
00000000 355 .psect tecoexeini, page,nopic,usr,con,rel,lcl, shr, exe,rd,nowrt
0000 356 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

9D	009D	.byte	.-ter_o_table
9E	009E	.byte	.-ter_o_table
9F	009F	.byte	.-ter_o_table
A0	00A0	.byte	.-ter_o_table
A1	00A1	.byte	.-ter_o_table
A2	00A2	.byte	.-ter_o_table
A3	00A3	.byte	.-ter_o_table
A4	00A4	.byte	.-ter_o_table
A5	00A5	.byte	.-ter_o_table
A6	00A6	.byte	.-ter_o_table
A7	00A7	.byte	.-ter_o_table
A8	00A8	.byte	.-ter_o_table
A9	00A9	.byte	.-ter_o_table
AA	00AA	.byte	.-ter_o_table
AB	00AB	.byte	.-ter_o_table
AC	00AC	.byte	.-ter_o_table
AD	00AD	.byte	.-ter_o_table
AE	00AE	.byte	.-ter_o_table
AF	00AF	.byte	.-ter_o_table
B0	00B0	.byte	.-ter_o_table
B1	00B1	.byte	.-ter_o_table
B2	00B2	.byte	.-ter_o_table
B3	00B3	.byte	.-ter_o_table
B4	00B4	.byte	.-ter_o_table
B5	00B5	.byte	.-ter_o_table
B6	00B6	.byte	.-ter_o_table
B7	00B7	.byte	.-ter_o_table
B8	00B8	.byte	.-ter_o_table
B9	00B9	.byte	.-ter_o_table
BA	00BA	.byte	.-ter_o_table
BB	00BB	.byte	.-ter_o_table
BC	00BC	.byte	.-ter_o_table
BD	00BD	.byte	.-ter_o_table
BE	00BE	.byte	.-ter_o_table
BF	00BF	.byte	.-ter_o_table
C0	00C0	.byte	.-ter_o_table
C1	00C1	.byte	.-ter_o_table
C2	00C2	.byte	.-ter_o_table
C3	00C3	.byte	.-ter_o_table
C4	00C4	.byte	.-ter_o_table
C5	00C5	.byte	.-ter_o_table
C6	00C6	.byte	.-ter_o_table
C7	00C7	.byte	.-ter_o_table
C8	00C8	.byte	.-ter_o_table
C9	00C9	.byte	.-ter_o_table
CA	00CA	.byte	.-ter_o_table
CB	00CB	.byte	.-ter_o_table
CC	00CC	.byte	.-ter_o_table
CD	00CD	.byte	.-ter_o_table
CE	00CE	.byte	.-ter_o_table
CF	00CF	.byte	.-ter_o_table
D0	00D0	.byte	.-ter_o_table
D1	00D1	.byte	.-ter_o_table
D2	00D2	.byte	.-ter_o_table
D3	00D3	.byte	.-ter_o_table
D4	00D4	.byte	.-ter_o_table
D5	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
00000020' 0200 389 ter_i_nor7_trm: ; normal 7-bit terminator mask
00000208' 0200 390 .long 20$-10$
0204 391 .long 10$
0208 392
FFFFE0FF 0208 393 10$: .long ^c< <1a8>! - ; all of 0- 31 except bs,
020C 394 <1a9>! - ; tab,
020C 395 <1a10>! - ; lf,
020C 396 <1a11>! - ; vt,
020C 397 <1a12>> ; ff
00000000 020C 398 .long 0 ; none of 32- 63
00000000 0210 399 .long 0 ; none of 64- 95
80000001 0214 400 .long < <1a<96-96>>! - ; none of 96-127 except accent grave,
0218 401 <1a<127-96>>> ; del
FFFFFFFFFF 0218 402 .long ^c<0> ; all of 128-159
FFFFFFFFFF 021C 403 .long ^c<0> ; all of 160-191
FFFFFFFFFF 0220 404 .long ^c<0> ; all of 192-223
FFFFFFFFFF 0224 405 .long ^c<0> ; all of 224-255
0228 406 20$:
0228 407
00000020' 0228 408 ter_i_any_trm: ; any character terminator mask
00000230' 0228 409 .long 20$-10$
022C 410 .long 10$
0230 411
FFFFFFFFFF 0230 412 10$: .long ^c<0> ; all of 0- 31
FFFFFFFFFF 0234 413 .long ^c<0> ; all of 32- 63
FFFFFFFFFF 0238 414 .long ^c<0> ; all of 64- 95
FFFFFFFFFF 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$m_cra16>
00000040 0052432D 0258 424 .long ^a/-CR/, fab$m_tecfmt!< 0a16>
00010040'00005446 0260 425 .long ^a/FT/, fab$m_tecfmt!<fab$m_ftna16>
00010040'004E5446 0268 426 .long ^a/FTN/, fab$m_tecfmt!<fab$m_ftna16>
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$, ^a/INI/ ; :EGINIS$ 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 .ascid ".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
```

```
00000000 0004 0004 0000 465 .word 4, dvi$_devclass
00000000 00000000 0004 466 .long devclass, 0
00000000 0006 0004 000C 467 .word 4, dvi$_devtype
00000000 00000001 0010 468 .long devtype, 0
00000000 0008 0004 0018 469 .word 4, dvi$_devbufsiz
00000000 00000002 001C 470 .long devbufsiz, 0
00000000 000A 0004 0024 471 .word 4, dvi$_devdepend
00000000 00000004 0028 472 .long devdepend, 0
00000000 001C 0004 0030 473 .word 4, dvi$_devdepend2
00000000 00000008 0034 474 .long devdepend2, 0
00000000 0020 0040 003C 475 .word 64, dvi$_devnam
00000000 0000000C 0040 476 .long devnam, 0
00000000 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
```

	000000EE'010E0000'	00E6	518	cli_null:		; a null string for string building
		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 tecocctl
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$: .long ^c< <1a8>! - ; all of 0- 31 except bs,
0034 581 <1a9>! - ; tab,
0034 582 <1a10>! - ; lf,
0034 583 <1a11>! - ; vt,
0034 584 <1a12>> ; ff
00000000 0034 585 .long 0 ; none of 32- 63
00000000 0038 586 .long 0 ; none of 64- 95
80000001 003C 587 .long < <1a<96-96>>! - ; none of 96-127 except accent grave,
0040 588 <1a<127-96>>> ; del
FFFFFFFFFF 0040 589 .long ^c<0> ; all of 128-159
FFFFFFFFFF 0044 590 .long ^c<0> ; all of 160-191 (initially...)
FFFFFFFFFF 0048 591 .long ^c<0> ; all of 192-223 (initially...)
FFFFFFFFFF 004C 592 .long ^c<0> ; all of 224-255 (initially...)
0050 593 20$:
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
```

```
000000E8 00E4 605 saved_sp: ; saved sp value for error exits
00000000 00E4 606 .blk1
00000000 00E8 607 still_free: ; amount of memory still free
00000000 00E8 608 .long 0
00000000 00EC 609 ctrlz_cnt: ; count of consecutive control/z's
00000000 00EC 610 .long 0
0000 00F0 611 ter_o_cc: ; terminal output carriage control
0001 00F0 612 .word 0
0001 00F2 613 output_sys_vfc: ; sys$output print control
0001 00F2 614 .word 1
0001 00F4 615 ter_o_cc: ; terminal output carriage control
0001 00F4 616 .word 0
0001 00F2 617 output_sys_vfc: ; sys$output print control
0001 00F2 618 .word 1
0001 00F4 619 .align word
0001 00F4 620 .align word
0001 00F4 621 ter_i_chan: ; channel # for terminal input
0001 00F4 622 .word 0
0001 00F6 623 ter_o_chan: ; channel # for terminal output
0001 00F6 624 .word 0
0001 00F8 625 ter_c_chan: ; channel # for terminal control/c ast
0001 00F8 626 .word 0
0001 00FA 627 ter_o_unit: ; terminal output device unit
0001 00FA 628 .word 0
0001 00FC 629 .align byte
0001 00FC 630 ctrlc_flag: ; exit on second control/c flip/flop
0001 00FC 631 .byte 0
0001 00FD 632 ctrlc_flag: ; control/o in effect flag
0001 00FD 633 .byte 0
0001 00FE 634 ter_o_force: ; terminal output being forced flag
0001 00FE 635 .byte 0
0001 00FF 636 ter_o_pend: ; terminal output pending count
0001 00FF 637 .byte -1
0001 0100 638 exiting_flag: ; exiting flag
0001 0100 639 .byte 1
0001 0101 640 .align long
0001 0104 641 err_msgvec: ; error message vector
0001 0104 642 .blk1 3
0001 0110 643 .no how meb
0001 0110 644 input_nor fab: ; fab for normal input
0001 0110 645 $fab - ; allocate a fab
0001 0110 646 fac=get, - ; allow gets
0001 0110 647 fna=file_spec_buf, - ; file name will come from here
0001 0110 648
```

```
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
0108 686
0108 687 input_nor_xab: ; xab for normal input
0108 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,rab>, - ; 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 $fab - ; allocate a fab
0274 700 fac=<put,trn>, - ; allow puts & truncates
0274 701 fna=file_spec_buf, - ; file name will come from here
0274 702 nam=output_nor_nam, - ; catch the resultant filespec
0274 703 org=seq ; organization is sequential
000002C8 0204 704 .assume fab$l_tecsts eq .-output_nor_fab ; private status longword
0204 705 .blkl
00000334 0208 706 .assume fab$l_tecrab eq .-output_nor_fab ; private pointer to the correct rab
0208 707 .long output_nor_rab
000002D4 020C 708 .assume fab$q_tecque eq .-output_nor_fab ; private data line queue
020C 709 .blkq
0204 710
0204 711 output_nor_nam: ; nam for normal output
0204 712 $nam - ; allocate a nam
0204 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=sq, -                     ; (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
00000000 03C8 733 .assume fab$l_tecsts eq .-indir_cmd_fab
03C8 734          .long 0                      ; (pre-set) private status longword
00000440' 03CC 735 .assume fab$l_tecrab eq .-indir_cmd_fab
03CC 736          .long indir_cmd_rab        ; private pointer to the correct rab
000003D0'000003D0' 03D0 737 .assume fab$q_tecque eq .-indir_cmd_fab
03D0 738          .long 10$, 10$             ; (pre-set) private data line queue
00000891' 03D8 739 .assume fab$l_tecdsp eq .-indir_cmd_fab
03D8 740          .long getbyt_first            ; (pre-set) private dispatch longword
00 00 00 0A 03DC 741 .assume fab$l_tecctl eq .-indir_cmd_fab
03DC 742          .byte 10, 0, 0, 0            ; (pre-set) private control bytes
03E0 743
03E0 744 indir_cmd_nam:                          ; nam for 'ei'
03E0 745          $nam -                          ; alloc a nam
03E0 746          rsa=indir_cmd_rsa, -          ; resultant filespec goes here
03E0 747          rss=nam$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,rab>, -               ; 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          $fab -                          ; allocate a fab
0484 759          fac=get, -                     ; allow gets
0484 760          fna=file_spec_buf, -          ; file name will come from here
0484 761          nam=input_alt_nam, -          ; catch the resultant filespec
0484 762          rat=cr, -                     ; (pre-set) attributes of implied cc
0484 763          rfm=var, -                     ; (pre-set) record format of variable
0484 764          xab=input_alt_xab             ; catch the protection code
000004D8 04D4 765 .assume fab$l_tecsts eq .-input_alt_fab
04D4 766          .blk 1                        ; private status longword
000005A4' 04D8 767 .assume fab$l_tecrab eq .-input_alt_fab
04D8 768          .long input_alt_rab           ; private pointer to the correct rab
000004E4 04DC 769 .assume fab$q_tecque eq .-input_alt_fab
04DC 770          .blk 4                        ; private data line queue
```

```
000004E8 04E4 771 .assume fab$l_tecdsp eq .-input_alt_fab
000004EC 04E4 772 .blkf ; private dispatch longword
04E8 773 .assume fab$l_tecctl eq .-input_alt_fab
04E8 774 .blkf ; 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,fah>, - ; 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
0638 799 .blkf ; private status longword
063C 800 .assume fab$l_tecrab eq .-output_alt_fab
063C 801 .long output_alt_rab ; private pointer to the correct rab
0640 802 .assume fab$q_tecque eq .-output_alt_fab
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 '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
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' occurrence 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$l_tecdsp eq .-input_sys_fab ; (pre-set) private dispatch longword
00000B91' 0800 843 .long getbyt_first ;
0804 844 .assume fab$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
000008E4 08E0 875 file_spec_opt: ; file specification options
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' occurrence 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 .blk1
0000000C 000C 942 getdvi_info_len = .-getdvi_info
0000000C 000C 943
0000004C 000C 944 devnam: ; device name string
0000004C 004C 945 .blkb 64
0000004C 004C 946
00000050 004C 947 unit: ; device unit number
00000050 004C 948 .blk1
00000050 0050 949
00000050 0050 950 .noshw meb
00000050 0050 951
00000050 0050 952 cli_req_getcmd: ; request block to get command line
00000050 0050 953 $cli_reqdesc - ; define cli request descriptor block
00000050 0050 954 rctype=cli$k_getcmd ; to get the whole command line
00000050 006C 955
00000050 006C 956 .show meb
00000050 006C 957
00000050 006C 958 cli_result: ; cli parse result string
00000050 006C 959 .word 0
00000050 006E 960 .byte dsc$k_dtype_t
00000050 006F 961 .byte dsc$k_class_d
00000050 0070 962 .long 0
00000050 0074 963
00000050 0074 964 cli_command_line: ; built up cli parse command line
00000050 0074 965 .word 0
00000050 0076 966 .byte dsc$k_dtype_t
00000050 0077 967 .byte dsc$k_class_d
00000050 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 985 ter_i_buf_pre:
00000600 0403 986      .blkb   ter_i_siz-<.-ter_i_buf>
0600 987
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 tecoeexeini
0000 1002
OFFC 0000 1003 .entry tec$teco, ^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11> ; startup entry point
0002 1004
0002 1005 .sbttl Initialization code
0002 1006
0002 1007 ; set up permanent pointer registers
0002 1008
0002 1009 mnegl #1, r10 ; junk up the ctl$al_cmctx 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$m_canceltrlo, io$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 #pdlrpt, w^tecopd(r11) ; set teco's pdl start
00'AB 0000'8F B0 002C 1018 movw #pdlrpt, 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 #inpnr, b^inpnr(r11) ; set input pointer to normal input
00'AB 0000'8F B0 004D 1023 movw #oupnr, b^oupnr(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^rwsiz(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
00 61 52 D4 00A9 1040 clrl r2 ; start at (relative) bit 0
00000000'EF 42 B5 00AF 1041 10$: bbss r2, (r1), 20$ ; ensure that this bit is set
04 19 00B6 1042 20$: tstw cnv8bt[r2] ; a hex pair for conversion?
00 61 52 E5 00B8 1043 blss 30$ ; yes, leave the bit set
E7 52 00000080 8F F2 00BC 1044 bbcc r2, (r1), 30$ ; no, clear the bit
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 $getdvi_s - ; get device characteristics
00C4 1051 devnam=w^ter_i devnam, - ; of terminal input device
00C4 1052 itmlst=w^getdvi_itmlst ; using this item list
                                CLRQ -(SP)
                                PUSHL #0
                                PUSHL #0
                                PUSHAL w^getdvi_itmlst
                                PUSHAL w^ter_i devnam
                                MOVZWL #0, -(SP)
                                PUSHL #0
                                CALLS #8, G^SYSSGETDVI
                                blbc r0, 20$ ; non-terminal if any failure
                                cmpb w^devclass, #dc$_term ; a terminal?
                                bneq 20$ ; nope
                                bbc #tt$_lower, w^devdepend, 10$ ; lowercase?
                                bs #tec$_et$lc, b^etype(r11) ; yes, allow lower case input
                                bisb s^#1attec$_et$lc, b^etype(r11)
                                $assign_s - ; assign channel for terminal input
                                devnam=w^ter_i devnam, - ; terminal input device name
                                chan=w^ter_i_chan ; into terminal input channel
                                CLRQ -(SP)
                                PUSHAL w^ter_i_chan
                                PUSHAL w^ter_i devnam
                                CALLS #4, G^SYSS$ASSIGN
                                bsbw success_else_die ; abort unless success completion
                                brb 60$ ; go continue
                                bs #tec$_et$lc, b^etype(r11) ; lower case for non-terminals
                                bisb s^#1attec$_et$lc, b^etype(r11)
                                $open - ; open
                                fab=w^input_sys_fab ; sys$input
                                PUSHAL w^input_sys_fab
                                CALLS $$$TMPT, G^SYSS$OPEN
                                movl w^input_sys_fab+fab$_stb, w^err_msgvec+8 ; save the STV value
                                bsbw success_else_die ; abort unless success completion
                                cmpb w^input_sys_fab+fab$_rfm, #fab$_vfc ; vfc record format?
                                bneq 30$ ; nope
                                movl #rms$_fsz, r0 ; pre-set bad vfc size error
                                cmpb w^input_sys_fab+fab$_fsz, #input_vfc_siz ; vfc size correct?
                                bgtru 50$ ; nope, go die with an error
                                cmpb w^input_sys_fab+fab$_rfm, #fab$_stm ; stream record format?
                                bneq 40$ ; nope
                                bisb #fab$_m_cr, w^input_sys_fab+fab$_rat ; yep, set implied lf/cr
                                $connect - ; connect
                                rab=w^input_sys_rab ; to it
                                PUSHAL w^input_sys_rab
                                CALLS $$$TMPT, G^SYSS$CONNECT
                                movl w^input_sys_rab+rab$_stb, w^err_msgvec+8 ; save the STV value
                                bsbw success_else_die ; abort unless success completion
                                ; continue

00000000'GF 00 7E 7C 00C4 1047 ; get terminal input device's characteristics
0061'CF 00 DD 00C4 1048
00000000'GF 28 50 DF 00C4 1049 getdvi_ter_i: ; get terminal input characteristics
00'8F 0000'CF 20 7F 00C4 1050 $getdvi_s - ; get device characteristics
04 0004'CF 07 3C 00C4 1051 devnam=w^ter_i devnam, - ; of terminal input device
00'AB 04 88 DD 00C4 1052 itmlst=w^getdvi_itmlst ; using this item list
                                CLRQ -(SP)
                                PUSHL #0
                                PUSHL #0
                                PUSHAL w^getdvi_itmlst
                                PUSHAL w^ter_i devnam
                                MOVZWL #0, -(SP)
                                PUSHL #0
                                CALLS #8, G^SYSSGETDVI
                                blbc r0, 20$ ; non-terminal if any failure
                                cmpb w^devclass, #dc$_term ; a terminal?
                                bneq 20$ ; nope
                                bbc #tt$_lower, w^devdepend, 10$ ; lowercase?
                                bs #tec$_et$lc, b^etype(r11) ; yes, allow lower case input
                                bisb s^#1attec$_et$lc, b^etype(r11)
                                $assign_s - ; assign channel for terminal input
                                devnam=w^ter_i devnam, - ; terminal input device name
                                chan=w^ter_i_chan ; into terminal input channel
                                CLRQ -(SP)
                                PUSHAL w^ter_i_chan
                                PUSHAL w^ter_i devnam
                                CALLS #4, G^SYSS$ASSIGN
                                bsbw success_else_die ; abort unless success completion
                                brb 60$ ; go continue
                                bs #tec$_et$lc, b^etype(r11) ; lower case for non-terminals
                                bisb s^#1attec$_et$lc, b^etype(r11)
                                $open - ; open
                                fab=w^input_sys_fab ; sys$input
                                PUSHAL w^input_sys_fab
                                CALLS $$$TMPT, G^SYSS$OPEN
                                movl w^input_sys_fab+fab$_stb, w^err_msgvec+8 ; save the STV value
                                bsbw success_else_die ; abort unless success completion
                                cmpb w^input_sys_fab+fab$_rfm, #fab$_vfc ; vfc record format?
                                bneq 30$ ; nope
                                movl #rms$_fsz, r0 ; pre-set bad vfc size error
                                cmpb w^input_sys_fab+fab$_fsz, #input_vfc_siz ; vfc size correct?
                                bgtru 50$ ; nope, go die with an error
                                cmpb w^input_sys_fab+fab$_rfm, #fab$_stm ; stream record format?
                                bneq 40$ ; nope
                                bisb #fab$_m_cr, w^input_sys_fab+fab$_rat ; yep, set implied lf/cr
                                $connect - ; connect
                                rab=w^input_sys_rab ; to it
                                PUSHAL w^input_sys_rab
                                CALLS $$$TMPT, G^SYSS$CONNECT
                                movl w^input_sys_rab+rab$_stb, w^err_msgvec+8 ; save the STV value
                                bsbw success_else_die ; abort unless success completion
                                ; continue

00000000'GF 07A0'CF 01 DF 00C4 1047 ; get terminal input device's characteristics
010C'CF 07AC'CF 30 FB 00C4 1048
03 07BF'CF 0E 7F 00C4 1049 getdvi_ter_i: ; get terminal input characteristics
50 00000000'8F 01 3C 00C4 1050 $getdvi_s - ; get device characteristics
0C 07DF'CF 1E 7F 00C4 1051 devnam=w^ter_i devnam, - ; of terminal input device
04 07BF'CF 05 3C 00C4 1052 itmlst=w^getdvi_itmlst ; using this item list
07BE'CF 02 88 DD 00C4 1053 blbc r0, 20$ ; non-terminal if any failure
                                cmpb w^devclass, #dc$_term ; a terminal?
                                bneq 20$ ; nope
                                bbc #tt$_lower, w^devdepend, 10$ ; lowercase?
                                bs #tec$_et$lc, b^etype(r11) ; yes, allow lower case input
                                bisb s^#1attec$_et$lc, b^etype(r11)
                                $assign_s - ; assign channel for terminal input
                                devnam=w^ter_i devnam, - ; terminal input device name
                                chan=w^ter_i_chan ; into terminal input channel
                                CLRQ -(SP)
                                PUSHAL w^ter_i_chan
                                PUSHAL w^ter_i devnam
                                CALLS #4, G^SYSS$ASSIGN
                                bsbw success_else_die ; abort unless success completion
                                brb 60$ ; go continue
                                bs #tec$_et$lc, b^etype(r11) ; lower case for non-terminals
                                bisb s^#1attec$_et$lc, b^etype(r11)
                                $open - ; open
                                fab=w^input_sys_fab ; sys$input
                                PUSHAL w^input_sys_fab
                                CALLS $$$TMPT, G^SYSS$OPEN
                                movl w^input_sys_fab+fab$_stb, w^err_msgvec+8 ; save the STV value
                                bsbw success_else_die ; abort unless success completion
                                cmpb w^input_sys_fab+fab$_rfm, #fab$_vfc ; vfc record format?
                                bneq 30$ ; nope
                                movl #rms$_fsz, r0 ; pre-set bad vfc size error
                                cmpb w^input_sys_fab+fab$_fsz, #input_vfc_siz ; vfc size correct?
                                bgtru 50$ ; nope, go die with an error
                                cmpb w^input_sys_fab+fab$_rfm, #fab$_stm ; stream record format?
                                bneq 40$ ; nope
                                bisb #fab$_m_cr, w^input_sys_fab+fab$_rat ; yep, set implied lf/cr
                                $connect - ; connect
                                rab=w^input_sys_rab ; to it
                                PUSHAL w^input_sys_rab
                                CALLS $$$TMPT, G^SYSS$CONNECT
                                movl w^input_sys_rab+rab$_stb, w^err_msgvec+8 ; save the STV value
                                bsbw success_else_die ; abort unless success completion
                                ; continue

00000000'GF 0808'CF 01 DF 00C4 1047 ; get terminal input device's characteristics
010C'CF 0814'CF 30 FB 00C4 1048
00000000'GF 0808'CF 01 DF 00C4 1049 getdvi_ter_i: ; get terminal input characteristics
010C'CF 0814'CF 30 FB 00C4 1050 $getdvi_s - ; get device characteristics
                                devnam=w^ter_i devnam, - ; of terminal input device
                                itmlst=w^getdvi_itmlst ; using this item list
                                CLRQ -(SP)
                                PUSHL #0
                                PUSHL #0
                                PUSHAL w^getdvi_itmlst
                                PUSHAL w^ter_i devnam
                                MOVZWL #0, -(SP)
                                PUSHL #0
                                CALLS #8, G^SYSSGETDVI
                                blbc r0, 20$ ; non-terminal if any failure
                                cmpb w^devclass, #dc$_term ; a terminal?
                                bneq 20$ ; nope
                                bbc #tt$_lower, w^devdepend, 10$ ; lowercase?
                                bs #tec$_et$lc, b^etype(r11) ; yes, allow lower case input
                                bisb s^#1attec$_et$lc, b^etype(r11)
                                $assign_s - ; assign channel for terminal input
                                devnam=w^ter_i devnam, - ; terminal input device name
                                chan=w^ter_i_chan ; into terminal input channel
                                CLRQ -(SP)
                                PUSHAL w^ter_i_chan
                                PUSHAL w^ter_i devnam
                                CALLS #4, G^SYSS$ASSIGN
                                bsbw success_else_die ; abort unless success completion
                                brb 60$ ; go continue
                                bs #tec$_et$lc, b^etype(r11) ; lower case for non-terminals
                                bisb s^#1attec$_et$lc, b^etype(r11)
                                $open - ; open
                                fab=w^input_sys_fab ; sys$input
                                PUSHAL w^input_sys_fab
                                CALLS $$$TMPT, G^SYSS$OPEN
                                movl w^input_sys_fab+fab$_stb, w^err_msgvec+8 ; save the STV value
                                bsbw success_else_die ; abort unless success completion
                                cmpb w^input_sys_fab+fab$_rfm, #fab$_vfc ; vfc record format?
                                bneq 30$ ; nope
                                movl #rms$_fsz, r0 ; pre-set bad vfc size error
                                cmpb w^input_sys_fab+fab$_fsz, #input_vfc_siz ; vfc size correct?
                                bgtru 50$ ; nope, go die with an error
                                cmpb w^input_sys_fab+fab$_rfm, #fab$_stm ; stream record format?
                                bneq 40$ ; nope
                                bisb #fab$_m_cr, w^input_sys_fab+fab$_rat ; yep, set implied lf/cr
                                $connect - ; connect
                                rab=w^input_sys_rab ; to it
                                PUSHAL w^input_sys_rab
                                CALLS $$$TMPT, G^SYSS$CONNECT
                                movl w^input_sys_rab+rab$_stb, w^err_msgvec+8 ; save the STV value
                                bsbw success_else_die ; abort unless success completion
                                ; 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
			0158		CLRQ -(SP)		
			015A		PUSHL #0		
			015C		PUSHL #0		
			015E		PUSHAL w^getdvi_itmlst		
			0162		PUSHAQ w^ter_o devnam		
			0166		MOVZWL #0,-(SP)		
			0169		PUSHL #0		
			016B		CALLS #8,G^SYSS\$GETDVI		
			0172	1089	#0		: guess at no terminal data to look at
			0174	1090	blbc r0, 70\$		: non-terminal if any failure
			0177	1091	cmpb w^devclass, #dc\$_term		: a terminal?
			017D	1092	bneq 70\$		: nope
			017F	1093	movab w^devnam, r0		: yep, address the device name string
			0184	1094	10\$: cmpb (r0)+, #^a/_/		: a leading underscore?
			0188	1095	beql 10\$		: yes, go ignore it
			018A	1096	cmpw -1(r0), #^a/TT/		: is it "TTC" format?
			0190	1097	bneq 20\$		: no, so no decode
			0192	1098	subb3 #^a/A/, 1(r0), w^ter_o_unit+1		: set controller # * 256.
			019A	1099	addw w^unit, w^ter_o_unit		: plus unit number
			01A1	1100	20\$: movab w^devdepend, r0		: address device dependent bits
			01A6	1101	bbs #tt\$V_lower, (r0), 30\$		: lowercase?
			01AA	1102	decw b^euflag(r11)		: yes, say no case flagging
			01AD	1103	30\$: bbs #tt\$V_wrap, (r0), 40\$		: wrap?
			01B1	1104	bs #tec\$V_et\$tru, b^etype(r11)		: no, indicate truncated lines
			01B1		bbcs #tec\$V_et\$tru, b^etype(r11), 30002\$		
			01B6	30002\$:			
			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
			01BA		bbcs #tec\$V_et\$8bt, b^etype(r11), 30003\$		
			01BF	30003\$:			
			01BF	1107	50\$: bbc #tt\$V_scope, (r0), 60\$		: scope?
			01C3	1108	movc #getdvi_info_len, w^getdvi_info, errbuf		: save scope info
			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 output device name
			01D4	1112	chan=w^ter_o_chan		: into terminal output channel
			01D4		CLRQ -(SP)		
			01D6		PUSHAQ w^ter_o_chan		
			01DA		PUSHAQ w^ter_o_devnam		
			01DE		CALLS #4,G^SYSS\$ASSIGN		
			01E5	1113	bsbw success_else_die		: abort unless success completion
			01E8	1114	brb 80\$		: go continue
			01EA	1115			
			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
			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
			01F2		PUSHAL w^output_sys_fab		
			01F6		CALLS #\$\$,TMP1,G^SYSS\$CREATE		

```
010C'CF 0858'CF D0 01FD 1120
          FEA0' 30 0204 1121
          0207 1122
          0207 1123
          089C'CF DF 0207
00000000'GF 01 FB 0208
010C'CF 08A8'CF D0 0212 1124
          FE8B' 30 0219 1125
          021C 1126 80$:
```

```
movl w^output_sys_fab+fab$sl_stv, w^err_msgvec+8 ; save STV value
bsbw success_else_die ; abort unless success completion
$connect - ; connect
          rab=w^output_sys_rab ; to it
PUSHAL w^output_sys_rab
CALLS $$$TMP1,G^SYS$CONNECT
movl w^output_sys_rab+rab$sl_stv, w^err_msgvec+8 ; save STV value
bsbw success_else_die ; abort unless success completion
          ; continue
```

```
021C 1128 ; get terminal command device's characteristics
021C 1129
021C 1130 getdvi_ter_c: ; get terminal command characteristics
021C 1131 $getdvi_s - ; get device characteristics
021C 1132 devnam=w^ter_c devnam, - ; of terminal command device
021C 1133 itmlst=w^getdvi_itmlst ; using this item list
021C 1134 CLRQ -(SP)
021E 1135 PUSHL #0
0220 1136 PUSHL #0
0222 1137 PUSHAL w^getdvi_itmlst
0226 1138 PUSHAQ w^ter_c devnam
022A 1139 MOVZWL #0, -(SP)
022D 1140 PUSHL #0
022F 1141 CALLS #8, G^SYSS$GETDVI
0236 1142 r0, 10$ ; non-terminal if any failure
0239 1143 cmpb w^devclass, #dc$_term ; a terminal?
023F 1144 bneq 10$ ; nope
0241 1145 $assign_s - ; assign channel for control/c ast's
0241 1146 devnam=w^ter_c devnam, - ; terminal command device name
0241 1147 chan=w^ter_c_chan ; into terminal control/c ast channel
0241 1148 CLRQ -(SP)
0243 1149 PUSHAW w^ter_c_chan
0247 1150 PUSHAQ w^ter_c devnam
024B 1151 CALLS #4, G^SYSS$ASSIGN
0252 1152 bsbw success_else_die ; abort unless success completion
0255 1153 bsbw enable_ctrlcast ; go enable the control/c ast
0258 1154 bsbw success_else_die ; abort unless success completion
025B 1155 10$ ; continue
```

7E 7C 00 DD 00 DD 0000'CF DF 007B'CF 7F 7E 00 3C 00 DD 00000000'GF 08 FB 22 50 E9 00'8F 0000'CF 91 1A 12 0241 1137 0241 1138 0241 1139 7E 7C 00F8'CF 3F 007B'CF 7F 00000000'GF 04 FB FE52' 30 FF74' 30 FE4C' 30 0252 1140 0255 1141 0258 1142 025B 1143 10\$:

```
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
025B 1150 movl w^cli_req_getcmd, r6 ; get address of request block
0260 1151 pushal (r6) ; set address of request block
0262 1152 calls #1, g^sys$cli ; go get the command line
0269 1153 tstw r0 ; the special 'no command' status?
026B 1154 bneq 10$ ; nope
026D 1155 brw 210$ ; yes, go continue
0270 1156
0270 1157 10$: bsbw success_else_die ; abort unless success completion
0273 1158 cmpb cli$b_rqstat(r6), #cli$verb_edit ; is this EDIT/TECO?
0278 1159 beql 20$ ; yes
027A 1160 brw 110$ ; no
027D 1161
027D 1162 20$: movaq w^cli_verb_teco, r5 ; set TECO verb
0282 1163 bsbw 200$ ; as command line verb
0285 1164 pushaq w^cli_qual_command ; /COMMAND
0289 1165 calls #1, g^cli$present ; present?
0290 1166 blbc r0, 30$ ; no, go use /NOINI
0293 1167 pushaq w^cli_result ; address result string
0297 1168 pushaq w^cli_qual_command ; /COMMAND=file
029B 1169 calls #2, g^cli$get_value ; have a value?
02A2 1170 blbc r0, 40$ ; no
02A5 1171 pushaq w^cli_result ; yes, form
02A9 1172 pushaq w^cli_dollar ; '$file'
02AD 1173 pushaq w^cli_result ; in the result
02B1 1174 calls #3, g^str$concat ; using concatenation
02B8 1175 bsbw success_else_die ; abort unless success completion
02B8 1176 $crelog_s - ; create a logical
02B8 1177 tblflg = #2, - ; use the process table
02B8 1178 lognam = w^cli_init, - ; logical is TEC$INIT
02B8 1179 eqlnam = w^cli_result ; with this equivalence string
02B8
02B8 PUSHAL #0
02BD PUSHAQ w^cli_result
02C1 PUSHAQ w^cli_init
02C5 PUSHAL #2
02C7 CALLS #4, G^SYS$CRELOG
02CE 1180 bsbw success_else_die ; abort unless success completion
02D1 1181 brb 40$ ; else continue
02D3 1182
02D3 1183 30$: movaq w^cli_no_ini, r5 ; set /NOINI
02D8 1184 bsbw 200$ ; and go add it
02DB 1185 40$: pushaq w^cli_qual_create ; /CREATE
02DF 1186 calls #1, g^cli$present ; present?
02E6 1187 blbs r0, 50$ ; yes
02E9 1188 movaq w^cli_no_create, r5 ; no, set /NOCREATE
02EE 1189 bsbw 200$ ; and go add it
02F1 1190 50$: pushaq w^cli_qual_memory ; /MEMORY
02F5 1191 calls #1, g^cli$present ; present?
02FC 1192 blbs r0, 60$ ; yes
02FF 1193 movaq w^cli_no_memory, r5 ; no, set /NOMEMORY
0304 1194 bsbw 200$ ; and go add it
0307 1195 60$: movaq w^cli_space, r5 ; set ' ' as the next separator
030C 1196 pushaq w^cli_result ; address result string
```

```
00000000'0144'CF 7F 0310 1197 pushaq w^cli_qual_execute ; /EXECUTE=file
00000000'GF 02 FB 0314 1198 calls #2, g^cli$get_value ; present with a value?
54 50 DO 031B 1199 movl r0, r4 ; remember /EXECUTE's existence
18 50 E9 031E 1200 blbc r0, 70$ ; no
00000078'FF 474E554D 8F DO 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
006C'CF 7F 0339 1206 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_rsize(r6) ; set command line length
0C A6 0078'CF DO 0395 1229 movl w^cli_command_line+4, cli$a_rqaddr(r6) ; and address
53 00'AB 3C 039B 1230 110$: movzwl b^qrsfor(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$w_rsize(r6), r1 ; get command line's size
52 0C A6 DO 03B6 1237 movl cli$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 movab 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 1241 movab w^tmp_string_buf, r0
70 50 DO 03C7 1242 movl r0, -(r0)
70 3F 3C 03CA 1243 movzwl #tmp_string_siz, -(r0)
03CD 1241 $strnlog_s - ; translate
03CD 1242 lognam=ini_dcd_lognam, - ; logical name 'TECO'
03CD 1243 rs(buf=(r0) ; putting result into the temp
00 DD 03CD
00 DD 03CF
00 DD 03D1
60 7F 03D3
00 DD 03D5
0000008E'EF 7F 03D7
00000000'GF 06 FB 03DD
```

```
0000'8F 50 B1 03E4 1244 cmpw r0, #ss$_notran ; did anything happen?
OF 13 03E9 1245 beql 130$ ; nope, use pre-set 'SYS$LOGIN:TECO'
FCB9' 30 03EB 1246 bsbw success else die ; abort unless success completion.
2C B6 4F434554 8F D0 03EE 1247 movl #^a/TECO/, @fab$_fna(r6) ; set file name of simply 'TECO'
34 A6 04 90 03F6 1248 movb #4, fab$_fns(r6) ; and its length
03FA 1249 130$: $open - ; open
03FA 1250 fab=(r6) ; the command decoder file file
00000000'GF 66 DF 03FA PUSHAL (r6)
01 FB 03FC CALLS #$_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$_rfm(r6), #fab$_vfc ; vfc record format?
06 12 040A 1253 bneq 140$ ; nope
0C 3F A6 91 040C 1254 cmpb fab$_fsz(r6), #input_vfc_siz ; vfc size correct?
2A 1A 0410 1255 bgtru 160$ ; nope, so don't use it...
04 1F A6 91 0412 1256 140$: cmpb fab$_rfm(r6), #fab$_stm ; stream record format?
04 12 0416 1257 bneq 150$ ; nope
1E A6 02 88 0418 1258 bisb #fab$_m_cr, fab$_rat(r6) ; always say implied lf/cr for stream
041C 1259 150$: $connect - ; connect
041C 1260 rab=@fab$_l_tecrab(r6) ; the correct rab
00000000'GF 54 B6 DF 041C PUSHAL @fab$_l_tecrab(r6)
01 FB 041F CALLS #$_TMP1,G^SYS$CONNECT
13 50 E9 0426 1261 blbc r0, 160$ ; branch if failure of any type
57 00000000'EF DE 0429 1262 movl cmdprm, r7 ; get where to store fab pointer
67 56 D0 0430 1263 movl r6, (r7) ; set file as open
08A6' 30 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$: $close - ; close
043C 1269 fab=(r6) ; the (bad) command decoder file file
00000000'GF 66 DF 043C PUSHAL (r6)
01 FB 043E CALLS #$_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
```

```
004'CF 00100008 8F CA 048A 1301 bicl #tt$m_halfdup!tt$m_escape, w^devdepend ; save the original characteristics
004'CF 0D5D' 30 0493 1302 bsbw tec$setmode ; go reset the terminal mode(s)
004'CF 8E' DO 0496 1303 movl (sp)+, w^devdepend ; restore original characteristics
FC09' 30 049B 1304 bsbw success_else_die ; abort unless success completion
049E 1305 10$: $dclcmh_s - ; declare compatibility mode handler
049E 1306 ; address=w^tec$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 03 DF 04A2 ; PUSHAL w^tec$cmtrap
00000000'GF 03 FB 04A6 ; CALLS #3,G^SYSDCLCMH
FBF7' 30 04AD 1308 bsbw success_else_die ; abort unless success completion
52 7C 04B0 1309 clrq r2 ; clear r2, r3
54 8E DO 04B2 1310 movl (sp)+, r4 ; set r4 if scope terminal, else clear
55 5B DO 04B5 1311 movl r11, r5 ; set teco's r5 r/w area pointer
56 00000000'EF 3E 04B8 1312 movaw spset, r6 ; set to reset the sp stack
83C00000 8F DD 04BF 1313 pushl #psl$m_cm!<psl$sc_user@psl$v_curmod>!<psl$sc_user@psl$v_prvmod>
00000000'EF 3F 04C5 1314 pushaw tecost ; stack startup pc7psl pair
FBA9' 31 04CB 1315 brw start_teco ; go purge working set, then start up!
```

```
04CE 1317 .sbtcl Compatibility mode trap handler
04CE 1318
04CE 1319      org      tecoexe
0000 1320
0000 1321      .align   page
0000 1322
0000 1323 tec$cmtrap:      ; compatibility mode traps come here
SA 50 20 C3 0000 1324      subl3    #i_bias, r0, r10      ; form ctl$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
50 10 10 0019 1331 10$:      bsbb     20$      ; go dispatch on exception code
52 8A 7D 001B 1332      movq      (r10)+, r0      ; restore r0, r1
54 8A 7D 001E 1333      movq      (r10)+, r2      ; restore r2, r3
56 8A 7D 0021 1334      movq      (r10)+, r4      ; restore r4, r5
7E 6A 7D 0024 1335      movq      (r10)+, r6      ; restore sp, junk r7
02 002A 1336      movq      (r10), -(sp)      ; stack pc/psl pair
002B 1337      rei      ; back to whatever...
00E4'CF 5E D0 002B 1338
0003'8F 01 1C AA AF 002B 1339 20$:      movl      sp, w^saved_sp      ; save calling sp for error exits
0037 1340      casew     i_code(r10), #1, #<<40$-30$>/2>-1 ; enter proper routine...
0207' 0037 1341 30$:      ; reference only
033A' 0037 1342      .word     tec$wait-30$      ; 1 => bpt
000B' 0039 1343      .word     tec$output-30$      ; 2 => iot
053F' 003B 1344      .word     50$-30$      ; 3 => emt
003D 1345      .word     tec$input-30$      ; 4 => trap
0423' 31 003F 1346 40$:      ; reference only
0042 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 $purgws_s - ; purge our working set
0077 1385 ; do the whole thing...
                                inadr=b^10$
                                PUSHAQ b^10$
                                CALLS #1,G^SYSS$PURGWS
                                success_or_announce
                                r0
0077 1386 bsbw ; announce any failure...
0077 1387 clrq ; clear r0, r1
0077 1388 rei ; enter compatibility mode...
0077 1389
0087 1390 10$: .long 0 ; purge from 00000000
0088 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$: movl w^err_msgvec+8, r1 ; point to message vector spot
71 000F0001 8F D0 0094 1398 movl r0, -(r1) ; load the error status code
00' 06 A1 B1 009E 1399 movl #<<8!4!2!1>a16>!1, -(r1) ; fac, sev, id, & text; arg count = 1
02 12 00A2 1400 cmpw 6(r1), s^#rms$_normal/65536 ; is this an RMS error?
61 B6 00A4 1401 bneq 20$ ; nope
05 00A6 1402 incw (r1) ; yes, include the STV value in count
00A7 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 $exit_s - ; force an image exit
00B9 1413 code = r0 ; with the reason code
PUSHL r0
CALLS #1,G^SYS$EXIT
00000000'GF 50 DD 00B9
01 FB 00B8
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 $PUTMSG to format the message
00C7 1419 msgvec = (r1) ; using the built up message vector
00 DD 00C7
00 DD 00C9
00 DD 00CB
00 DD 00CD
61 DF 00CD
00000000'GF 04 FB 00CF
00D6 1420 ;bsbw
05 00D6 1421 rsb ; announce any failure...
00D7 1422 ; exit
00D7 1423 success_or_abrt: ; check for success; abort if not
6A 50 E8 00D7 1424 blbs r0, 60$ ; 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
010C'CF 54 A6 D0 00E7 1430 fab$l_tecrab(r6), r1 ; get the rab pointer
OC A1 D0 00EB 1431 movl rab$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
PUSHAL (r6)
CALLS #$$,TMP1,G^SYS$CLOSE
00000000'GF 66 DF 00F3
01 FB 00F5
01 BA 00FC 1435 popr #^m<r0> ; restore the failure status
43 50 E8 00FE 1436 success_or_err: ; check for success; error if not
8C 10 0101 1437 blbs r0, 60$ ; exit if success
0103 1438 30$: bsbb 10$ ; go format the message vector
6A 2222 8F 3C 0103 1439 rad50 ERR
0103 1440 movzwl #$$$$$, i_r0(r10) ; set rad50 code for ERR into saved r0
```

```
00000000'8F 04 A1 D1 0108 1441      cmpl 4(r1), #rms$_inf      ; was the error file not found?
                                09 12 0110 1442      bneq 40$              ; it was not
                                02 A1 01 80 0112 1443      movw #1, 2(r1)          ; it was, set only message text
                                6A 27B6 8F 3C 0116 1444      FNF
                                00000000'EF 94 0116 1445      movzwl #$$$$$$, i_r0(r10) ; set rad50 code for FNF into saved r0
                                00000000'EF 94 011B 1446      clrb errbuf          ; "empty" the error message buffer
                                0121 1447      $putmsg_s -      ; use $PUTMSG to format the message
                                0121 1448      msgvec = (r1) -      ; using the built up message vector
                                0121 1449      actrtn = b^70$      ; catch the message w/ action routine
                                00 DD 0121
                                00 DD 0123
                                5E'AF DF 0125
                                61 DF 0128
                                00000000'GF 04 FB 012A
                                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 63 94 015A 1461      clrb (r3)              ; and ensure result is asciz
                                63 94 11 015C 1462      brb 50$              ; go set the message text address
                                D6 11 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$ctrlcast: ; control/c ast's come here
0194 1485 .word ^m<> ; no need to save any registers
14 00000000'EF 0F E4 0196 1486 bbsc #tec$vet$cc, etype+r5set, 10$ ; 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$vet$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
00FD'CF FF0B 30 01B4 1491 bsbw success_or_announce ; announce any failure...
00 92 01B7 1492 mcomb #0, w^ctrlc_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
MOVZWL w^ter_o_chan, -(SP)
CALLS #1, G^SYSCANCEL
01C8 1495 bsbw success_or_announce ; announce any failure...
01CB 1496 ret ; return
01CC 1497
01CC 1498 enable_ctrlcast: ; enable the control/c ast
01CC 1499 $qio_w_s - ; set up the control/c ast with a qio
01CC 1500 chan=w^ter_c_chan, - ; using the control/c ast channel
01CC 1501 func=#io$_setmode!io$m_ctrlcast, - ; function = control/c ast
01CC 1502 p1=b^tec$ctrlcast ; go to here on control/c ast's
CLRQ -(SP)
CLRQ -(SP)
PUSHL #0
PUSHAL b^tec$ctrlcast
CLRQ -(SP)
PUSHL #0
MOVZWL #io$_setmode!io$m_ctrlcast, -(SP)
MOVZWL w^ter_c_chan, -(SP)
PUSHL #0
CALLS #12, G^SYSQIOW
05 01EC 1503 rsb ; exit
```

```
01ED 1505 .sbttl Terminal output waits
01ED 1506
01ED 1507 .enable lsb
01ED 1508
01ED 1509 tec$wait_done_lf:
01ED 1510      movb #2!1, w^ter_o_force      ; wait for all output (inc. <LF>'s)
01F2 1511      cmpw ttotptr, #ttobuf    ; say we're forcing <LF>'s also
01FB 1512      bneq tec$wait_done        ; is the terminal output buffer empty?
0201 1513      tstl w^ter_o_cc           ; nope
0203 1514      beql tec$wait_done        ; any pending, saved <LF>?
0207 1515      movw b^outdne(r11), -(sp) ; nope
020B 1516      clrl w^ter_o_cc           ; yes, save current output done flag
020E 1517      bsbw echo_tf              ; forget about saving the <LF>
0212 1518      movw (sp)+, b^outdne(r11) ; because we want to (re-)buffer it
0212 1519      tec$wait_done:            ; restore original output done flag
0212 1520      bisb #1, w^ter_o_force      ; wait for all output to complete
0217 1521      bsbb tec$wait              ; say we're forcing terminal output
0219 1522      clrb w^ter_o_force         ; and go do an initial wait
021D 1523      tstb w^ter_o_pending       ; now back to non-forced output
0221 1524      blss 20$                  ; is there any in-progress output?
0223 1525      pushab b^tec$wait_done     ; nope, go exit
0226 1526      10$: $waitfr_s -          ; yes, check again after waiting
0226 1527      efn=#1                    ; wait for single event flag
0226 1528      PUSHL #1                  ; on this event flag
0228 1529      CALLS #1,G^SYSS$WAITFR
022F 1530      bsbw success_or_announce   ; announce any failure...
0232 1531      $clref_s -                ; do an event flag clear
0232 1532      efn=#1                    ; on this event flag
0232 1533      PUSHL #1
0234 1534      CALLS #1,G^SYSS$CLREF
023B 1535      bsbw success_or_announce   ; announce any failure...
023E 1536      tec$wait:                  ; wait for output completion
023E 1537      cmpw ttotptr, #ttobuf     ; is the buffer empty?
0247 1538      beql 20$                  ; yes, go exit
0249 1539      tstw ttoint                ; no, can we initiate output?
024F 1540      bgeq 10$                  ; output's already pending, go wait
0251 1541      bsbw tec$output            ; else initiate whatever output we can
0254 1542      20$: rsb                  ; exit
0255 1543
0255 1544 .disable lsb
```

0000'8F 00FE'CF 03 90 01ED 1505 .sbttl Terminal output waits
00000000'EF B1 01F2 1511 cmpw ttotptr, #ttobuf
15 12 01FB 1512 bneq tec\$wait_done
00F0'CF D5 01FD 1513 tstl w^ter_o_cc
OF 13 0201 1514 beql tec\$wait_done
7E 00'AB B0 0203 1515 movw b^outdne(r11), -(sp)
00F0'CF D4 0207 1516 clrl w^ter_o_cc
047F 30 020B 1517 bsbw echo_tf
00'AB 8E B0 020E 1518 movw (sp)+, b^outdne(r11)
00FE'CF 01 88 0212 1519 tec\$wait_done:
25 10 0212 1520 bisb #1, w^ter_o_force
00FE'CF 94 0217 1521 bsbb tec\$wait
00FF'CF 95 0219 1522 clrb w^ter_o_force
31 19 021D 1523 tstb w^ter_o_pending
EC AF 9F 0221 1524 blss 20\$
0223 1525 pushab b^tec\$wait_done
0226 1526 10\$: \$waitfr_s -
0226 1527 efn=#1
01 DD 0226 1528 PUSHL #1
00000000'GF 01 FB 0228 1529 CALLS #1,G^SYSS\$WAITFR
FE90 30 022F 1530 bsbw success_or_announce
0232 1531 \$clref_s -
0232 1532 efn=#1
01 DD 0232 1533 PUSHL #1
00000000'GF 01 FB 0234 1534 CALLS #1,G^SYSS\$CLREF
FE84 30 023B 1535 bsbw success_or_announce
023E 1536 tec\$wait:
0000'8F 00000000'EF B1 023E 1537 cmpw ttotptr, #ttobuf
0B 13 0247 1538 beql 20\$
00000000'EF B5 0249 1539 tstw ttoint
D5 18 024F 1540 bgeq 10\$
011D 30 0251 1541 bsbw tec\$output
05 0254 1542 20\$: rsb
0255 1543
0255 1544 .disable lsb

```
0255 1542 .sbttl Terminal output
0255 1543
0255 1544 check_esc csi: ; check ESCape/CSI sequences
2A 00FE'CF E8 0255 1545 btlbs 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$ ; yes
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$: movb (r1)+, (r5)+ ; copy over the ESCape/CSI character
50 50 D7 0287 1562 decl r0 ; and remove it from the count
0B 13 0289 1563 beql 50$ ; 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 BF 1D 0294 1565 bvs check_esc_csi ; another ESCape/CSI, go process it
05 0296 1566 50$: rsb ; exit
53 D6 0297 1567 60$: incl r3 ; advance the remaining pointer
52 D7 0299 1568 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$, (sp) ; no more, change return to finish up
0000'8F 50 B1 02A1 1572 cmpw r0, #ttobfl ; about to re-buffer the whole buffer?
15 1E 02A6 1573 bgequ 70$ ; yes, 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), ttotptr ; setting a new buffering pointer
55 8E D0 02BA 1577 ; How about CTRL/D and/or Binary Output?
05 02BD 1578 movl (sp)+, r5 ; restore destination pointer
02BE 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^ctrl_o_flag ; say control/o is now in effect
02CD 1587 $cancel_s - ; cancel all pending I/O
02CD 1588 chan=w^ter_o_chan ; on the terminal output channel
MOVZWL w^ter_o_chan, -(SP)
CALLS #1, G^SYSS$CANCEL
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$w_rsz ; yes, anything in the record yet?
      02 13 02FA 1602 beql 30$ ; 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$: movb #<1a7>!13, w^output_sys_vfc+1 ; indicate a <CR> as postfix
      65'AF 9F 0310 1610 pushab b^90$ ; 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$w_rsz ; always reset the record length
      00F2'CF B4 0330 1616 clrw w^output_sys_vfc ; and the print control information
      FD70 31 0334 1617 brw success_else_die ; check for success completion; exit
      0337 1618
      52 0C BB 0337 1619 60$: pushr #^m<r2,r3> ; save r2 & r3
      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 91 0343 1622 70$: cmpb (r3)+, #13 ; is this a <CR>?
      C2 13 0346 1623 beql 40$ ; yes, go set postfixing
      0A 73 91 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$w_rsz, #output_sys_siz ; room left?
      02 1F 0354 1627 blssu 80$ ; yes
      BB 10 0356 1628 bsbb 50$ ; nope, must dump this buffer
08C4'DF 83 90 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$: decw ttoint ; take away the buffer interlock
      05 0370 1635 rsb ; exit
      0371 1636
      00'AB 02 A8 0371 1637 tec$output: ; terminal output initiation
      0371 1638 bisw #2, b^outdne(r11) ; say some output was done
      0375 1639 tec$output_more: ; more terminal output initiation
      0375 1640 incw ttoint ; interlock the terminal output buffer
51 50 00000000'EF 00' A1 037B 1641 addw3 s^#io$ writevblk, ttomod, r0 ; form function code
      00000000'EF 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 9E 0395 1644 movab ttobuf, ttoptr ; and empty terminal output buffer
      04 50 00' E1 03A0 1645 bbc s^#io$v_canceltrlo, r0, 110$ ; canceling control/o?
      00FD'CF 94 03A4 1646 clrb w^ctrl_o_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^ctrl_o_flag      ; yes, 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?
OA 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
OD BB 03D2 1660 120$:      pushr   #^m<r0,r2,r3> ; save function, iosb, and buffer
1B 00000000'EF 51 2F 03D4 1661      movtuc  r1, ttobuf, #27, w^ter_o_table, r1, (r3) ; move data to buffer
63 51 0000'CF 03 1C 03E1 1662      bvc     130$      ; nothing special...
FE6F 30 03E3 1663      bsbw    check_esc_csi ; else go check ESCape/CSI sequences
OD 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?
OA 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 ; yes, remove the <LF> from the count
6E 008A0000 8F D0 040B 1675      movl    #<<1a7>!10>a16, (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$output_ast
62 7F 0424      PUSHAQ   (r2)
7E 50 3C 0426      MOVZWL   r0, -(SP)
7E 00F6'CF 3C 0429      MOVZWL   w^ter_o_chan, -(SP)
01 DD 042E      PUSHL   #1
00000000'GF 0C FB 0430      CALLS   #12, G^SYSSQIO
00F0'CF 8E D0 0437 1686      (sp)+, w^ter_o_cc ; set carriage control for next time
FC68 30 043C 1687      bsbw    success_else_die ; check for success completion
3C BA 043F 1688 160$:      popr    #^m<r2,r3,r4,r5> ; restore r2 through r5
00000000'EF B7 0441 1689      decw    ttoint ; take away the buffer interlock
05 0447 1690      rsb ; exit
0448 1691
0448 1692 .disable lsb
```

```
0448 1694 .sbtll Terminal input
0448 1695
0448 1696 .enable lsb
0448 1697
00F4'CF B5 0448 1698 10$: tstw w^ter_i_chan ; have a real terminal?
77 12 044C 1699 bneq 110$ ; yes
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$ ; yes
FC73 30 0461 1705 bsbw success_or_abrt ; else check for success completion
0F 58 00' E1 0464 1706 bbc s^#io$_cvflow, r8, 20$ ; converting lower case?
61 8F 51 91 0468 1707 cmpb r1, #^a7A/+32 ; yes, 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$: movzbl r1, i_r0(r10) ; copy character to here
00EC'CF D4 047A 1713 clrl w^ctrlz_cnt ; and clear control/z counter
OC 58 00' E0 047E 1714 30$: bbs s^#io$_noecho, r8, 40$ ; 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$ ; yes, 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$: movaw textit, i_pc(r10) ; set for exiting from teco
17 11 04A2 1724 brb 90$ ; and go exit
04A4 1725
50 04 A6 66 A9 04A4 1726 70$: bisw3 (r6), 4(r6), r0 ; any character(s) or terminator(s)?
73 12 04A9 1727 bneq 130$ ; yes, 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$: movaw teco, i_pc(r10) ; set for restarting teco
0020'CF D4 04B8 1733 90$: 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$: movl w^ter_i_nor_trm_ptr, r1 ; guess at the normal terminator mask
05 58 00' E1 04CA 1739 bbc s^#io$_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$: $qio_w_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
51 DD 04D5 CLRL -(SP)
PUSHL r1
```

00	DD	04D7		PUSHL	#0	
59	DD	04D9		PUSHL	r9	
00 B7	DF	04DB		PUSHAL	a(r7)	
7E	7C	04DE		CLRQ	-(SP)	
66	7F	04E0		PUSHAQ	(r6)	
7E 58	3C	04E2		MOVZWL	r8, -(SP)	
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	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	cmpw	r0, #ss\$_controlc	; was control/c typed?
B9	12	0505	1754	bneq	100\$	; none of the above...
00FC'CF	00FC'CF	92	0507	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	bbc	#tec\$v_et\$cc, i_r0(r10), 80\$	; allowing return of control/c?
66	01	B0	0514	movw	#1, (r6)	; set data count = 1
02 A6	D4	0517	1759	clrl	2(r6)	; and no terminator(s)
00 B7	03	90	051A	movb	#^a/C/-64, a(r7)	; then buffer a control/c
77	86	3C	051E	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	bbs	s^#io\$v_noecho, r8, 140\$	; did they echo?
00'AB	02	A8	052B	bisw	#2, b^outdne(r11)	; echoed, so say so
50	02	A6	052F	movzwl	2(r6), r0	; get the terminator's length
51	67	87	C1	beql	180\$	; no length, so no terminator(s)
56	61	9A	0539	addl3	(r7)+, (r7), r1	; else point just beyond the data
77	50	C0	053C	movzbl	(r1), r6	; get the terminator itself
1A	56	91	053F	addl	r0, -(r7)	; now count the terminator(s)
09	12	0542	1771	cmpb	r6, #^a/Z/-64	; control/z?
07 00EC'CF	03	F2	0544	bneq	150\$	; no
FF4D	31	054A	1773	aoblss	#3, w^ctrlz_cnt, 160\$	; continue if not third control/z
		054D	1774	brw	60\$	; else go exit from teco
00EC'CF	D4	054D	1776	clrl	w^ctrlz_cnt	; no, reset control/z counter
0D	56	91	0551	cmpb	r6, #13	; carriage return?
13	12	0554	1777	bneq	170\$	; no
67	D6	0556	1779	incl	(r7)	; yes, count 1 more in buffer
57 67	87	C1	0558	addl3	(r7)+, (r7), r7	; find position for line feed
77	0A	90	055C	movb	#10, -(r7)	; and store it also
23 58	00	E0	055F	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	movzbl	#10, r6	; and set to echo a line feed
19 58	00	E0	0569	bbs	s^#io\$v_noecho, r8, 180\$	; skip echo if not echoing
7F 8F	56	91	056D	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	bsbw	tec\$input:	; terminal input
00 00000000'EF	00	E3	0576	bs	s^#io\$v_cancel, ctloig	; do a control/o cancel
		057E	1790	bbcs	s^#io\$v_cancel, ctloig, 30005\$	
04 6A	01	E5	057E	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	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$: movzbl    i_r0(r10), r0      ; get character about to be returned
        FA66' 30 0597 1799      bsbw      tecoxelbr      ; and go check it
        0B 13 059A 1800      beql      200$      ; 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$: clrb      w^ctrlc_flag      ; turn off control/; exit flag if here
        05 05AB 1804      rsb          ; and exit
        05AC 1805
        6A 1B 9A 05AC 1806 210$: movzbl    #27, i_r0(r10)      ; set final initial command <ESC>
        F6 11 05AF 1807      brb      200$      ; and go exit with it...
        05B1 1808
20 AA 010C'CF 51 D0 05B1 1809 220$: movl      r1, w^err_msgvec+8 ; save any STV value
        00000000'EF 3E 05B6 1810      movaw    ioerrs, i_pc(r10) ; set exit to teco's error processor
        10 AA D4 05BE 1811      clrl      i_r4(r10)      ; with R4 = 0 for filename exists
        FB3A 31 05C1 1812      brw      success_or_err      ; now go die with the error...
        05C4 1813
        56 00'AB 3C 05C4 1814 230$: 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)?
        DB 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$8bf, b^etype(r11), 250$ ; 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$: movab    w^ter_i_buf, -(r7) ; reset terminal input buffer pointer
        56 0018'CF 7E 05F2 1826      movaq    w^ter_i_status, r6 ; get pointer to terminal input iosb
        3C 05F7 1827      movzwl    #ios_readvblk! - ; set function as read virtual,
        05F8 1828      iosm_nofiltr! - ; no filtering,
        05F8 1829      iosm_trmnoecho! - ; no terminator echo,
        05F8 1830      iosm_dsablmbx, r8 ; keep any mailbox off
        58 0000'8F 3C 05FC 1831      movzwl    #ter_i_siz-1, r9 ; set terminal input buffer size
        59 01FF 8F 3C 0601 1832      bbs      #tec$V_et$lc, b^etype(r11), 260$ ; allowing lower case?
        04 00'AB 02 E0 0606 1833      bs      s^#iosv_cvtlow, r8 ; no, so don't
        00 58 00' E3 0606      bbcs      s^#iosv_cvtlow, r8, 30006$
        060A 1834 30006$:
        04 6A 03 E1 060A 1834 260$: bbc      #tec$V_et$snch, i_r0(r10), 270$ ; echoing?
        060E 1835      bs      s^#iosv_noecho, r8 ; no, so don't
        00 58 00' E3 060E      bbcs      s^#iosv_noecho, r8, 30007$
        0612 1836 30007$:
        04 6A 05 E1 0612 1836 270$: bbc      #tec$V_et$cke, i_r0(r10), 280$ ; checking type ahead?
        0616 1837      bs      s^#iosv_timed, r8 ; yes, so do
        00 58 00' E3 0616      bbcs      s^#iosv_timed, r8, 30008$
        061A 1838 30008$:
        03 6A E9 061A 1838 280$: blbc      i_r0(r10), 290$ ; single character input mode?
        59 01 D0 061D 1839      movl      #T, r9 ; yes, so buffer size is 1 character
        FE25 31 0620 1840 290$: brw      10$ ; long branch for real terminal input
        0623 1841
        57 0378'CF 77 D4 0623 1842 300$: clrl      -(r7) ; ensure the input buffer is empty
        58 50 A7 D0 0625 1843      movaw    w^indir_cmd fab, r7 ; address the "ei" fab
        062A 1844      movl      fab$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
56 66 D0 0632 1846 movl
OC 13 0635 1847 beql
0627 30 0637 1848 bsbw
00000000'8F 50 D1 063A 1849 cmpl
OA 13 0641 1850 beql
0643 1851 310$: bc
50 A7 20 8A 0643 bicb
071F 30 0647 1852 bsbw
00'AB B6 064A 1853 incw
53 A7 51 90 064D 1854 320$: movb
03 58 05 E0 0651 1855 bbs
FF2E 31 0655 1856 brw
0658 1857
6A 58 08 18 EF 0658 1858 330$: extzv
FF34 31 065D 1859 brw
0660 1860
0660 1861 .disable lsb
```

```
s^#1a fab$ v_tecnxt, fab$l_tecsts(r7)
(r6), r6 ; get "ei" fab pointer
310$ ; none?
getbyt ; get the next character
r0, #rms$_normal ; normal completion?
320$ ; yep, character was "pre-fetched"
#fab$ v_tecnxt, fab$l_tecsts(r7) ; say nothing "pre-fetched"
s^#1a fab$ v_tecnxt, fab$l_tecsts(r7)
reset indir ; close the indirect file
b^indir(r11) ; then mark as "funny"(1)
r1, fab$l_tecsts+3(r7) ; save (possible) "pre-fetch" character
#fab$ v_tecnxt, r8, 330$ ; originally have "pre-fetched" data?
180$ ; nope, go loop to try again...

#24, #8, r8, i_r0(r10) ; set the "pre-fetched" character
190$ ; and go check it out...
```

```
0660 1863 .sbttl Echoing, etc.
0660 1864
OFFC 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$: bsbb echo_crlf ; now go do the <CR><LF>
50 01 D0 067C 1875 movl #1, r0 ; set the success return status
00FD'CF 95 067F 1876 tstb w^ctrl_o_flag ; is control/o in effect?
02 13 0683 1877 beql 30$ ; nope
50 D4 0685 1878 clrl r0 ; yep, change return status to failure
04 0687 1879 30$: ret ; return
0688 1880
56 0D 9A 0688 1881 echo_crlf: ; buffer and dump a <CR><LF>
07 10 0688 1882 movzbl #13, r6 ; set a <CR>
068D 1883 bsbb echo_buffer ; and go output it
56 0A 9A 068D 1884 echo_lf: ; buffer and dump a <LF>
0690 1885 movzbl #10, r6 ; set a <LF>
07F0'CF 9F 0690 1886 echo_char: ; buffer and dump a character
0694 1887 pushab w^echo_dump ; dump buffer after buffering
25 56 07 E0 0694 1888 echo_buffer: ; buffer a echo character
20 56 91 0698 1889 bbs #7, r6, 20$ ; a 'negative' character?
5F 1E 069B 1890 cmpb r6, #32 ; a control character?
0D 56 91 069D 1891 bgequ echo_byte ; nope
52 1A 06A0 1892 cmpb r6, #13 ; carriage return or higher
07 56 91 06A2 1893 bgtru 40$ ; it's higher
55 1A 06A5 1894 cmpb r6, #7 ; bell or lower?
02 1F 06A7 1895 bgtru echo_byte ; it's bs, tab, lf, vt, ff, or cr
51 10 06A9 1896 blssu 10$ ; lower than bell
56 DD 06AB 1897 bsbb echo_byte ; it's bell, ring the bell first
56 5E 8F 9A 06AD 1898 10$: pushl r6 ; save character
49 10 06B1 1899 movzbl #^a/^/, r6 ; prefix with an '^'
56 8E 00000040 8F C9 06B3 1900 bsbb echo_byte ; go output the '^'
3F 11 06B8 1901 bisl3 #64, -(sp)+, r6 ; restore character making it visible
06BD 1902 brb echo_byte ; and go output it
06BD 1903
50 51 5B5D 8F 3C 06BD 1904 20$: movzwl #<^a/[/@8>+^a/]/, r1 ; guess at hex digits trailing/leading
FFFFF00'EF46 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$vet$8bt, b^etype(r1), echo_byte ; 8-bit terminal?
51 3C3E 8F 3C 06D1 1908 movzwl #<^a/<7@8>+^a/>/, r1 ; set compose sequence trailing/leading
7E 50 7D 06D6 1909 30$: 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$: cmpb r6, #27 ; escape?
      B2 12 12 06F7 1921 bneq 10$ ; nope, use an uparrow
      56 24 9A 06F9 1922 movzbl #^a/$/, r6 ; yep, use '$'
      0000'8F 00000000'EF B1 06FC 1923 echo_byte: ; buffer a echo byte
      00000000'EF 00000000'EF B1 06FC 1924 cmpw ttotptr, #ttobuf ; is the terminal output buffer empty?
      00000000'EF 00000000'EF B1 0705 1925 beql 30$ ; yes, always o.k. to buffer, etc.
      00000000'EF 0000'8F B1 0707 1926 cmpw ctlofg, ttomod ; mode correct?
      0000'8F 00000000'EF B1 0712 1927 beql 10$ ; it's totally correct
      00000000'EF 0000'8F B1 0714 1928 cmpw #io$m_cancello, ttomod ; mode correct (control/o cancel)?
      0000'8F 00000000'EF B1 071D 1929 bneq 20$ ; nope, must be really changing modes
      00000000'EF 00000000'EF B1 071F 1930 10$: cmpw ttotptr, #ttobuf+ttobfl ; any room to buffer a character?
      00000000'EF 00000000'EF B1 0728 1931 blssu 30$ ; room left, just go buffer character
      00000000'EF 00000000'EF B5 072A 1932 20$: tstw ttoint ; is output currently in progress?
      00000000'EF 00000000'EF B5 0730 1933 bgeq 40$ ; yes, we must wait for it to finish
      00000000'EF 00000000'EF B5 0732 1934 bsbw tec$output ; no, start it going...
      00000000'EF 00000000'EF B6 0735 1935 30$: 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 00000000'EF B4 0746 1937 clrw ctlofg ; clear out control/o cancel request
      00'AB 10 8A 074C 1938 bc #tec$V_et$cco, b^etype(r11) ; and say we did it
      00000000'FF 56 90 0750 1939 bicb s^#1attec$V_et$cco, b^etype(r11)
      00000000'EF B6 0757 1940 movb r6, @ttotptr ; store the character in the buffer
      00000000'EF B7 075D 1941 incw ttotptr ; then bump the buffer pointer
      05 0763 1942 decw ttoint ; and take away the buffer interlock
      0764 1943 rsb ; exit
      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$deln: ; control/u handler
1D 00'AB 01 E0 0769 1952 bbs #tec$vet$scr, b^etype(r11), 40$ ; 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 bsbw 70$ ; and go do it
57 00000000'EF 3C 0794 1963 movzwl crtcup, r7 ; get cursor up sequence base
40 10 079B 1964 bsbw 70$ ; and go do that also
CF 11 079D 1965 brb 10$ ; now go do a <cr><lf>
079F 1966
079F 1967 tec$delch: ; delete handler
D0 00'AB 01 E1 07A2 1968 movzbl i_r0(r10), r6 ; get the deleted character
58 01 D0 07A7 1969 bbc #tec$vet$scr, b^etype(r11), 20$ ; scope mode?
59 56 07 07AA 1970 movl #1, r8 ; set count for 1 erase sequence
20 56 91 07AE 1971 bbs #7, r6, 100$ ; a 'negative' character?
65 1E 07B1 1972 cmpb r6, #32 ; normal graphic?
1B 56 91 07B3 1973 bgequ 120$ ; yes, 1 sequence
60 13 07B6 1974 cmpb r6, #27 ; escape?
58 D6 07B8 1975 beql 120$ ; yes, 1 sequence
0D 56 91 07BA 1976 incl r8 ; no, set count for 2 sequences
59 1A 07BD 1977 cmpb r6, #13 ; is it carriage return or higher?
B9 13 07BF 1978 bgtru 120$ ; higher, do 2 sequences
07 56 91 07C1 1979 beql 30$ ; it's carriage return, re-print line
52 1B 07C4 1980 cmpb r6, #7 ; is it bell or lower?
0A 56 91 07C6 1981 blequ 120$ ; that it is, do 2 sequences
AF 1F 07C9 1982 cmpb r6, #10 ; check around line feed
09 13 07CB 1983 blssu 30$ ; it's backspace or tab, re-print line
0C 56 91 07CD 1984 beql 60$ ; it's line feed, do 1 cursor up
02 1F 07D0 1985 cmpb r6, #12 ; check around form feed
00 10 07D2 1986 blssu 50$ ; it's vertical tab, do 2 cursor ups
00 10 07D4 1987 bsbw 50$ ; do 2 cursor ups, fall to do 2 more
57 00000000'EF 3C 07D6 1988 50$: bsbw 60$ ; do 1 cursor up, fall to do another
57 00'AB A0 07DD 1989 60$: movzwl crtcup, r7 ; get cursor up sequence base
57 67 3C 07E1 1990 70$: addw b^crtype(r11), r7 ; find correct sequence pointer
58 87 9A 07E4 1991 movzwl (r7), r7 ; and get pointer to count, string
56 87 9A 07E7 1992 movzbl (r7)+, r8 ; pickup the count, correct pointer
FF0F 30 07EA 1993 80$: movzbl (r7)+, r6 ; get a character
F7 58 F5 07ED 1994 bsbw echo_byte ; and output it
07F0 1995 sobgtr r8, 80$ ; then loop for more...
00000000'EF B5 07F0 1996 echo_dump: ; dump the buffered echoing
OE 18 07F6 1997 tstw ttoint ; is terminal output buffer free?
0000'8F 00000000'EF B1 07F8 1998 bgeq 90$ ; nope, go exit
03 13 0801 1999 cmpw ttoptr, #ttobuf ; yep, anything buffered therein?
FB6B 30 0803 2000 beql 90$ ; nothing there, go exit
05 0806 2001 bsbw tec$output ; else start that output going
0807 2002 rsb ; exit
0807 2003
```

```
FFFFF00'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$y_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      $qio_w_s -      ; queue an i/o request with wait
               081B 2010      chan = w^ter_o_chan, -      ; on the terminal write channel
               081B 2011      func = s^#io$_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$_writevblk, -(SP)
               7E 00F6'CF 3C 082C      MOVZWL w^ter_o_chan, -(SP)
               00 DD 0831      PUSHL  #0
00000000'GF    0C FB 0833      CALLS  #12, G^SYSS$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 .sbtll Page backwards
0870 2034
0870 2035 .enable lsb
0870 2036
0870 2037 10$: err NFI, <'No file for input'>
      bsbw err
      .long $$$$$$
      .word $$$$$$
      .ascic 'No file for input'

F8D2 30 0870
00000351' 0873
5879 0351
72 6F 66 20 65 6C 69 66 20 6F 4E 00' 0353
74 75 70 6E 69 20 035F
11 0353
0877 2038
0877 2039 20$: err NFO, <'No file for output'>
      bsbw err
      .long $$$$$$
      .word $$$$$$
      .ascic 'No file for output'

F8CB 30 0877
00000365' 087A
587F 0365
72 6F 66 20 65 6C 69 66 20 6F 4E 00' 0367
74 75 70 74 75 6F 20 0373
12 0367
087E 2040
087E 2041 tec$backup:
      movzwl b^inpnt(r11), r6 ; page backwards
      movl (r6), r6 ; get pointer to input file pointer
      beql 10$ ; then get input file fab pointer
      pushl r6 ; no file...
      movzwl b^oupnt(r11), r6 ; save input file fab pointer
      movl (r6), r6 ; get pointer to output file pointer
      beql 20$ ; then get output file fab pointer
      pushaq @fab$q_tecque+4(r6) ; no file...
      pushl r6 ; save current last buffer pointer
      bsbw tec$putuf ; save output file fab pointer
      movl (sp)+, r6 ; now go dump the text buffer
      bbs #fab$v_tecbuf, fab$l_tecsts(r6), 30$ ; already into buffering?
      $rewind - ; rewind the output file
      rab=@fab$l_tecrab(r6) ; so we can re-write it later
      PUSHAL @fab$l_tecrab(r6)
      CALLS #$$,TMP1,G^SYSS$REWIND
      bsbw success or err ; check for success completion
      bs #fab$v_tecbuf, fab$l_tecsts(r6) ; indicate data is buffered
      bisb s^#1@fab$v_tecbuf, fab$l_tecsts(r6)
      mnegw i_r4(r10), r0 ; get number of pages to back up
      movq (sp)+, r1 ; get start pointer & input fab pointer
      movaq fab$q_tecque(r6), r3 ; address the output queue's root
      cmpl r1, r3 ; have we backed up to the top?
      beql 50$ ; yep, must exit the loop early
      movl 4(r1), r1 ; else link backwards to the next
      decw r0 ; should we look at it?
      bgtr 40$ ; yes, so loop...
      mnegw r0, i_r4(r10) ; return any remaining count
      cmpl r1, fab$q_tecque+4(r6) ; have we finished pruning?
      beql 70$ ; yes
      remque @fab$q_tecque+4(r6), r0 ; no, remove a buffer from output tail
      insque (r0), fab$q_tecque(r2) ; and add it to input queue beginning
      bs #fab$v_tecbuf, fab$l_tecsts(r2) ; say using buffered data now
      bisb s^#1@fab$v_tecbuf, fab$l_tecsts(r2)
      brb 60$ ; loop for the next buffer...

56 00'AB 3C 087E 2042
56 66 D0 0882 2043
E9 13 0885 2044
56 DD 0887 2045
56 00'AB 3C 0889 2046
56 66 D0 088D 2047
E5 13 0890 2048
5C B6 7F 0892 2049
56 DD 0895 2050
01AD 30 0897 2051
56 8E D0 089A 2052
11 50 A6 02 E0 089D 2053
08A2 2054
08A2 2055
54 B6 DF 08A2
00000000'GF 01 FB 08A5
F84F 30 08AC 2056
08AF 2057
50 A6 04 88 08AF
50 10 AA AE 08B3 2058 30$:
51 8E 7D 08B7 2059
53 58 A6 7E 08BA 2060
53 51 D1 08BE 2061 40$:
51 04 A1 D0 08C1 2062
50 B7 08C3 2063
F3 14 08C7 2064
10 AA 50 AE 08CB 2065 50$:
5C A6 51 D1 08CF 2066 60$:
OE 13 08D3 2068
50 5C B6 OF 08D5 2069
58 A2 60 OE 08D9 2070
50 A2 04 88 08DD 2071
EC 11 08E1 2072
08E3 2073
```

TECONAT
V39.02

VAX-11 TECO
Page backwards

I 15

16-SEP-1984 02:11:05 VAX/VMS Macro V04-00
10-SEP-1984 13:16:05 [TECO.SRC]TECONAT.MAR;3

Page 62
(26)

05 08E3 2074 70\$: rsb
08E4 2075
08E4 2076 .disable lsb

; erit

```
08E4 2078 .sbttl Get input
08E4 2079
08E4 2080 .enable lsb
08E4 2081
08E4 2082 10$: err NFI, <'No file for input'>
          bsbw err
          .long $$$$$$
          .word $$$$$$
          .ascii 'No file for input'

F85E 30 08E4
0000037A' 08E7
5879 037A
72 6F 66 20 65 6C 69 66 20 6F 4E 00' 037C
74 75 70 6E 69 20 0388
11 037C
08EB 2083
08EB 2084 20$: incl fab$l_tecsts(r6) ; set internal end-of-file indicator
08EE 2085 .assume fab$m_tec eof eq 1
08EE 2086 $close - ; close
08EE 2087 fab=(r6) ; the file
08EE PUSHAL (r6)
08F0 CALLS $$$TMP1,G^SYSS$CLOSE
08F7 2088 movl fab$l_stv(r6), w^err_msgvec+8 ; save the STV value
08FD 2089 bsbw success_or_err ; check for success completion
0900 2090 30$: mcomw #0, b^eoflag(r11) ; set end-of-file flag
0904 2091 rsb ; and exit
0905 2092
0905 2093 40$: bbc #fab$v_tecb2, fab$l_tecsts(r6), 50$ ; /b2 mode?
090A 2094 bsbw 150$ ; yes, go do trailing trims
090D 2095 50$: mcomw #0, b^ffflag(r11) ; set form feed flag
0911 2096 rsb ; and exit
0912 2097
0912 2098 tec$getbf: ; get input
0912 2099 clrw b^eoflag(r11) ; preset not at end-of-file
0915 2100 clrw b^ffflag(r11) ; preset no form feed at buffer's end
0918 2101 movzwl b^inpnt(r11), r6 ; get pointer to input file pointer
091C 2102 movl (r6), r6 ; then get input file fab pointer
091F 2103 beql 10$ ; no file
0921 2104 movl i_r0(r10), r7 ; get buffer pointer
0924 2105 movl i_r1(r10), r8 ; and room counter
0928 2106 beql 120$ ; no room at all, we'd better exit...
092A 2107 60$: decl r8 ; count down the room counter
092C 2108 70$: bbs #fab$v_tecbuf, fab$l_tecsts(r6), 80$ ; never eof if buffering
0931 2109 blbs fab$l_tecsts(r6), 30$ ; keep saying end-of-file if true
0935 2110 .assume fab$m_tec eof eq 1
0935 2111 80$: bsbw getbyt ; get the next character
0938 2112 cmpl r0, #rms$_eof ; end-of-file?
093F 2113 beql 20$ ; yes, so quit
0941 2114 bsbw success_or_err ; else check for success completion
0944 2115 bbc #fab$v_tecb2, fab$l_tecsts(r6), 90$ ; /b2 mode?
0949 2116 cmpb r1, #13 ; a <cr>?
094C 2117 beql 70$ ; yes, ignore the <cr> (for now...)
094E 2118 90$: cmpb r1, #12 ; it is <ff>?
0951 2119 beql 40$ ; yes, we're done
0953 2120 movb r1, (r7)+ ; nope, store in the text buffer
0956 2121 incw b^zz(r11) ; and count it as stored
0959 2122 cmpb r1, #10 ; <lf>?
095C 2123 bneq 110$ ; nope
095E 2124 bbc #fab$v_tecb2, fab$l_tecsts(r6), 100$ ; /b2 mode?
0963 2125 bsbw 130$ ; yes, go do trailing trims
0965 2126 movw #13+<10a8>, (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$:    rsb                          ; exit
0981 2134
05 57 D7 0981 2135 130$: 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 18 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
099D 2153 .word ^m<r2,r3,r4,r5,r6,r7,r8,r9>
7E 58 A6 03FC 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 a-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
OD9F '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$: ret ; return
09D7 2172
09D7 2173 save_data: ; save the put buffer data
09D7 2174 .word ^m<r2,r3,r4,r5>
7E 7C 09D9 2175 clrq -(sp) ; make room for LIB$GET_VM args
63 54 0C 3A 09DB 2176 10$: locc #12, r4, (r3) ; is there an embedded Zff?
10 13 09DF 2177 beql 20$ ; nope
52 54 50 C3 09E1 2178 subl3 r0, r4, r2 ; yes, 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 ; yes, so count it in the count
52 54 D0 09F7 2187 30$: movl r4, r2 ; now use the remaining count
08 13 09FA 2188 beql 40$ ; 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] ; yes, load in the <ff>
50 01 D0 0A07 2193 40$: movl #1, r0 ; set success
04 0A0A 2194 50$: ret ; exit
0A0B 2195
FB AD 52 14 C1 0A0B 2196 60$: addl3 #8+4+8, r2, -8(fp) ; set hunk size w/ overhead included
FC AD DF 0A10 2197 pushal -4(fp) ; stack address of (returned) address
FB AD DF 0A13 2198 pushal -8(fp) ; stack address of size
00000000 'GF 02 FB 0A16 2199 calls #2, g^lib$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), a^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      .ascii  'No file for output'
          74 75 70 74 75 6F 20 039C
          12 0390
          0A47 2214
          0A47 2215      tec$putbf: ; put output
          53 6A D0 0A47 2216      movl    i_r0(r10), r3 ; get text buffer pointer
          54 04 AA D0 0A4A 2217      movl    i_r1(r10), r4 ; and character count
          59 08 AA D0 0A4E 2218      movl    i_r2(r10), r9 ; put emit <ff> flag here
          56 00'AB 3C 0A52 2219      put_buffer: ; internal put output
          56 66 D0 0A56 2220      movzwl   b^oupntr(r11), r6 ; get pointer to output file pointer
          E5 13 0A59 2221      movl    (r6), r6 ; then get output file fab pointer
          57 54 A6 D0 0A5B 2222      beql    10$ ; no file
          FF73 CF 00 FB 0A5F 2223      movl    fab$l_tecrab(r6), r7 ; file, get the rab pointer
          F697 30 0A64 2224      calls    #0, save_data ; go save the put buffer data
          01 50 A6 02 E1 0A67 2225      bsbw     success_or_err ; check for success completion
          05 0A6C 2226      bbc      #fab$v_tecBuf, fab$l_tecsts(r6), 20$ ; proceed if not buffering
          0A6D 2227      rsb      ; else exit...
          51 53 D0 0A6D 2228
          52 D4 0A70 2229 20$:      movl    r3, r1 ; save starting position
          54 D7 0A72 2230      clrl    r2 ; and reset count
          7E 19 0A74 2231 30$:      decl    r4 ; more to look at?
          52 D6 0A76 2232      blss    130$ ; nope
          0C 83 91 0A78 2233      incl    r2 ; yep, count another character
          0A FF A3 91 0A7B 2234      cmpb    (r3)+, #12 ; <ff> or higher?
          EF 1F 0A81 2235      bgtru    30$ ; higher, keep looking...
          67 12 0A83 2236      cmpb    -1(r3), #10 ; <lf> or lower?
          1E A6 03 93 0A85 2237      blssu   30$ ; lower, keep looking...
          01 52 D1 0A88 2238      bneq    110$ ; higher, it's <vt> or <ff>, do record
          0D FE A3 91 0A90 2239      bitb    #fab$m_cr!fab$m_ftn, fab$b_rat(r6) ; lf/cr and/or ftn cc?
          02 52 D1 0A96 2240      beql    110$ ; no, so no additions, etc., do record
          1B FD A3 91 0A98 2241      cmpl    r2, #1 ; anything before the <lf>?
          52 02 C2 0AA1 2242      bleq    110$ ; nope, go do a record
          43 50 A6 09 E1 0AA4 2243      cmpb    -2(r3), #13 ; yep, it it a <cr>?
          52 D5 0AA9 2244      bneq    110$ ; no, do a record also
          06 13 0AAB 2245      cmpl    r2, #2 ; anything before the <cr><lf>?
          26 FD A3 91 0AAD 2246      bleq    40$ ; nope
          39 13 0AB1 2247      cmpb    -3(r3), #27 ; yep, is it <esc><cr><lf>?
          54 D5 0AB3 2248      beql    110$ ; it is, don't remove <cr><lf>, etc.
          30 63 91 0AB7 2249 40$:      subl    #2, r2 ; take away the <cr><lf>
          52 D5 0AA9 2250      bbc      #fab$v_tecb2, fab$l_tecsts(r6), 110$ ; /b2 mode?
          06 13 0AAB 2251      tstl    r2 ; anything in the record?
          26 FD A3 91 0AAD 2252      beql    50$ ; nope
          39 13 0AB1 2253      cmpb    -3(r3), #^a/&/ ; yep, is it <8><cr><lf>?
          54 D5 0AB3 2254      beql    110$ ; that it is, go output as is
          30 63 91 0AB7 2255 50$:      tstl    r4 ; more to come?
          52 D5 0AA9 2256      bleq    100$ ; nope, but check for <ff> coming
          06 13 0AAB 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
63 7E 73 B0 0AC1 2261 60$: movw -(r3), -(sp) ; save the <cr><lf> from text buffer
2609 8F B0 0AC4 2262 movw #9+<^a/&/ab>, (r3) ; guess at adding <tab><8> to record
07 52 D1 0AC9 2263 cmpl r2, #7 ; already across the first tab stop?
        0D 14 0ACC 2264 bgtr 80$ ; 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$: movb #32, (r3) ; change to adding <space><8>
52 02 C0 0AE1 2271 90$: 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
        59 D5 0AE8 2275 ;
        D5 12 0AEA 2276 100$: tstl r9 ; a <ff> coming?
        22 10 0AEC 2277 bneq 60$ ; yes, go add '&' to record
F60D 30 0AEE 2278 110$: bsbb 150$ ; go put that record
FF79 31 0AF1 2279 120$: bsbw success_or_err ; check for success completion
        59 D5 0AF4 2280 brw 20$ ; then loop...
        10 13 0AF6 2281 ;
        7E 63 90 0AF8 2282 130$: tstl r9 ; add a <ff>?
63 0C 90 0AFB 2283 beql 140$ ; nope
        52 D6 0AFE 2284 movb (r3), -(sp) ; yep, save the next byte
        0E 10 0B00 2285 movb #12, (r3) ; then make it a <ff>
63 8E 90 0B02 2286 incl r2 ; and count that <ff>
F5F6 31 0B05 2287 bsbb 150$ ; go put the final record
        52 D5 0B08 2288 movb (sp)+, (r3) ; restore the next byte
        24 13 0B0A 2289 brw success_or_err ; check for success completion & exit
F5EE CF 9F 0B0C 2290 ;
28 A7 51 D0 0B10 2291 140$: tstl r2 ; any size?
22 A7 52 B0 0B14 2292 beql 170$ ; nope
        58 D4 0B18 2293 pushab w^success_or_err ; check for success completion on exit
        67 DF 0B1A 2294 150$: movl r1, rab$l_rbf(r7) ; set starting address of record
00000000'GF 01 FB 0B1C 2295 movw r2, rab$w_rsz(r7) ; and record's size
010C'CF 0C A7 D0 0B23 2296 clrl r8 ; clear our 'over quota' flag
        04 50 E8 0B29 2297 $put - ; put a record
01 58 00 E3 0B2C 2298 rab=(r7) ; to the file
        05 0B30 2299 (r7)
00000000'8F 0C A7 D1 0B31 2300 CALLS $$$TMP1,G^SYSS$PUT
        F5 12 0B39 2301 movl rab$l_stv(r7), w^err_msgvec+8 ; save the STV value
00 00000000'EF 00' E3 0B3B 2302 blbs r0, 170$ ; all done if no error
        0323'CF 7F 0B43 2303 bbsc #0, r8, 180$ ; really an error if second time
FB14 CF 01 FB 0B47 2304 170$: rsb ; exit
        CC 11 0B4C 2305 cmpl rab$l_stv(r7), #ss$_exdiskquota ; is it the quota error?
        0323'CF 7F 0B43 2306 bneq 170$ ; nope, a real true error
        01 01 FB 0B47 2307 bs s^#io$v_canceltrlo, ctlofg ; do a control/o cancel
        01 01 FB 0B47 2308 bbsc s^#io$v_canceltrlo, ctlofg, 30014$
        01 01 FB 0B47 2309 pushaq w^quota_msg_desc ; set the quota exceeded message
        01 01 FB 0B47 2310 calls #1, w^tec$out_ascid ; and go output it
        01 01 FB 0B47 2311 brb 160$ ; now go try, try again...
        01 01 FB 0B47 2312
```

TECONAT
V39.02

VAX-11 TECO
Put output

0B4E 2311 .disable lsb

B 16

16-SEP-1984 02:11:05 VAX/VMS Macro V04-00
10-SEP-1984 13:16:05 [TECO.SRC]TECONAT.MAR;3

Page 68
(28)

```

      OB4E 2313 .sbttl Get an input byte
      OB4E 2314
      OB4E 2315 .enable lsb
      OB4E 2316
      OB4E 2317 getbyt_emit_ctl:
51  FFFFFFFA0 8F CA OB4E 2318 bicl #^c<<126>!31>, r1 ; emit a control character
      OE 51 06 E5 OB55 2319 bbcc #6, r1, 10$ ; trim to 8-bit flag & control char
      00 51 07 E3 OB59 2320 bs #7, r1 ; use 7-bit set control if <6>=0
      08 11 OB5D 30015$: bbcs #7, r1, 30015$ ; else use 8-bit set control
      08 11 OB5D 2321 brb 10$ ; and go emit it
      51 0D 9A OB5F 2322 getbyt_emit_cr:
      03 11 OB5F 2323 movzbl #13, r1 ; emit a <cr>
      03 11 OB62 2324 brb 10$ ; set the <cr>
      03 11 OB64 2325 ; then go emit it
      51 0A 9A OB64 2326 getbyt_emit_lf:
      60 A6 8E D0 OB64 2327 movzbl #10, r1 ; emit a <lf>
      05 05 OB67 2328 10$: movl (sp)+, fab$l_tecdsp(r6) ; set the <lf>
      05 05 OB6B 2329 rsb ; set the next dispatch address
      05 05 OB6C 2330 ; exit
      1D 1E A6 02 E1 OB6C 2331 20$: bbc #fab$v_prn, fab$b_rat(r6), 40$ ; br if not print file format
      00000000'8F 50 D1 OB71 2332 cmpl r0, #rms$_eof ; end-of-file?
      14 12 OB78 2333 bneq 40$ ; nope, other error
      06 50 A6 03 E0 OB7A 2334 bbs #fab$v_tecicr, fab$l_tecsts(r6), 30$ ; just ignored a <cr>?
      OD 64 A6 91 OB7F 2335 cmpb fab$l_tecctl(r6), #13 ; was last character a <cr>?
      09 12 OB83 2336 bneq 40$ ; not a <cr>, nothing more needed...
      DD 10 OB85 2337 30$: bsbb getbyt_emit_lf ; ignored or last <cr>, go emit a <lf>
50 00000000'8F D0 OB87 2338 movl #rms$_eof, r0 ; now restore the end-of-file code
      8E D5 OB8E 2339 40$: tstl (sp)+ ; pop the return exit
      05 05 OB90 2340 rsb ; and exit with error code
      05 05 OB91 2341
      05 05 OB91 2342 getbyt_first:
      60 A6 FD AF 9E OB91 2343 movab b^getbyt_first, fab$l_tecdsp(r6) ; initial dispatch entry point
      55 54 A6 D0 OB96 2344 movl fab$l_tecrab(r6), r5 ; set new record dispatch
      05 05 OB9A 2345 $get - ; get the rab pointer
      05 05 OB9A 2346 ; get
      05 05 OB9A 2347 rab=(r5) ; the next record
      05 05 OB9A 2348 PUSHAL (r5)
      010C'CF 0C A5 D0 OBA3 2349 CALLS #$$,TMP1,G^SYSS$GET
      00000000'8F 50 D1 OBA9 2350 movl rab$l_stv(r5), w^err_msgvec+8 ; save the STV value
      BA 12 OB80 2351 cmpl r0, #rms$_normal ; normal completion?
      26 1E A6 02 E1 OB82 2352 bneq 20$ ; nope, exit with error code
      66 A6 2C B5 B0 OB87 2353 bbc #fab$v_prn, fab$b_rat(r6), 90$ ; print file format?
      51 66 A6 98 OB8B 2354 movw @rab$l_rhb(r5), fab$l_tecctl+2(r6) ; copy vfc bytes
      15 13 OBC0 2355 cvtbl fab$l_tecctl+2(r6), r1 ; get the 'prefix' byte
      10 19 OBC2 2356 beql 80$ ; none
      05 50 A6 01 E2 OBC4 2357 blss 70$ ; a control
      66 A6 97 OBC9 2358 50$: decb fab$l_tecctl+2(r6) ; first time?
      20 15 OBCC 2359 bleq 100$ ; another 'prefix' <cr><lf> to do?
      8F 10 OBCE 2360 60$: bsbb getbyt_emit_cr ; nope, go do the real data
      92 10 OBDO 2361 bsbb getbyt_emit_lf ; yep, emit a <cr>
      F5 11 OBD2 2362 brb 50$ ; emit a <lf>
      FF77 30 OBD4 2363 70$: bsbw getbyt_emit_ctl ; now loop...
      50 A6 02 88 OBD7 2364 80$: bs #fab$v_tecnolst, fab$l_tecsts(r6) ; emit a control character
      50 A6 02 88 OBD7 2364 bisb s^#fab$v_tecnolst, fab$l_tecsts(r6) ; not first time anymore
```

```

      11      11      OBDB      2365      brb      100$      ; then continue
      67 A6      94      OBDD      2366
1E A6      03      93      OBE0      2367 90$:      clrb      fab$l_tecctl+3(r6)      ; guess at no carriage control
      08      13      OBE4      2368      bitb      #fab$m_cr!fab$m_ftn, fab$b_rat(r6) ; implied lf/cr or ftn ccl?
      67 A6      96      OBE6      2369      beql      100$      ; neither
      22 A5      B5      OBE9      2370      incb      fab$l_tecctl+3(r6)      ; one or the other, do cr/lf at end
      36      13      OBEC      2371      tstw      rab$w_rsz(r5)      ; a null record?
60 A6      F3 AF      9E      OBEE      2372      beql      130$      ; yes, no need to process it...
      54 A6      D0      OBF3      2373 100$:      movab      b^110$, fab$l_tecdsp(r6) ; set dispatch for record data
      22 A5      B7      OBF7      2374 110$:      movl      fab$l_tecrab(r6), r5      ; get the rab pointer
      08      19      OBFA      2375      decw      rab$w_rsz(r5)      ; more in the record?
      51      28 B5      9A      OBFC      2376      blss      120$      ; nope
      28 A5      D6      OC00      2377      movzbl     @rab$l_rbf(r5), r1      ; yes, so get a byte
      05      05      OC03      2378      incl      rab$l_rbf(r5)      ; and bump the record pointer
      05      05      OC03      2379      rsb      ; then exit
      51      67 A6      98      OC04      2380
      25      13      OC08      2381 120$:      cvtbl      fab$l_tecctl+3(r6), r1 ; get the 'postfix' byte
      26      19      OC0A      2382      beql      140$      ; none
      02      E0      OC0C      2383      blss      150$      ; a control
13 1E A6      02      E0      OC0C      2384      bbs      #fab$v_prn, fab$b_rat(r6), 130$ ; print file format?
      51      64 A6      9A      OC11      2385      movzbl     fab$l_tecctl(r6), r1 ; get last record data byte
      1B      51      91      OC15      2386      cmpb      r1, #27      ; was it escape?
      15      13      OC18      2387      beql      140$      ; it was, no additions
      0C      51      91      OC1A      2388      cmpb      r1, #12      ; was it <ff> or higher?
      05      1A      OC1D      2389      bgtru      130$      ; higher, go add a <cr><lf>
      0A      51      91      OC1F      2390      cmpb      r1, #10      ; was it <lf>, <vt>, or <ff>?
      0B      1E      OC22      2391      bgequ      140$      ; one of the above, no additions
      FF38      30      OC24      2392 130$:      bsbw      getbyt_emit_cr      ; emit a <cr>
      FF3A      30      OC27      2393      bsbw      getbyt_emit_lf      ; emit a <lf>
      67 A6      97      OC2A      2394      decb      fab$l_tecctl+3(r6) ; another 'postfix' <lf> to do?
      F5      14      OC2D      2395      bgtr      130$      ; yes, so do another
      FF5F      31      OC2F      2396 140$:      brw      getbyt_first      ; now is the time for a new record...
      FF19      30      OC32      2397
      F8      11      OC35      2398 150$:      bsbw      getbyt_emit_ctl      ; emit a control character
      11      11      OC35      2399      brb      140$      ; then go to the next record
      11      11      OC37      2400
      11      11      OC37      2401 .disable lsb
      11      11      OC37      2402
      11      11      OC37      2403 .enable lsb
      11      11      OC37      2404
      50      58 A6      D0      OC37      2405 10$:      movl      fab$q_tecque(r6), r0 ; get queued data buffer pointer
      51      10 B0      9A      OC3B      2406      movzbl     @8+4+4(r0), r1 ; get character from buffer
      10 A0      D6      OC3F      2407      incl      8+4+4(r0) ; bump the buffer pointer
      0C A0      D7      OC42      2408      decl      8+4(r0) ; count down the count
      46      12      OC45      2409      bneq      60$      ; more remains, go exit
      50      58 B6      0F      OC47      2410      remque     @fab$q_tecque(r6), r0 ; no more, remove buffer from queue
      04      12      OC4B      2411      bneq      20$      ; another buffer remains in the queue
      04      12      OC4D      2412      bc      #fab$v_tecbuf, fab$l_tecsts(r6) ; else turn off buffering
      50 A6      04      8A      OC4D      2412      bicb      s^#1a@fab$v_tecbuf, fab$l_tecsts(r6)
      0D9F CF      00      FB      OC51      2413 20$:      calls      #0, w^free_data ; go free up the buffer
      35      11      OC56      2414      brb      60$      ; now go exit
      65 A6      95      OC58      2415
      49      12      OC5B      2416 30$:      tstb      fab$l_tecctl+1(r6) ; have <cr>, already at left margin?
      08      88      OC5D      2417      bneq      90$      ; nope, we need this <cr>
      08      88      OC5D      2418      bs      #fab$v_tecicr, fab$l_tecsts(r6) ; yes, say <cr> ignored...
      50 A6      08      88      OC5D      2418      bisb      s^#1a@fab$v_tecicr, fab$l_tecsts(r6)
      08      88      OC61      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_tecscr, fab\$L_tecsts(r6), 80\$; do <lf> if extra <cr>
60 86 16	16 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
50 A6 08	8A 0C73 2424	bc	#fab\$V_tecicr, fab\$L_tecsts(r6) ; undo <cr> ignored flag
OD 51	91 0C73 2425	bicb	s^#1a@fab\$V_tecicr, fab\$L_tecsts(r6)
09 1A	1A 0C77 2426	cmpb	r1, #13 ; <cr> or greater?
DA 13	13 0C7A 2427	bgtru	40\$; greater, normal & no left margin
0A 51	91 0C7C 2428	beql	30\$; it's <cr>, so left margin
06 1A	1A 0C7E 2429	cmpb	r1, #10 ; is it <lf>, <vt>, or <ff>?
10 13	13 0C81 2430	bgtru	50\$; it's <vt> or <ff>
65 A6 01	90 0C83 2431	beql	70\$; it's <lf>, check for needing <cr>
64 A6 51	90 0C85 2432	movb	#1, fab\$L_tecctl+1(r6) ; set non-zero for not at left margin
50 00000000'8F	D0 0C89 2433	movb	r1, fab\$L_tecctl(r6) ; remember last character returned
	05 0C8D 2434	movl	#rms\$_normal, r0 ; set normal completion
	0C94 2435	rsb	; and exit
64 A6 51	B1 0C95 2436	cmpw	r1, fab\$L_tecctl(r6) ; 2 <lf>'s at left margin?
EE 12	12 0C99 2437	bneq	50\$; nope
51 OD	9A 0C9B 2438	movzbl	#13, r1 ; yep, set the missing <cr>
EA 50 A6 04	E3 0C9E 2439	bbc	#fab\$V_tecscr, fab\$L_tecsts(r6), 60\$; emit the extra <cr>
51 OA	9A 0CA3 2440	movzbl	#10, r1 ; now (re-)set the <lf>
65 A6 94	94 0CA6 2441	clrb	fab\$L_tecctl+1(r6) ; indicate left margin
DE 11	11 0CA9 2442	brb	50\$; and go exit with <cr> or <lf>
	OCAB 2443		
	OCAB 2444		.disable lsb

```
00'AB 0000'8F B0 OCAB 2446 .sbttl Switch to alternate output
OCAB 2447
OCAB 2448 .enable lsb
OCAB 2449
OCAB 2450 tec$outsv: ; switch output file
OCAB 2451 movw #oupalt, b^oupntr(r11) ; do the pointer switch
OCB1 2452 set_outputname: ; set output file name, etc.
57 00'AB 3C OCB1 2453 movzwl b^oupntr(r11), r7 ; get pointer to output file pointer
OD 11 OCB5 2454 brb 10$ ; go set file name, etc.
OCB7 2455
OCB7 2456 .sbttl Switch to alternate input
OCB7 2457
OCB7 2458 tec$inpsv: ; switch input file
00'AB 0000'8F B0 OCB7 2459 movw #inpalt, b^inpntntr(r11) ; do the pointer switch
OCBD 2460 set_inputname: ; set input file name, etc.
00'AB B4 OCB7 2461 clrw b^eoflag(r11) ; guess at not at end-of-file
57 00'AB 3C OCC0 2462 movzwl b^inpntntr(r11), r7 ; get pointer to input file pointer
00000000'EF 94 OCC4 2463 10$: clrb filsr ; guess at file closed (no name)
56 67 D0 OCCA 2464 movl (r7), r6 ; get file's fab pointer
43 13 OCCD 2465 beql 50$ ; file is closed
08 50 A6 02 E0 OCCF 2466 bbs #fab$ tecbuf, fab$l tecsts(r6), 20$ ; never eof if buffering
04 50 A6 E9 OCD4 2467 blbc fab$l tecsts(r6), 20$ ; branch if not at end-of-file
00'AB 00 B2 OCD8 2468 .assume fab$m_tec eof eq 1
OCDC 2469 mcomw #0, b^eoflag(r11) ; eof, indicate such
50 28 A6 D0 OCD8 2470 set_filename: ; set a file name, etc.
51 04 A0 D0 OCE0 2471 20$: movl fab$l_nam(r6), r0 ; get pointer to nam from fab
52 03 A0 9A OCE4 2472 movl nam$l_rsa(r0), r1 ; and pointer to filename
08 13 OCE8 2473 movzbl nam$b_rsl(r0), r2 ; and get filename's length
61 95 OCEA 2474 beql 30$ ; no length?
04 12 OCEC 2475 tstb (r1) ; a starting null?
51 D6 OCEE 2476 bneq 30$ ; nope
52 D7 OCFO 2477 incl r1 ; yes, skip it
FB'8F 52 91 OCF2 2478 decl r2 ; also skip in count
04 1B OCF6 2479 30$: cmpb r2, #filsiz-3-1-1 ; will the whole specification fit?
52 FB'8F 9A OCF8 2480 blequ 40$ ; yes
00000000'EF 61 52 28 OCF8 2481 movzbl #filsiz-3-1-1, r2 ; nope, we must truncate it
63 07 50 A6 09 E1 OD06 2482 40$: movc r2, (r1), filsr ; move in the file specification
0032422F 8F D0 OD08 2483 clrb (r3) ; and make result asciz
05 OD12 2484 bbc #fab$ tecb2, fab$l tecsts(r6), 50$ ; /b2?
OD13 2485 movl #^a"/BZ", (r3) ; yes, add the switch and asciz again
OD13 2486 50$: r,b ; exit
OD13 2487
OD13 2488 .disable lsb
```

```

      OD13 2490 .sbttl Close input & output files
      OD13 2491
      OD13 2492 .enable lsb
      OD13 2493
      OD13 2494 tec$clsfl: ; close input & output files
      3E 10 OD13 2495 bsbb close_input ; close the input file first
      OD15 2496
      OD15 2497 .sbttl Close output file
      OD15 2498
      OD15 2499 tec$clsof: ; close output file
      57 00'AB 3C OD15 2500 movzwl b^oupntr(r11), r7 ; get pointer to output file pointer
      56 67 D0 OD19 2501 10$: movl (r7), r6 ; get file's fab pointer
      47 13 OD1C 2502 beql 40$ ; no file
      0A 04 A6 0F E5 OD1E 2503 bbcc #fab$V_dlt, fab$l_fop(r6), 20$ ; a file, don't delete it
      05 50 A6 02 E5 OD23 2504 bbcc #fab$V_tecbuf, fab$l_tecsts(r6), 20$ ; need data buffer dump?
      FC70 CF 00 FB OD28 2505 calls #0, w^dump_data ; yes, so go do so
      67 D4 OD2D 2506 20$: clrl (r7) ; file is open no longer
      32 50 A6 E8 OD2F 2507 blbs fab$l_tecsts(r6), 40$ ; already closed if at end-of-file
      OD33 2508 .assume fab$m_tecEOF eq 1
      0062 30 OD33 2509 bsbw 70$ ; release any and all data lines...
      OD36 2510 $close - ; close
      OD36 2511 fab=(r6) ; the file
      66 DF OD36 2512 PUSHAL (r6)
      00000000'GF 01 FB OD38 2513 CALLS #$$,TMP1,G^SYSS$CLOSE
      010C'CF 0C A6 D0 OD3F 2512 movl fab$l_stv(r6), w^err_msgvec+8 ; save the STV value
      00000000'8F 50 D1 OD45 2513 cmpl r0, #rms$eof ; did we get end-of-file (why??)
      02 12 OD4C 2514 bneq 30$ ; nope
      50 D6 OD4E 2515 incl r0 ; yep, fudge for success (RMS bug?)
      F3AB 31 OD50 2516 30$: brw success_or_err ; check for success completion & exit
      OD53 2517
      OD53 2518 close_input: ; close input file
      57 00'AB B4 OD53 2519 clrw b^eoflag(r11) ; never eof if input file closed
      00'AB 3C OD56 2520 movzwl b^inpnr(r11), r7 ; get pointer to input file pointer
      BD 11 OD5A 2521 brb 10$ ; then go close it & exit
      OD5C 2522
      OD5C 2523 .sbttl Kill output file
      OD5C 2524
      OD5C 2525 tec$kilfl: ; delete output file
      57 00'AB 3C OD5C 2526 movzwl b^oupntr(r11), r7 ; get pointer to output file pointer
      56 67 D0 OD60 2527 movl (r7), r6 ; then get output file fab pointer
      C8 12 OD63 2528 bneq 20$ ; a file, go close & delete it
      05 OD65 2529 40$: rsb ; else just exit
      OD66 2530
      OD66 2531 .sbttl Close indirect command file
      OD66 2532
      OD66 2533 close_indir: ; close indirect w/ error checking
      E7 AF 9F OD66 2534 pushab b^30$ ; error check upon exit
      OD69 2535 reset_indir: ; close indirect command file
      00'AB B4 OD69 2536 clrw b^indir(r11) ; ensure indirect file looks closed
      50 00000000'8F D0 OD6C 2537 movl #rms$normal, r0 ; pre-set o.k. if file's not open
      56 00000000'EF D0 OD73 2538 movl cmdprm, r6 ; get indirect file fab pointer
      17 13 OD7A 2539 beql 50$ ; no file
      1A 10 OD7C 2540 bsbb 70$ ; release any and all data lines...
      OD7E 2541 $close - ; close
      OD7E 2542 fab=(r6) ; the file
      66 DF OD7E 2543 PUSHAL (r6)
      00000000'GF 01 FB OD80 2544 CALLS #$$,TMP1,G^SYSS$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
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
03 00'AB 02 0496 30 0DB4 2566      bsbw      not_exiting ; not exiting if error
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      tec$exelbr ; and go do it
A2 11 0DC5 2571      brb      reset_indir ; go ensure no indirect and exit
0DC7 2572
0DC7 2573      tmporg      tec$exelbr
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$getfl:
0DC7 2583     bsbw  fetch_filbuf
0DCA 2584     bneq  non_null
0DCC 2585     tstw  i_r2(r10)
0DCF 2586     bgtr  20$
0DD1 2587     beql  10$
0DD3 2588     cmpw  i_r2(r10), #^a/I/-^a/R/
0DD9 2589     beql  cclose_indir
0ddb 2590     blss  non_null
0DDD 2591     brw   en_next
0DE0 2592
00'AB 0000'8F 80 0DE0 2593 10$:  movw  #inpnor, b^inpntr(r11)
      FED4 31 0DE6 2594     brw   set_inputname
00'AB 0000'8F 80 0DE9 2595 20$:  movw  #oupnor, b^oupntr(r11)
      FEBF 31 0DEF 2596     brw   set_outputname
0DF2 2597
0DF2 2598 non_null:
51 08ED'CF 9E 0DF2 2599     movab w^file_spec_buf, r1
      52 51 D0 0DF7 2600     movl  r1, r2
      08EC'CF 94 0DFA 2601     clrb  w^file_spec_len
      08E0'CF D4 0DFE 2602     clrl  w^file_spec_opt
      53 D4 0E02 2603     clrl  r3
      02BB 30 0E04 2604 10$:  bsbw  get_file_char
      48 13 0E07 2605     beql  70$
      05 53 E8 0E09 2606     blbs  r3, 20$
      2F 50 91 0E0C 2607     cmpb  r0, #^a'/'
      09 13 0E0F 2608     beql  30$
      81 50 90 0E11 2609 20$:  movb  r0, (r1)+
      08EC'CF 96 0E14 2610     incb  w^file_spec_len
      EA 11 0E18 2611     brb   10$
      0E1A 2612
51 08E4'CF DE 0E1A 2613 30$:  moval  w^file_spec_swt, r1
      61 D4 0E1F 2614     clrl  (r1)
      029E 30 0E21 2615 40$:  bsbw  get_file_char
      10 13 0E24 2616     beql  50$
      2F 50 91 0E26 2617     cmpb  r0, #^a'/'
      0B 13 0E29 2618     beql  50$
      81 50 90 0E2B 2619     movb  r0, (r1)+
      08E7'CF 95 0E2E 2620     tstb  w^file_spec_swt+3
      ED 13 0E32 2621     beql  40$
      21 11 0E34 2622     brb   80$
      0E36 2623
51 024C'CF 7E 0E36 2624 50$:  movaq  w^switch_list-4, r1
      81 D5 0E3B 2625 60$:  tstl  (r1)+
      61 95 0E3D 2626     tstb  (r1)
      16 13 0E3F 2627     beql  80$
      81 08E4'CF D1 0E41 2628     cmpl  w^file_spec_swt, (r1)+
      F3 12 0E46 2629     bneq  60$
      08E0'CF 61 C8 0E48 2630     bisl  (r1), w^file_spec_opt
      50 95 0E4D 2631     tstb  r0
      C9 12 0E4F 2632     bneq  30$
      08EC'CF 95 0E51 2633 70$:  tstb  w^file_spec_len
      OD 12 0E55 2634     bneq  do_non_null
50 00000000'8F D0 0E57 2635 80$:  movl  #rms$_syn, r0
      2636
```

; get files (EB, EI, EN, ER, EW)
; go fetch the filename buffer
; go off to process non-null...
; null, but what is it for?
; it's EW
; it's ER
; come on now, what is it really?
; it's EI, go close indirect
; it's EB, we'll die in parse...
; it's EN, go get next occurrence
; do the pointer switch
; and go set input file name, etc.
; do the pointer switch
; and go set output file name, etc.
; non-null file specification
; get pointer to filespec buffer
; in two registers
; reset the filespec's length
; and options
; say outside of quotes initially
; get a character
; null, the end
; branch if inside quotes
; else check for a switch
; found one, switches start here...
; (re-)store into filespec's buffer
; and (re-)count it in the length
; then loop for more...
; point to the switch buffer
; and clear it
; get a switch character
; null, end of this switch & spec too
; start of another switch?
; yes, go end this one first
; else store a switch character
; did we store too many?
; nope, continue
; yep, go give an error...
; get the switch list
; skip the bit pattern
; more to check?
; nope, go give an error
; a match?
; nope, keep checking...
; yep, set the correct bit(s)
; is there more to come?
; yes, go get it...
; did we get any length at all?
; yep
; say that's illegal

```

      OE5E 2637
      OE5E 2638 .enable lsb
      OE5E 2639
      F29D 31 OE5E 2640 10$: brw success_or_err ; go die with the error
      OE61 2641
      028B 31 OE61 2642 20$: brw en_preset ; go do preset for 'en'
      OE64 2643
      OE64 2644 do_non_null: ; do real opens, etc.
      08 AA B5 OE64 2645 tstw i r2(r10) ; what is it for?
      OC 14 OE67 2646 bgtr 30$ ; it's EW
      42 13 OE69 2647 beql 70$ ; it's ER
      FFF7 8F 08 AA B1 OE6B 2648 cmpw i r2(r10), #^a/I/-^a/R/ ; come on now, what is it really?
      15 13 OE71 2649 beql 40$ ; it's EI
      EC 14 OE73 2650 bgtr 20$ ; it's EN
      57 00'AB 3C OE75 2651 30$: movzwl b^oupntr(r11), r7 ; EB/EW, point to output file pointer
      56 67 D0 OE79 2652 movl (r7), r6 ; then get output file fab pointer
      27 13 OE7C 2653 beql 60$ ; closed is o.k.
      FE30 30 OE7E 2654 bsbw set_outputname ; already open, set cpen file's name
      OE81 2655 err OF0, <'Output file already open'>
      F2C1 30 OE81
      000003A3' OE84
      SEBF 03A3 .long $$$$$$
      65 6C 69 66 20 74 75 70 74 75 4F 00' 03A5 .word $$$$$$
      65 70 6F 20 79 64 61 65 72 6C 61 20 03B1 .ascii 'Output file already open'
      6E 03BD
      18 03A5
      OE88 2656
      FEDB 30 OE88 2657 40$: bsbw close_indir ; close the current indirect file
      56 0378'CF DE OE8B 2658 movl w^indir_cmd_fab, r6 ; get the indirect command fab
      57 00000000'EF DE OE90 2659 movl cmdprm, r7 ; and where to store fab pointer
      28 11 OE97 2660 brb 80$ ; then join common open code
      OE99 2661
      00'AB B5 OE99 2662 50$: tstw b^nflg(r11) ; are we returning a value?
      C0 18 OE9C 2663 bgeq 10$ ; no, just die with the error
      OE9E 2664 err FNF, <'File not found'>
      F2A4 30 OE9E
      000003BE' OEA1
      27B6 03BE
      6F 66 20 74 6F 6E 20 65 6C 69 46 00' 03C0
      64 6E 75 03CC
      OE 03C0
      OEA5 2665
      08 AA B5 OEA5 2666 60$: tstw i r2(r10) ; is it EB or EW?
      03 15 OEA8 2667 bleq 70$ ; it's EB
      00D8 31 OEAA 2668 brw 170$ ; it's EW
      OEAD 2669
      FEA3 30 OEAD 2670 70$: bsbw close_input ; close the current input file
      56 0110'CF DE OEBO 2671 movl w^input_nor_fab, r6 ; guess at normal input
      0000'8F 57 B1 OEB5 2672 cmpw r7, #inpnr ; good guess?
      05 13 OEBA 2673 beql 80$ ; yep
      56 0484'CF DE OEBC 2674 movl w^input_alt_fab, r6 ; nope, alternate input
      50 A6 08E0'CF D0 OEC1 2675 80$: movl w^file_spec_opt, fab$l_tecsts(r6) ; set file spec options
      60 A6 FCC6 CF 9E OEC7 2676 movab w^getbyt_first, fab$l_tecdsp(r6) ; reset the get byte dispatch
      64 A6 0A 9A OECB 2677 movzbl #10, fab$l_tecctl(r6) ; and the control bytes
      04 A6 00000040 8F D0 OED1 2678 movl #fab$m_sqo, fab$l_fop(r6) ; reset file options
      1E A6 0202 8F B0 OED9 2679 movw #<fab$c_vara8>!fab$m_cr, fab$b_rat(r6) ; reset record fmt/attr
      OEDF 2680 .assume fab$b_rfm eq fab$b_rat+1
```

```

    05 17 A6 02 90 0EDF 2681 movb #fab$m_get, fab$b_shr(r6) ; set sharing to only other gets
    50 50 A6 07 E1 0EE3 2682 bbc #fab$v_tecrw, fab$l_tecsts(r6), 90$ ; /rw?
    00 04 A6 07 E3 0EE8 2683 bs #fab$v_rwo, fab$l_fop(r6) ; set rewind before open
    05 50 A6 08 E1 0EED 2684 30020$: bbc #fab$v_tecsh, fab$l_tecsts(r6), 100$ ; /sh?
    17 A6 45 8F 90 0EF2 2685 90$: movb #fab$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
    00000000'GF 66 DF 0EFD 2687 $open - ; open
    010C'CF 0C A6 01 FB 0EFF 2688 (r6) ; the input file
    03 1F A6 91 0F0F 2689 CALLS #$$,TMP1,G^SYSS$OPEN
    50 00000000'8F 0D 12 0F13 2690 movl fab$l_stv(r6), w^err_msgvec+8 ; save the STV value
    0C 3F A6 91 0F1C 2691 blbc r0, 50$ ; branch if failure of any type
    04 1F A6 91 0F22 2692 cmpb fab$b_rfm(r6), #fab$c_vfc ; vfc record format?
    1E A6 02 88 0F28 2693 bneq 110$ ; nope
    00000000'GF 54 B6 DF 0F2C 2694 movl #rms$fsz, r0 ; pre-set bad vfc size error
    0A 50 A6 06 E1 0F39 2695 cmpb fab$b_fsz(r6), #input_vfc_siz ; vfc size correct?
    1E A6 F7 8F 8A 0F3E 2696 bgtru 130$ ; nope, go die with an error
    58 A6 58 A6 7E 0F48 2697 cmpb fab$b_rfm(r6), #fab$c_stm ; stream record format?
    5C A6 58 A6 7E 0F4D 2698 bneq 120$ ; nope
    FFF7 8F 08 AA B1 0F55 2699 bisb #fab$m_cr, fab$b_rat(r6) ; always say implied lf/cr for stream
    00'AB 57 B0 0F5F 2700 $connect - ; connect
    FD76 30 0F63 2701 rab=a^fab$l_tecrab(r6) ; the correct rab
    0A 50 A6 06 E1 0F39 2702 CALLS #$$,TMP1,G^SYSS$CONNECT
    1E A6 F7 8F 8A 0F3E 2703 bsbw success or cls ; check for success completion
    58 A6 58 A6 7E 0F48 2704 bbc #fab$v_tecfmt, fab$l_tecsts(r6), 140$ ; any format options?
    5C A6 58 A6 7E 0F4D 2705 bicb #^c<fab$m_blk>, fab$b_rat(r6) ; yes, clear all but this
    FFF7 8F 08 AA B1 0F55 2706 bisb fab$l_tecsts+2(r6), fab$b_rat(r6) ; then apply options
    00'AB 57 B0 0F5F 2707 movaq fab$q_tecque(r6), fab$q_tecque(r6) ; initialize the
    FD76 30 0F63 2708 movaq fab$q_tecque(r6), fab$q_tecque+4(r6) ; data buffer queue
    50 28 A6 D0 0F67 2709 movl r6, (r7) ; set file as open
    51 03 A0 9A 0F6B 2710 cmpw i_r2(r10), #^a/I/-^a/R/ ; what is it?
    08EC'CF 51 90 0F6F 2711 blss 160$ ; it's EB
    04 B0 51 2C 0F74 2712 bgtr 150$ ; it's ER
    08ED'CF 0F7C 2713 movw r7, b^indir(r11) ; it's EI, set indirect as active
    00 08E0'CF 0A E3 0F7F 2714 bsbw set_filename ; set the file's name, etc.
    57 00'AB 3C 0F85 2715 rsb ; and exit
    56 0274'CF DE 0F89 2716 movl fab$l_nam(r6), r0 ; get pointer to nam from fab
    0000'8F 57 B1 0F8E 2717 movzbl nam$b_rsl(r0), r1 ; and get resultant filename's length
    56 05E8'CF DE 0F95 2718 movb r1, w^file_spec_len ; set a new file spec length
    04 50 A6 D0 0F9A 2719 movc5 r1, a^nam$l_rsa(r0), #0, - ; move input's resultant filename
    10000044 8F D0 0FA0 2720 #nam$c_maxrss, w^file_spec_buf ; into file spec buffer
    36 A6 B4 0FA8 2721 bs #fab$v_tecnv, w^file_spec_opt ; ensure maximized version
    00 08E0'CF 0A E3 0F7F 2722 bbsc #fab$v_tecnv, w^file_spec_opt, 30021$
    57 00'AB 3C 0F85 2723 30021$: movzwl b^oupntr(r11), r7 ; get place to store fab pointer
    56 0274'CF DE 0F89 2724 movl w^output_nor_fab, r6 ; guess at normal output
    0000'8F 57 B1 0F8E 2725 cmpw r7, #oupnor ; good guess?
    56 05E8'CF DE 0F95 2726 beql 180$ ; nope
    04 50 A6 D0 0F9A 2727 movl w^output_alt_fab, r6 ; nope, alternate output
    10000044 8F D0 0FA0 2728 movl w^file_spec_opt, fab$l_tecsts(r6) ; set file spec options
    36 A6 B4 0FA8 2729 movl #fab$m_sqr!fab$m_sup!fab$m_tef, fab$l_fop(r6) ; reset options
    00 08E0'CF 0A E3 0F7F 2730 clrw fab$m_mrs(r6) ; maximum record size = 0 for variable
```

1F A6 02 90 OFAB 2729	movb	#fab\$var, fab\$b_rfm(r6) ; guess at variable record format
1E A6 02 90 OFAF 2730	movb	#fab\$m_cr, fab\$b_rat(r6) ; guess at implied lf/cr records
24 A6 D4 OFB3 2731	clrl	fab\$l_xab(r6) ; guess at no specific protection code
17 A6 42 8F 90 OFB6 2732	movb	#fab\$m_get!fab\$m_upi, fab\$b_shr(r6) ; set sharing to gets
05 50 A6 07 E1 OFBB 2733	bbc	#fab\$var_tecrw, fab\$l_tecsts(r6), 190\$; /rw?
00 04 A6 07 E3 OFC0 2734	bs	#fab\$var_rwo, fab\$l_fop(r6) ; set rewind before open
	bbcs	#fab\$var_rwo, fab\$l_fop(r6), 30022\$
05 50 A6 08 E1 OFC5 2735	bbc	#fab\$var_tecsh, fab\$l_tecsts(r6), 200\$; /sh?
17 A6 43 8F 90 OFCA 2736	movb	#fab\$m_get!fab\$m_put!fab\$m_upi, fab\$b_shr(r6) ; set sharing
04 50 A6 0A E1 OFCF 2737	bbc	#fab\$var_tecnv, fab\$l_tecsts(r6), 210\$; /nv?
	bs	#fab\$var_mxv, fab\$l_fop(r6) ; set maximized version numbers
04 A6 02 88 OFD4 2738	bisb	s^10fab\$var_mxv, fab\$l_fop(r6)
50 00 AB 3C OFD8 2739	movzwl	b^inpnr(r1), r0 ; get input file's pointer
50 60 D0 OFDC 2740	movl	(r0), r0 ; to get its fab pointer
22 13 OFDF 2741	beql	230\$; no input file
04 1E A0 00 E1 OFE1 2742	bbc	#fab\$var_ftn, fab\$b_rat(r0), 220\$; really want fortran ccl?
1E A6 01 90 OFE6 2743	movb	#fab\$m_ftn, fab\$b_rat(r6) ; yes, so set it
08 AA B5 OFEA 2744	tstw	i_r2(r10) ; is it EB or EW?
14 14 OFED 2745	bgtr	230\$; it's EW
51 24 A0 D0 OFEF 2746	movl	fab\$l_xab(r0), r1 ; it's EB, get input's protection xab
08 A1 00 F0 8F AA OFF3 2747	bicw	#<xab\$m_noread! - ; clear no read
		xab\$m_nowrite! - ; and no write
		xab\$m_noexe! - ; and no execute
		xab\$m_nodel>@xab\$var_own, - ; and no delete @ owner spot
		xab\$var_pro(r1) ; to ensure owner stuff is o.k.
24 A6 51 D0 OFF9 2751	movl	r1, fab\$l_xab(r6) ; now use that protection code
04 1F A0 91 OFFD 2752	cmpb	fab\$b_rfm(r0), #fab\$var_stm ; input in stream record format?
05 13 1001 2753	beql	240\$; yes, go default output to stream
08 50 A6 08 E1 1003 2754	bbc	#fab\$var_tecstm, fab\$l_tecsts(r6), 250\$; stream format option?
1F A6 04 90 1008 2755	movb	#fab\$var_stm, fab\$b_rfm(r6) ; set output for stream format
1E A6 02 90 100C 2756	movb	#fab\$m_cr, fab\$b_rat(r6) ; with implied lf/cr records
04 50 A6 0C E1 1010 2757	bbc	#fab\$var_tecvar, fab\$l_tecsts(r6), 260\$; variable format option?
1F A6 02 90 1015 2758	movb	#fab\$var_var, fab\$b_rfm(r6) ; set output for variable format
05 50 A6 06 E1 1019 2759	bbc	#fab\$var_tecfmt, fab\$l_tecsts(r6), 270\$; any format options?
1E A6 52 A6 90 101E 2760	movb	fab\$l_tecsts+2(r6), fab\$b_rat(r6) ; yes, apply options
34 A6 08 EC CF 90 1023 2761	movb	w^file_spec_len, fab\$b_fns(r6) ; set file spec's length
	\$create	- ; create
		fab=(r6) ; the output file
	PUSHAL	(r6)
	CALLS	\$\$\$.TMP1,G^SYSS\$CREATE
00000000'GF 66 DF 1029 2765	movl	fab\$l_stv(r6), w^err_msgvec+8 ; save the STV value
010C'CF 0C A6 D0 1032 2766	movl	fab\$l_xab(r6), r2 ; save protection xab use indicator
52 24 A6 D0 1038 2767	clrl	fab\$l_xab(r6) ; then remove the protection xab
24 A6 D4 103C 2768	tstl	r2 ; did we use a protection xab?
52 D5 103F 2769	beql	280\$; nope
09 13 1041 2770	cmpl	r0, #rms\$_prv ; yes, a privilege violation error?
00000000'8F 50 D1 1043 2771	beql	270\$; that we did, go try, try again...
D7 13 104A 2772	bs	#fab\$var_dlt, fab\$l_fop(r6) ; (pre-)mark file for deletion
00 04 A6 0F E3 104C 2773	bbcs	#fab\$var_dlt, fab\$l_fop(r6), 30024\$
		30024\$:
00000000'8F 50 D1 1051 2774	cmpl	r0, #rms\$_supersede ; did we supersede something?
03 13 1058 2775	beql	290\$; yes, that's o.k.
FOA1 30 105A 2776	bsbw	success_or_err ; else check for success completion
	\$connect	- ; connect
		rab=@fab\$l_tecrab(r6) ; the correct rab
	PUSHAL	@fab\$l_tecrab(r6)
54 B6 DF 105D 2777		

```
00000000'GF 01 FB 1060 CALLS $$$TMP1,G^SYSS$CONNECT
                    F07A 30 1067 2778 bsbw success_or_cls ; check for success completion
                    58 A6 58 A6 7E 106A 2779 movaq fab$q_tecque(r6), fab$q_tecque(r6) ; initialize the
                    5C A6 58 A6 7E 106F 2780 movaq fab$q_tecque(r6), fab$q_tecque+4(r6) ; data buffer queue
                    67 56 D0 1074 2781 movl r6, (r7) ; set file as open
                    FC37 31 1077 2782 brw set_outputname ; set the file's name, etc. & exit
                    107A 2783
                    107A 2784 .disable lsb
                    107A 2785
                    107A 2786 fetch_filbuf: ; fetch (converted) filename buffer
                    107A 2787 movtuc #filsiz, filsrt, - ; move translated from TECO's buffer
                    1084
                    108D 2788 #0, w^file_spec_table, - ; 'til 0(end), 128(spec), 255(end?)
                    108D 2789 #nam$c_maxrss, w^file_spec_buf ; into our file spec buffer
                    29 1C 108D 2790 30$ ; Terminator not seen??
                    80 8F 81 91 108F 2791 10$: bvc (r1)+, #128 ; is it the special (128)?
                    23 12 1093 2792 cmpb r0, #128 ; nope, other, call it the end
                    50 D7 1095 2793 bneq r0 ; yes, remove the special from count
                    1F 13 1097 2794 beql 30$ ; 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
                    50 82 9A 10C2 2807 get_file_char: ; get file specification character
                    27 13 10C5 2808 10$: movzbl (r2)+, r0 ; get next byte from filename buffer
                    1A 53 E8 10C7 2809 beql 40$ ; null, the end, exit 'beql'
                    20 50 91 10CA 2810 blbs r3, 20$ ; branch if inside of quotes
                    7F 8F 50 91 10CD 2811 cmpb r0, #32 ; is this a junky character?
                    ED 13 10D3 2812 blequ 10$ ; yes, ignore it
                    61 8F 50 91 10CF 2813 cmpb r0, #127 ; other kind of junky character?
                    09 1F 10D9 2814 beql 10$ ; yes, ignore it
                    7A 8F 50 91 10DB 2815 cmpb r0, #^a/A/+32 ; is it lower case?
                    50 03 1A 10DF 2816 blssu 20$ ; not lower case
                    22 50 91 10E1 2817 cmpb r0, #^a/Z/+32 ; might be
                    53 01 CC 10E9 2818 bgtru 20$ ; but it isn't...
                    50 03 12 10E7 2819 bicb #32, r0 ; lower case, make into upper case
                    95 10EC 2820 20$: cmpb r0, #^a/'/ ; changing quote mode?
                    05 10EE 2821 bneq 30$ ; nope
                    2822 xorl #1, r3 ; yep, so flip the flop
                    2823 30$: tstb r0 ; ensure 'bneq' exit
                    2824 40$: rsb ; exit
```

```

10EF 2826 .sbttl Do 'en' processing
10EF 2827
10EF 2828 .enable lsb
10EF 2829
10EF 2830 en_preset: ; preset the 'en' specification
10EF 2831      movl    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      $sparse - ; parse the specification
10FA 2834      fab=(r6) ; from the 'en' fab
10FA 2835      PUSHAL (r6)
00000000'GF 66 DF 10FA 2835      CALLS    $$$T.MP1,G^SYSS$PARSE
010C'CF 0C A6 DO 1103 2835      movl    fab$l_stv(r6), w^err_msgvec+8 ; save the STV value
50 A6 01 DO 1109 2836 10$:      movl    #fab$m_tecEOF, fab$l_tecsts(r6) ; reset private status (fnf)
EFEE 30 110D 2837      bsbw    success_or_err ; check for success completion
50 A6 D4 1110 2838      clrl    fab$l_tecsts(r6) ; made it, mark as active
05 1113 2839      rsb ; then exit
1114 2840
1114 2841 en_next: ; get next 'en' occurrence
1114 2842      movl    w^en fab, r6 ; get the 'en' fab pointer
34 50 A6 E8 1119 2843      blbs    fab$l_tecsts(r6), 50$ ; keep saying 'fnf' if we need to
111D 2844      .assume fab$m_tecEOF eq 1
111D 2845 20$:      $search - ; else search for next occurrence
111D 2846      fab=(r6) ; using the 'en' fab
111D 2847      PUSHAL (r6)
00000000'GF 66 DF 111F 2847      CALLS    $$$T.MP1,G^SYSS$SEARCH
010C'CF 0C A6 DO 1126 2847      movl    fab$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      .ascii    'File not found'
64 6E 75 03DD
OE 03D1
1158 2863
1158 2864 .disable lsb

```

TEC
Sym

INI
INP
INP
INP
INP
INP
INP
INP
INP
INP
INP
INP
INP
INP
INP
INP
INP
INP
INP
INP
INP
IOS
IOS
IOS
IOS
IOS
IOS
IOS
IOS
IOS
IOS
IOS
IOS
IOE
IOE
I-B
I-C
I-P
I-P
I-R
I-R
I-R
I-R
I-R
I-S
JPI
JPI
LIB
LIB
LIB

			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\$p_id, -(r6)	; guess at wanting process id(+2)
	58	05 D0	116C	2874 movl #5, r8	; guess at -lej (5 => vms)
		6A B5	116F	2875 tstw i_r0(r10)	; what does the user really want?
		37 19	1171	2876 blss 20\$	; it is -lej, return 5 for vms
		OF 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 lej, terminal unit
	02	6A B1	117A	2879 cmpw i_r0(r10), #2	; what is it really?
		2B 19	117D	2880 blss 20\$	; it is lej, 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 itm1st=(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^SYSS\$GETJPI	
	EF 3A	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 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

TEC	Sym
RAB	RAB
RAB	RAB
RAB	RAB
RAB	RAB
RAB	RAB
RAB	RAB
RAB	RAB
RAB	RAB
RAB	RAB
RAB	RAB
RAB	RAB
RES	RES
RMS	RMS
RMS	RMS
RMS	RMS
RMS	RMS
RMS	RMS
RMS	RMS
RMS	RMS
RMS	RMS
RMS	RMS
RWS	RWS
SAV	SAV
SAV	SAV
SCH	SCH
SCH	SCH
SET	SET
SET	SET
SET	SET
SIZ	SIZ
SPS	SPS
SS\$	SS\$
SS\$	SS\$
SS\$	SS\$
SS\$	SS\$
SS\$	SS\$
SS\$	SS\$
SS\$	SS\$
STA	STA
STI	STI
STR	STR
STR	STR
STS	STS
STS	STS
STS	STS
SUC	SUC
SUC	SUC
SUC	SUC
SUC	SUC
SUC	SUC
SWI	SWI
SYM	SYM
SYS	SYS
SYS	SYS

```
11AE 2893 .sbttl Handle line truncation mode changes
11AE 2894
11AE 2895 .enable lsb
11AE 2896
11AE 2897 tec$truln: ; change truncate lines mode
52 08 D0 11AE 2898      movl    #tec$vet$tru, r2 ; set the ET bit to check
53 09 D0 11B1 2899      movl    #tt$vet$wrap, r3 ; and corresponding characteristic
50 6A D2 11B4 2900      mcoml   i_r0(r10), r0 ; get the new bit(s) to check against
09 11 11B7 2901      brb     10$ ; go join common code
11B9 2902
11B9 2903 .sbttl Handle 8-bit terminal mode changes
11B9 2904
11B9 2905 tec$eight: ; change 8-bit terminal mode
52 0C D0 11B9 2906      movl    #tec$vet$8bt, r2 ; set the ET bit to check
53 0F D0 11BC 2907      movl    #tt$vet$eightbit, r3 ; and corresponding characteristic
50 6A D0 11BF 2908      movl    i_r0(r10), r0 ; get the new bit(s) to check against
00F6'CF B5 11C2 2909 10$: 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
00 06 50 52 E1 11D2 2913 20$: bbc     r2, r0, 30$ ; should we set the characteristic?
00 0004'CF 53 E2 11D6 2914      bbsb    r3, w^devdepend, 30$ ; set bit on in characteristics
0004'CF 15 10 11DC 2915 30$: bsbb    tec$setmode ; go set the terminal mode
EF18 8E D0 11DE 2916      movl    (sp)+, w^devdepend ; restore original characteristics
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+rwsiz(r11), w^devbufsiz ; set the new terminal width
F048 30 11F3 2925 tec$setmode: ; do a IOS_SETMODE on terminal output
11F6 2926      bsbw    tec$wait ; wait for all output to be queued
11F6 2927      $qiow_s - ; queue an I/O request
11F6 2928      ch_n = w^ter_o_chan, - ; on the terminal output channel
11F6 2929      func = s^#io$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
7E 7C 11F8
0C DD 11FA
0000'CF DF 11FC
7E 7C 1200
0010'CF 7F 1202
7E 00' 3C 1206
7E 00F6'CF 3C 1209
00 DD 120E
00000000'GF 0C FB 1210
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$xitnw: ; stop terminal hacks
0100'CF 01 90 1220 2944 movb #1, w^exiting_flag ; exiting flag <- on
50 00000000'GF 9E 1225 2945 enable_ctrlt: ; (re-)enable CTRL/T actions
51 08E8'CF DE 1225 2946 movab g^lib$enable_ctrl, r0 ; address enable oob cli service
00F4'CF B5 122C 2947 moval w^ter_oob_msk, r1 ; enable mask is saved old mask(s)
15 13 1231 2948 10$: tstw w^ter_i_chan ; do we have a terminal for input?
7E 13 1235 2949 beql 30$ ; nope
61 DF 1237 2950 pushal -(sp) ; arg #2 -> returned old mask
60 02 FB 1239 2951 pushal (r1) ; arg #1 -> enable/disable mask
05 50 E9 123B 2952 calls #2, (r0) ; call the enable/disable cli service
08E8'CF 6E C8 1241 2953 blbc r0, 20$ ; skip .OR. if an error occurred...
5E 04 C0 1246 2954 20$: bisl (sp), w^ter_oob_msk ; .OR. together old mask(s)
EE76 30 1249 2955 addl #4, sp ; clean up the stack
05 124C 2956 bsbw success_or_announce ; announce any failure...
124D 2957 30$: 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

```

[illegible]

PSE

SAB
A
TEC
TEC
TEC
TEC
TEC
TEC
TEC
TEC

Pha

Ini
Com
Pas
Syn

```
1263 2967 .sbttl Process special functions
1263 2968
1263 2969 .enable lsb
1263 2970
1263 2971 10$: movzbl (r7)+, r0 ; get a byte from the buffer
1266 2972 beql 40$ ; end of buffer, exit Z=1
1268 2973 cmpb r0, #^a/A/+32 ; should we convert case?
126C 2974 blssu 20$ ; nope
126E 2975 cmpb r0, #^a/Z/+32 ; really?
1272 2976 bgtru 20$ ; nope again
1274 2977 bicb #32, r0 ; yep, force upper case
1277 2978 20$: cmpb r0, #32 ; is it a space?
127A 2979 bneq 40$ ; nope, exit Z=0
127C 2980 30$: cmpb (r7)+, r0 ; yep, is next also a space?
127F 2981 beql 30$ ; multiple spaces, collapse them
1281 2982 decl r7 ; else back up the pre-fetch (& Z=0)
1283 2983 40$: rsb ; exit, Z=1 => end-of-buffer
1284 2984
1284 2985 50$: movzbl #nam$c_maxrss, (r6) ; set maximum length of value
1288 2986 movl r7, 4(r6) ; set address part of value desc
128C 2987 locc #0, (r6), (r7) ; locate the terminating null
1290 2988 subl r0, (r6) ; and find the true length
1293 2989 pushaq (r6) ; arg #2 -> symbol's value
1295 2990 pushaq (r8) ; arg #1 -> symbol's name
1297 2991 calls #2, g^lib$set_symbol ; call the set symbol cli service
129E 2992 53$: bsbw success_or_abrt ; check for success completion
12A1 2993 brw 130$ ; go move value to filename buffer
12A4 2994
12A4 2995 57$: tstw (r8) ; zero length symbol name?
12A6 2996 beql 90$ ; yep, error, go return a value of 0
12A8 2997 pushaw (r6) ; arg #3 -> symbol's value length
12AA 2998 pushaq (r6) ; arg #2 -> symbol's value
12AC 2999 pushaq (r8) ; arg #1 -> symbol's name
12AE 3000 calls #3, g^lib$get_symbol ; call the get symbol cli service
12B5 3001 cmpl r0, #lib$_nosuchsym ; did the symbol exist?
12B8 3002 bneq 53$ ; yes, or other error...
12BE 3003 brw 120$ ; go move null to filename buffer
12C1 3004
12C1 3005 60$: bsbb 10$ ; get the next character
12C3 3006 cmpb r0, #32 ; is it a space?
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
12C8 3009 movab w^tmp_string2_buf, r8
12CD 3010 movl r8, -(r8)
12D0 3011 movzwl #tmp_string2_siz, -(r8)
12D3 3012 movl (r8)+, r1 ; get length of symbol's name buffer
12D6 3013 movl (r8), r2 ; and get a pointer to it's storage
12D9 3014 clrl -(r8) ; reset length of symbol's name desc
12DB 3015 decl r1 ; we store before check; fix length...
12DD 3016 70$: bsbb 10$ ; get next character of symbol's name
12DF 3017 beql 57$ ; end, must be reading the symbol
12E1 3018 cmpb r0, #32 ; is it a space?
12E4 3019 beql 50$ ; yep, that's the name's end
12E6 3017 movb r0, (r2)+ ; store the symbol's name character
12E9 3018 acbl r1, #1, (r8), 70$ ; bump length and check for too far
12EF 3019 brb 90$ ; it's too far, go return a value of 0
12F1 3020
```

```

57  FD86 30 12F1 3021 tec$gexit: ; process special functions
    08ED'CF 9E 12F1 3022 ; go fetch the filename buffer
    FF67 30 12F4 3023 ; address the fetched data
    14 13 12F9 3024 ; get the first command byte
56  02AD'CF 9E 12FC 3025 ; the end, go exit n=0
    56 03 C0 1303 3026 80$: ; get (biased) list of functions
    58 86 D0 1306 3027 ; skip uncomparared characters
    86 50 91 1309 3028 ; pick up pointer to logical name desc
    08 13 130C 3029 ; could it be this function?
    58 D5 130E 3030 ; might be...
    F1 12 1310 3031 ; more to check?
    00'AB B4 1312 3032 90$: ; yes, loop...
    05 1315 3033 ; set return value of 0
    1316 3034 ; exit
    FF4A 30 1316 3036 100$: ; get next character of name
    F7 13 1319 3037 ; no more, go return value of 0
    86 50 91 131B 3038 ; is the character correct?
    F2 12 131E 3039 ; nope, go return value of 0
    66 95 1320 3040 ; yes, more to check?
    F2 12 1322 3041 ; there's more, loop...
    1324 3042 ; reset & get desc for temp string
56  0058'CF 9E 1324 ; getdesc tmp_string, r6
    76 56 D0 1329 ; movab w^tmp_string_buf, r6
    76 3F 3C 132C ; movl r6, -(r6)
    58 D5 132F 3043 ; movzwl #tmp_string_siz, -(r6)
    8E 13 1331 3044 ; tstl r8
    FF2D 30 1333 3045 ; beql 60$
    30 13 1336 3046 ; bsbw 10$
    20 50 91 1338 3047 ; beql 110$
    D5 12 133B 3048 ; cmpb r0, #32
    86 3F 3C 133D 3049 ; bneq 90$
    66 57 D0 1340 3050 ; movzwl #63, (r6)+
    67 3F 00 3A 1343 3051 ; movl r7, (r6)
    76 50 C2 1347 3052 ; locc #0, #63, (r7)
    4B 13 134A 3053 ; subl r0, -(r6)
    87'AF 9F 134C 3054 ; beql 150$
    7E 68 7D 134F 3055 ; pushab b^130$
    7E 66 7D 1352 3056 ; movq (r8), -(sp)
    6E 7F 1355 3057 ; movq (r6), -(sp)
    0C AE 7F 1357 3058 ; pushaq (sp)
    00000000'GF 02 FB 135A 3059 ; pushaq 4+8(sp)
    52 0000'8F 3C 1361 3060 ; calls #2, g^lib$set_logical
    45 11 1366 3061 ; movzwl #ss$-supersede, r2
    1368 3062 ; brb 160$
    1368 3063 110$: ; $trnlog_s -
    1368 3064 ; lognam=(r8), -
    1368 3065 ; rsl len=(r6), -
    1368 3066 ; rslbuf=(r6)
    00 DD 1368 ; PUSHL #0
    00 DD 136A ; PUSHL #0
    00 DD 136C ; PUSHL #0
    66 7F 136E ; PUSHAQ (r6)
    66 3F 1370 ; PUSHAW (r6)
    68 7F 1372 ; PUSHAQ (r8)
    00000000'GF 06 FB 1374 ; CALLS #6, G^SYS$STRNLOG
    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, filprt ; move to filename buffer
00000000'EF 63 94 138F 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
```

```
1388 3088 .sbtll Get additional memory
1388 3089
1388 3090 .enable lsb
1388 3091
24 AA D6 1388 3092 10$: incl i_ps(r10) ; indicate failure with c=1
05 1388 3093 rsb ; and exit with failure
138C 3094
138C 3095 tec$size: ; get additional memory
138C 3096 addl3 #512-1, i_r1(r10), r1 ; get the request amount
51 04 AA 000001FF 8F C1 138C 3096 bicl #512-1, r1 ; rounded up to a page multiple
51 000001FF 8F CA 13C5 3097 cmpl r1, w^still_free ; is there room left?
00E8'CF 51 D1 13CC 3098 bgtru 10$ ; nope, we must fail
00E8'CF 51 E5 1A 13D1 3099 subl r1, w^still_free ; move the requested amount
00'AB 51 C2 13D3 3100 addw r1, b^curfre(r11) ; to teco's free space
6B 00'AB 07 E0 13DC 3102 bbs #tec$vet$xit, b^etype(r11), 40$ ; doing announcements?
0000'8F 00'AB B1 13E1 3103 cmpw b^qcmd(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
00 00000000'EF 00' E3 13F3 3108 bs s^io$vet$cancel, ctlofg ; do a control/o cancel
13F3 3108 bbs s^io$vet$cancel, ctlofg, 30025$
13FB 3109
56 5B 8F 9A 13FB 3109 movzbl #^a/[/, r6 ; start with '['
F292 30 13FF 3110 bsbw echo_buffer ; in the message
56 00'AB 00'AB A1 1402 3111 addw3 b^qrstor(r11), b^qmax(r11), r6 ; find end of
56 00'AB A0 1408 3112 addw b^curfre(r11), r6 ; total space in use
56 5B A2 140C 3113 subw r11, r6 ; now find total byte size in use
56 5B 3C 140F 3114 movzwl r6, r6 ; now convert that to a longword
56 000001FF 8F C0 1412 3115 addl #511, r6 ; round and
56 00000200 8F C6 1419 3116 divl #512, r6 ; find that size in pages
57 56 56 0A 7B 1422 3117 clrl r7 ; clear the high order
11 13 1427 3119 ediv #10, r6, r6, r7 ; and divide to tens/units
58 56 0A C3 1429 3120 beql 30$ ; no tens
09 19 142D 3121 subl3 #10, r6, r8 ; are tens really up to hundreds?
56 31 9A 142F 3122 blss 20$ ; nope
F25F 30 1432 3123 movzbl #^a/1/, r6 ; yep, set a '1' for the hundreds
56 5B D0 1435 3124 bsbw echo_buffer ; and go output it
13 10 1438 3125 movl r8, r6 ; restore corrected tens
56 57 90 143A 3126 bsbb 50$ ; output result (tens)
0E 10 143D 3127 movb r7, r6 ; copy the remainder (units)
0342'CF 7F 143F 3128 bsbb 50$ ; and go output it
F218 CF 01 FB 1443 3129 pushaq w^sizing_msg_desc ; set the rest of the sizing message
00'AB 01 A8 1448 3130 calls #1, w^tec$out_ascid ; and go output it
05 144C 3131 bisw #1, b^outdne(r11) ; say memory message output done
144D 3132 rsb ; exit
56 30 80 144D 3133 40$: addb #^a/0/, r6 ; make character into a digit
F241 31 1450 3134 brw echo_buffer ; and, then, go buffer it
1453 3135
1453 3136 .disable lsb
```

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

```
1492 3168 .sbtll Exit from TECO
1492 3169
1492 3170 .enable lsb
1492 3171
18 13 1492 3172 10$: beql 20$ ; no file
1494 3173 $close - ; else close
1494 3174 fab=(r6) ; the file
1494 3175 PUSHAL (r6)
00000000'GF 01 DF 1494 3176 CALLS $$$TMP1,G^SYSS$CLOSE
010C'CF 0C A6 DO 149D 3175 movl fab$l_stv(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
1480 3180
1480 3181 tec$teexit: ; exit from teco
56 00000000'EF 30 1480 3182 bsbw echo_dump ; dump out any partial terminal output
D6 10 14B3 3183 movl cmdprm, r6 ; get indirect file fab pointer
56 00000000'EF 30 14BA 3184 bsbw 10$ ; and close it if necessary
CD 10 14BC 3185 movl inpnor, r6 ; get normal input fab pointer
56 00000000'EF 30 14C3 3186 bsbw 10$ ; and close it if necessary
C4 10 14C5 3187 movl inpalt, r6 ; get alternate input fab pointer
56 00000000'EF 30 14CC 3188 bsbw 10$ ; and close it if necessary
BB 10 14CE 3189 movl oupnor, r6 ; get normal output fab pointer
56 00000000'EF 30 14D5 3190 bsbw 10$ ; and close it if necessary
B2 10 14D7 3191 movl oupalt, r6 ; get alternate output fab pointer
50 00F8'CF 3C 14DE 3192 bsbw 10$ ; and close it if necessary
OD 13 14E0 3193 movzwl w^ter_c_chan, r0 ; get terminal control/c ast channel
14E5 3194 beql 30$ ; none
14E7 3195 $dassgn_s - ; deassign channel
14E7 3196 chan=r0 ; from control/c ast channel
00000000'GF 7E 50 3C 14E7 3197 MOVZWL r0,-(SP)
EBCE 30 14EA 3198 CALLS #1,G^SYSS$DASSGN
50 00F4'CF 3C 14F1 3197 bsbw success_or_announce ; announce any failure...
12 13 14F4 3198 30$: movzwl w^ter_i_chan, r0 ; get terminal input channel
14F9 3199 beql 40$ ; none
14FB 3200 $dassgn_s - ; deassign channel
14FB 3201 chan=r0 ; from terminal input
00000000'GF 7E 50 3C 14FB 3202 MOVZWL r0,-(SP)
EBBA 30 14FE 3203 CALLS #1,G^SYSS$DASSGN
FD1A 30 1505 3202 bsbw success_or_announce ; announce any failure...
OE 11 1508 3203 bsbw enable_ctrlt ; go (re-)enable CTRL/T actions
150B 3204 brb 50$ ; continue
150D 3205
150D 3206 40$: $close - ; close
150D 3207 fab=w^input_sys_fab ; sys$input
00000000'GF 07A0'CF DF 150D 3207 PUSHAL w^input_sys_fab
EBA7 30 1511 3208 CALLS $$$TMP1,G^SYSS$CLOSE
00F6'CF B5 1518 3208 bsbw success_or_announce ; announce any failure...
1A 13 151B 3209 50$: tstw w^ter_o_chan ; a terminal output channel?
ECEE 30 151F 3210 beql 60$ ; none
FCCC 30 1521 3211 bsbw tec$wait_done ; wait for terminal output to complete
EB98 30 1524 3212 bsbw tec$setmode ; go (re-)set correct terminal modes
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 3215 MOVZWL w^ter_o_chan,-(SP)
```

[illegible]

00000136	R		04
0000015D	R		04
0000C16B	R		04
00000050	R		07
0000006C	R		07
000000EE	R		04
0000009A	R		04
00000D66	R		08
00000D53	R		08
*****		X	0A
*****		X	08
*****		X	0A
000002B0	R		03
*****		X	08
*****		X	08
*****		X	08
*****		X	08
*****		X	08
000000FC	R		06
000000FD	R		06
000000EC	R		06
*****		X	08
*****		X	0A
00000002	R		07
00000000	R		07
00000004	R		07
00000008	R		07
0000000C	R		07
00000001	R		07
00000E64	R		08
= 00000002			
= 0000000E			
= 0000099D	R		08
= 00000008			
= 00000004			
= 0000000A			
= 0000001C			
= 00000020			
= 00000006			
= 0000000C			
00000694	R		08
000006FC	R		08
00000690	R		08
00000688	R		08
000007F0	R		08
0000068D	R		08
*****		X	08
000001CC	R		08
00001225	R		08
000006EC	R		06
00000740	R		06
00001114	R		08
00000FE7	R		06
000010EF	R		08
00000EE8	R		06
*****		X	08
00000145	R		08

[illegible]

[illegible]

82

[illegible]

TECONAT
Symbol table

VAX-11 TECO

C 2

16-SEP-1984 02:11:05 VAX/VMS Macro V04-00
10-SEP-1984 13:16:05 [TECO.SRC]TECONAT.MAR;3Page 94
(40)

RAB\$B-RAC	=	0000001E		
RAB\$C-BID	=	00000001		
RAB\$C-BLN	=	00000044		
RAB\$C-SEQ	=	00000000		
RAB\$C-CTX	=	00000018		
RAB\$C-RBF	=	00000028		
RAB\$C-RHB	=	0000002C		
RAB\$C-ROP	=	00000004		
RAB\$C-STV	=	0CJ0000C		
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
RM\$S-EOF	*****		X	08
RM\$S-FNF	*****		X	08
RM\$S-FSZ	*****		X	0A
RM\$S-IFI	*****		X	08
RM\$S-NMF	*****		X	08
RM\$S-NORMAL	*****		X	08
RM\$S-PRV	*****		X	08
RM\$S-SUPERSEDE	*****		X	08
RM\$S-SYN	*****		X	08
RW\$SIZE	*****		X	0A
SAVED-SP		000000E4	R	06
SAVE-DATA		000009D7	R	08
SCH\$OF	*****		X	0A
SCH\$RT	*****		X	0A
SET-FILENAME		00000CDC	R	08
SET-INPUTNAME		00000CBD	R	08
SET-OUTPUTNAME		00000CB1	R	08
SIZING-MSG-DESC		00000342	R	03
SP\$SET	*****		X	0A
SS\$-ABORT	*****		X	0A
SS\$-CONTROL C	*****		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
ST\$SK-SEVERE	=	00000004		
ST\$SM-INHIB MSG	=	10000000		
ST\$SM-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
SYM\$PC	*****		X	0A
SY\$S\$ASSIGN	*****		GX	0A
SY\$S\$CANCEL	*****		GX	08

SY\$S\$CLI	*****	X	0A
SY\$S\$CLOSE	*****	GX	0A
SY\$S\$CLREF	*****	GX	08
SY\$S\$CONNECT	*****	GX	0A
SY\$S\$CREATE	*****	GX	0A
SY\$S\$CRELOG	*****	GX	0A
SY\$S\$DASSGN	*****	GX	08
SY\$S\$DCLCMH	*****	GX	0A
SY\$S\$EXIT	*****	GX	08
SY\$S\$GET	*****	GX	08
SY\$S\$GETDVI	*****	GX	0A
SY\$S\$GETJPI	*****	GX	08
SY\$S\$NUMTIM	*****	GX	08
SY\$S\$OPEN	*****	GX	0A
SY\$S\$PARSE	*****	GX	08
SY\$S\$PURGWS	*****	GX	08
SY\$S\$PUT	*****	GX	08
SY\$S\$PUTMSG	*****	GX	08
SY\$S\$QIO	*****	GX	08
SY\$S\$QIOW	*****	GX	08
SY\$S\$REWIND	*****	GX	08
SY\$S\$SEARCH	*****	GX	08
SY\$S\$STRNLOG	*****	GX	0A
SY\$S\$WAITFR	*****	GX	08
TS\$ANSI	00000100	G	
TS\$DEC	00002000	G	
TS\$EDIT	00001000	G	
TS\$VT05	00000001	G	
TAG\$BUF	*****	X	0A
TAG\$SRT	*****	X	0A
TEC\$S\$ALLER	00000DB4	R	08
TEC\$S\$BAKUP	0000087E	R	08
TEC\$S\$CLSFL	00000D13	R	08
TEC\$S\$CLSOF	00000D15	R	08
TEC\$S\$CMTPAP	00000000	R	08
TEC\$S\$CTRLCAST	00000194	R	08
TEC\$S\$DATE	00001453	R	08
TEC\$S\$DELCH	0000079F	R	08
TEC\$S\$DELLN	00000769	R	08
TEC\$S\$EIGHT	000011B9	R	08
TEC\$S\$EJFLG	00001158	R	08
TEC\$S\$GETBF	00000912	R	08
TEC\$S\$GETFL	00000DC7	R	08
TEC\$S\$GEXIT	000012F1	R	08
TEC\$S\$INPSV	00000CB7	R	08
TEC\$S\$INPUT	00000576	R	08
TEC\$S\$KILFL	00000D5C	R	08
TEC\$S\$ED\$CTL	00000001		
TEC\$S\$ED\$EXP	00000004		
TEC\$S\$ED\$IMD	00000020		
TEC\$S\$ED\$INC	00000040		
TEC\$S\$ED\$SRH	00000010		
TEC\$S\$ED\$WCH	00000080		
TEC\$S\$ED\$YNK	00000002		
TEC\$S\$ET\$8BT	00001000		
TEC\$S\$ET\$BIN	00000001		
TEC\$S\$ET\$CC	00008000		

TECONAT
Symbol table

VAX-11 TECO

D 2

16-SEP-1984 02:11:05 VAX/VMS Macro V04-00
10-SEP-1984 13:16:05 [TECO.SRC]TECONAT.MAR;3Page 95
(40)

**F

TECSM_ETSCCO	00000010		
TECSM_ETSCKE	00000020		
TECSM_ETSCRT	00000002		
TECSM_ETSDET	00000040		
TECSM_ETSGRV	00002000		
TECSM_ETSIAF	00000200		
TECSM_ETSIC	00000004		
TECSM_ETSINCH	00000008		
TECSM_ETSIFS	00000400		
TECSM_ETSTRU	00000100		
TECSM_ETSXIT	00000080		
TECSOUPUT	00000371	R	08
TECSOUPUT_AST	000002BE	R	08
TECSOUPUT_MORE	00000375	R	08
TECSOUTSV	00000CAB	R	08
TECSOUT_ASCID	00000660	RG	08
TECSPUTBF	00000A47	R	08
TECSSETMODE	000011F3	R	08
TECSSIZER	000013BC	R	08
TECSTEKO	00000000	RG	0A
TECSTEXIT	00001480	R	08
TECSTIME	0000146B	R	08
TECSTRULN	000011AE	R	08
TECSV_ED\$CTL	00000000		
TECSV_ED\$EXP	00000002		
TECSV_ED\$IMD	00000005		
TECSV_ED\$INC	00000006		
TECSV_ED\$SRH	00000004		
TECSV_ED\$WCH	00000007		
TECSV_ED\$YNK	00000001		
TECSV_ET\$BET	0000000C		
TECSV_ET\$BIN	00000000		
TECSV_ET\$CC	0000000F		
TECSV_ET\$CCK	00000004		
TECSV_ET\$CCK	00000005		
TECSV_ET\$CCK	00000001		
TECSV_ET\$DET	00000006		
TECSV_ET\$GRV	00000000		
TECSV_ET\$IAS	00000009		
TECSV_ET\$IC	00000002		
TECSV_ET\$INCH	00000003		
TECSV_ET\$IFS	0000000A		
TECSV_ET\$TRU	00000008		
TECSV_ET\$XIT	00000007		
TECSWAIT	0000023E	R	08
TECSWAIT_DONE	00000212	R	08
TECSWAIT_DONE_LF	000001ED	R	08
TECSWIDTR	000011E6	R	08
TECSXITNW	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	00000028	R	03
TER_I_BUF	00000000	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	00000020	R	03
TER_I_NOR8_TRM	00000028	R	06
TER_I_NOR_TRM_PTR	000000E0	R	06
TER_I_SIZ	00000020		
TER_I_STATUS	00000018	R	06
TER_OOB_MSK	000000E8	R	06
TER_O_BUF1	00000000	R	05
TER_O_BUF2	00000020	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	00000020		
TER_O_STATUS1	00000000	R	06
TER_O_STATUS2	00000008	R	06
TER_O_TABLE	00000000	R	03
TER_O_UNIT	000000FA	R	06
TEXT	*****	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		
TTSM_ESCAPE	= 00000008		
TTSM_HALFDUP	= 00100000		
TTSV_EIGHTBIT	= 0000000F		
TTSV_LOWER	= 00000007		
TTSV_SCOPE	= 0000000C		
TTSV_WRAP	= 00000009		
TTT_VTOS	= 00000001		
TT2SM_ANSICRT	= 01000000		
TT2SM_DECCRT	= 20000000		

TECONAT
Symbol table

VAX-11 TECO

E 2

16-SEP-1984 02:11:05 VAX/VMS Macro V04-00
10-SEP-1984 13:16:05 [TECO.SRC]TECONAT.MAR;3Page 96
(40)

```
TT2SM EDIT      = 10000000
TTOBFC          ***** X 08
TTOBUF          ***** X 08
TTOINT          ***** X 08
TTOMOD          ***** X 08
TTOPTR          ***** X 08
TXSTOR          ***** X 0A
UNIT            0000004C R 07
XABSB_MTACC     = 0000000A
XABSB_PROT_MODE = 00000010
XABSB_PROT_OPT  = 00000008
XABSC_PRO       = 00000013
XABSC_PROLEN    = 00000058
XABSL_ACLBUF    = 00000018
XABSL_ACLCTX    = 00000020
XABSL_NXT       = 00000004
XABSM_MODEL     = 00000008
XABSM_NOEXE     = 00000004
XABSM_NOREAD    = 00000001
XABSM_NOWRITE   = 00000002
XABSV_OWN       = 00000004
XABSV_ACLSIZ    = 0000001C
XABSV_GRP       = 0000000E
XABSV_MBM       = 0000000C
XABSV_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
\$ABSS	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

TECONAT
VAX-11 Macro Run Statistics

VAX-11 TECO

F 2

16-SEP-1984 02:11:05 VAX/VMS Macro V04-00
10-SEP-1984 13:16:05 [TECO.SRC]TECONAT.MAR;3

Page 97
(40)

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

_S255SDUA28:[SYSLIB]STARLET.MLB;2

59

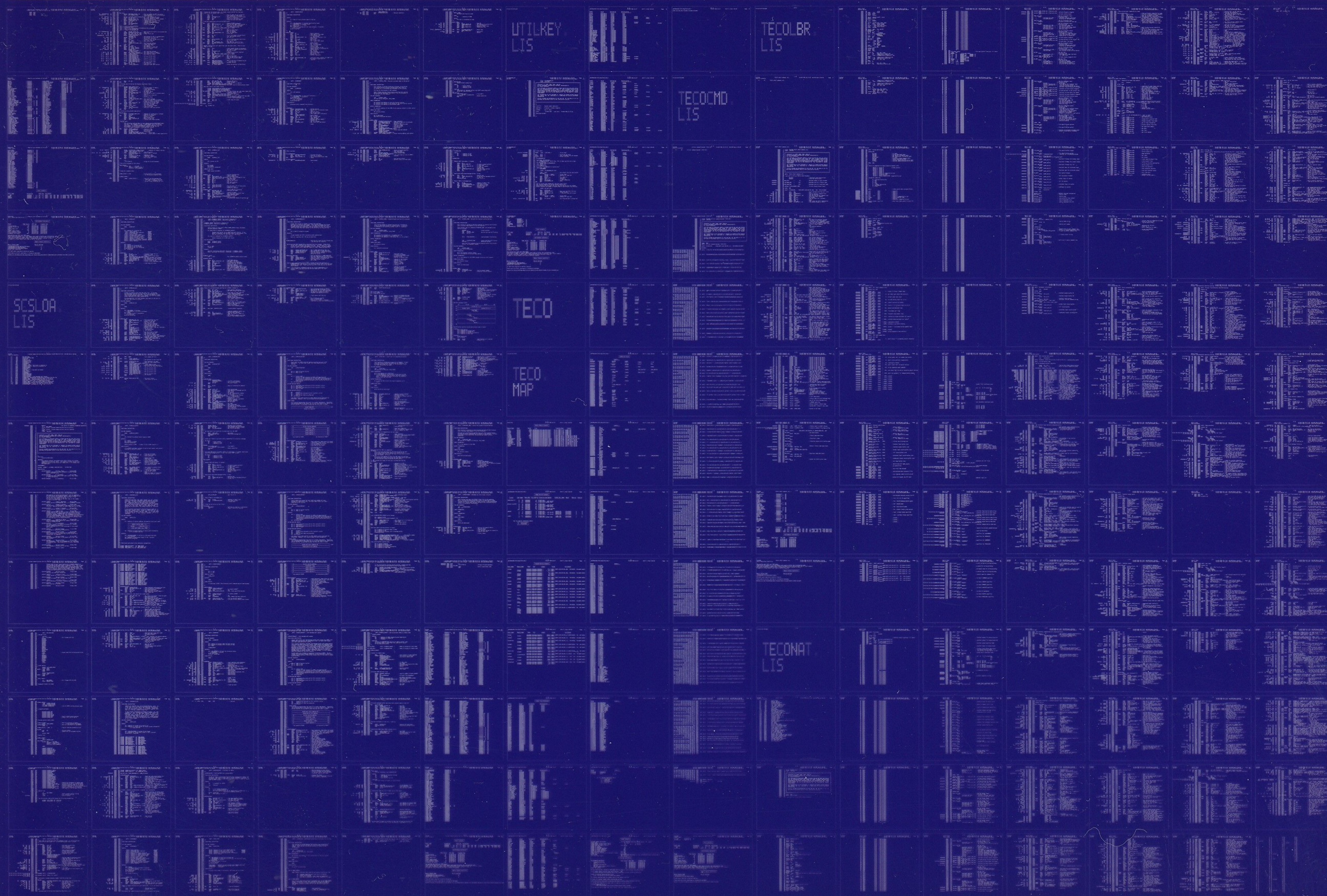
2009 GETS were required to define 59 macros.

There were no errors, warnings or information messages.

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

0399 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



0400 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

TRACMSG
MDL

TBKLIB
REQ

TRACE

TRACE
MAP

STRUCDEF
REQ

TBKPROLOG
REQ

STRUCDEF
LIS

TBKDST
REQ