

RRRRRRRRRRR		TTTTTTTTTTTT	PPPPPPPPPPP		AAAAAAAAA		DDDDDDDDDDD	
RRRRRRRRRRR		TTTTTTTTTTTT	PPPPPPPPPPP		AAAAAAAAA		DDDDDDDDDDD	
RRRRRRRRRRR		TTTTTTTTTTTT	PPPPPPPPPPP		AAAAAAAAA		DDDDDDDDDDD	
RRR	FRR	TTT	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RRR	TTT	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RRR	TTT	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RRR	TTT	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RRR	TTT	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RRR	TTT	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RRR	TTT	PPP	PPP	AAA	AAA	DDD	DDD
RRRRRRRRRRR		TTT	PPPPPPPPPPP		AAA	AAA	DDD	DDD
RRRRRRRRRRR		TTT	PPPPPPPPPPP		AAA	AAA	DDD	DDD
RRRRRRRRRRR		TTT	PPPPPPPPPPP		AAA	AAA	DDD	DDD
RRR	RRR	TTT	PPP	PPP	AAAAAAAAAAAAAAAA	AAA	DDD	DDD
RRR	RRR	TTT	PPP	PPP	AAAAAAAAAAAAAAAA	AAA	DDD	DDD
RRR	RRR	TTT	PPP	PPP	AAAAAAAAAAAAAAAA	AAA	DDD	DDD
RRR	RRR	TTT	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RRR	TTT	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RRR	TTT	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RRR	TTT	PPP	PPP	AAA	AAA	DDDDDDDDDDD	DDD
RRR	RRR	TTT	PPP	PPP	AAA	AAA	DDDDDDDDDDD	DDD
RRR	RRR	TTT	PPP	PPP	AAA	AAA	DDDDDDDDDDD	DDD

```

CCCCCCCC  TTTTTTTTTT  DD DDDDD  SSSSSSSS  EEEEEEEEE  NN  NN  SSSSSSSS  EEEEEEEEE
CCCCCCCC  TTTTTTTTTT  DD DDDDD  SSSSSSSS  EEEEEEEEE  NN  NN  SSSSSSSS  EEEEEEEEE
CC         TT         DD     DD  SS         EE         NN  NN  SS         EE
CC         TT         DD     DD  SS         EE         NN  NN  SS         EE
CC         TT         DD     DD  SS         EE         NN  NN  SS         EE
CC         TT         DD     DD  SS         EE         NN  NN  SS         EE
CC         TT         DD     DD  SS         EE         NN  NN  SS         EE
CC         TT         DD     DD  SS         EE         NN  NN  SS         EE
CC         TT         DD     DD  SS         EE         NN  NN  SS         EE
CC         TT         DD     DD  SS         EE         NN  NN  SS         EE
CCCCCCCC  TT         DDDDDDD  SSSSSSSS  EEEEEEEEE  NN  NN  SSSSSSSS  EEEEEEEEE
CCCCCCCC  TT         DDDDDDD  SSSSSSSS  EEEEEEEEE  NN  NN  SSSSSSSS  EEEEEEEEE

```

```

LL         IIIII  SSSSSSSS
LL         IIIII  SSSSSSSS
LL         II     SS
LL         II     SS
LL         II     SS
LL         II     SS
LL         II     SSSSSS
LL         II     SSSSSS
LL         II     SS
LL         II     SS
LL         II     SS
LL         II     SS
LLLLLLLLLL IIIII  SSSSSSSS
LLLLLLLLLL IIIII  SSSSSSSS

```

(2) 67
(3) 298

DECLARATIONS
CT_POST_SENSE - Map TSA into VMS

```

0000 1 .IF DF RTPAD
0000 2 .TITLE CTSETRT - RTPAD/CTERM SET CHARACTERISTICS
0000 3 .IFF
0000 4 .TITLE CTDSENSE - CTDRIVER SENSE MODE PROCESSING
0000 5 .ENDC
0000 6 .IDENT 'V04-000'
0000 7 .ENABLE SUPPRESSION
0000 8 :
0000 9 :*****
0000 10 :*
0000 11 :* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 12 :* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 13 :* ALL RIGHTS RESERVED.
0000 14 :*
0000 15 :* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 16 :* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 17 :* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 18 :* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 19 :* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 20 :* TRANSFERRED.
0000 21 :*
0000 22 :* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 23 :* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 24 :* CORPORATION.
0000 25 :*
0000 26 :* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 27 :* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 28 :*
0000 29 :*
0000 30 :*****
0000 31 :
0000 32 :
0000 33 :++
0000 34 :
0000 35 : FACILITY:
0000 36 :
0000 37 : CTERM Remote terminal protocol driver
0000 38 :
0000 39 : ABSTRACT:
0000 40 :
0000 41 : This module is called to map a characteristics item list received
0000 42 : from the net into VMS terminal characteristics.
0000 43 :
0000 44 : ENVIRONMENT:
0000 45 :
0000 46 :
0000 47 :
0000 48 :--
0000 49 :
0000 50 : AUTHOR: Jake VanNoy, CREATION DATE: 23-Aug-1982
0000 51 :
0000 52 : MODIFIED BY:
0000 53 :
0000 54 :
0000 55 : V03-003 JLV0336 Jake VanNoy 28-FEB-1984
0000 56 : Use constant names in SENSETAB table.
0000 57 :

```

0000	58	:	V03-002	JLV0288	Jake VanNoy	28-JUL-1983
0000	59	:			Update symbols that identify characteristics.	
0000	60	:				
0000	61	:	V03-001	JLV0261	Jake VanNoy	26-MAY-1983
0000	62	:			Change baud rate tables to new format.	
0000	63	:				
0000	64	:				
0000	65	**				

```

0000 67          .SBTTL  DECLARATIONS
0000 68          :
0000 69          : INCLUDE FILES:
0000 70          :
0000 71          $DCDEF
0000 72          $$SDEF
0000 73          $TTDEF
0000 74          $TT2DEF
0000 75          $TSADEF
0000 76          $UCBDEF          ; for protocol errors
0000 77          :
0000 78          :
0000 79          : MACROS:
0000 80          :
0000 81          :
0000 82          .MACRO  CHAR    LABEL
0000 83          .WORD    char_type!char_count
0000 84          .WORD    LABEL-SENSE_ROOTINES
0000 85          char_count = char_count + 1
0000 86          .ENDM   CHAR
0000 87          :
0000 88          .MACRO  BAUDTAB BPS
0000 89          .WORD    BPS
0000 90          .BYTE    TT$c_BAUD_'BPS'
0000 91          .ENDM   BAUDTAB
0000 92          :
0000 93          .MACRO  TERMTAB TERM,VALUE
0000 94          .BYTE    VALUE
0000 95          .ASCIC   /TERM/
0000 96          .ENDM   TERMTAB
0000 97          :
0000 98          .MACRO  SENSETAB CHAR
0000 99          .WORD    char_type!char
0000 100         .ENDM   SENSETAB
0000 101         :
0000 102         :
0000 103         :
0000 104         : EQUATED SYMBOLS:
0000 105         :
0000 106         :
00000000 0000 107 physical = ch$c_physical@8          ; 0 leftshifted 1 byte
00000100 0000 108 logical  = ch$c_logical@8          ; 1 leftshifted 1 byte
00000200 0000 109 cterm   = ch$c_cterm@8            ; 2 leftshifted 1 byte
0000 110         :
00000000 0000 111 TT_BUF   = 0
00000004 0000 112 TT_CHAR1 = 4
00000008 0000 113 TT_CHAR2 = 8
00000000 0000 114 TT_IOSB  = 0
0000 115         :
0000 116         :
0000 117         : OWN STORAGE:
0000 118         :
0000 119         :
0000 120         .IF DF RTPAD
0000 121         .PSECT RTPAD,NOWRT
0000 122         :
0000 123         .EXTERNAL OUTBAND_NEW

```

```

0000 124 .IFF
00000000 125 .PSECT $$$115_DRIVER, LONG
0000 126
0000 127 .ENDC
0000 128
0000 129 SENSE_TABLE:
0000 130
00000000 0000 131 char_type = physical
00000001 0000 132 char_count = 1
0000 133 CHAR INPUT_SPEED : 1
0004 134 CHAR OUTPUT_SPEED : 2
0008 135 CHAR CHARACTER_SIZE : 3
000C 136 CHAR PARITY_ENABLE : 4
0010 137 CHAR PARITY_TYPE : 5
0014 138 CHAR MODEM_PRESENT : 6
0018 139 CHAR AUTOBAUD_DETECT : 7
001C 140 CHAR MANAGEMENT_GUARANTEED : 8
0020 141 CHAR SWITCH_CHAR_1 : 9
0024 142 CHAR SWITCH_CHAR_2 : 10
0028 143
00000100 0028 144 char_type = logical
00000001 0028 145 char_count = 1
0028 146
0028 147 CHAR MODE_WRITING_ALLOWED : 1
002C 148 CHAR TERMINAL_TYPE : 2
0030 149 CHAR TERMINAL_SUBTYPE : 3
0034 150 CHAR OUTPUT_FLOW_CONTROL : 4
0038 151 CHAR OUTPUT_PAGE_STOP : 5
003C 152 CHAR FLOW_CHAR_PASS_THRU : 6
0040 153 CHAR INPUT_FLOW_CONTROL : 7
0044 154 CHAR LOSS_NOTIFICATION : 8
0048 155 CHAR LINE_WIDTH : 9
004C 156 CHAR PAGE_LENGTH : 10
0050 157 CHAR STOP_LENGTH : 11
0054 158 CHAR CR_FILL : 12
0058 159 CHAR LF_FILL : 13
005C 160 CHAR WRAP : 14
0060 161 CHAR HORIZONTAL_TAB : 15
0064 162 CHAR VERTICAL_TAB : 16
0068 163 CHAR FORM_FEED : 17
006C 164
00000200 006C 165 char_type = cterm
00000001 006C 166 char_count = 1
006C 167
006C 168 CHAR IGNORE_INPUT : 1
0070 169 CHAR CHAR_ATTRIBUTES : 2
0074 170 CHAR CONTROL_O_PASS_THRU : 3
0078 171 CHAR RAISE_INPUT : 4
007C 172 CHAR NORMAL_ECHO : 5
0080 173 CHAR INPUT_ESCAPE_ENABLE : 6
0084 174 CHAR OUTPUT_ESCAPE_ENABLE : 7
0088 175 CHAR INPUT_COUNT_STATE : 8
008C 176 CHAR AUTO_PROMPT : 9
0090 177 CHAR ERROR_PROCESSING : 10
0094 178
FFFF 0094 179 .WORD -1 : End of list (any negative number)
U096 180

```

```

0096 181 SENSE_BAUD: ; Table must be in ascending order
0096 182 BAUDTAB 50
0099 183 BAUDTAB 75
009C 184 BAUDTAB 110
009F 185 BAUDTAB 134
00A2 186 BAUDTAB 150
00A5 187 BAUDTAB 300
00A8 188 BAUDTAB 600
00AB 189 BAUDTAB 1200
00AE 190 BAUDTAB 1800
00B1 191 BAUDTAB 2000
00B4 192 BAUDTAB 2400
00B7 193 BAUDTAB 3600
00BA 194 BAUDTAB 4800
00BD 195 BAUDTAB 7200
00C0 196 BAUDTAB 9600
00C3 197 BAUDTAB 19200
00000000 00C6 198 .LONG 0
00CA 199
00CA 200 CT$AB_TERM TABLE: ; Table is in order for most likely first
00CA 201 TERMTAB VT100, TTS_VT100
00D1 202 TERMTAB VT200, TTS_VT200-Series
00D8 203 TERMTAB VT101, TTS_VT101
00DF 204 TERMTAB VT102, TTS_VT102
00E6 205 TERMTAB VT105, TTS_VT105
00ED 206 TERMTAB VT125, TTS_VT125
00F4 207 TERMTAB VT131, TTS_VT131
00FB 208 TERMTAB VT132, TTS_VT132
0102 209 TERMTAB VT52, TTS_VT52
0108 210 TERMTAB VT05, TTS_VT05
010E 211 TERMTAB VK100, TTS_VK100
0115 212 TERMTAB VT173, TTS_VT173
011C 213 : TERMTAB TTS_FT1
011C 214 : TERMTAB TTS_FT2
011C 215 : TERMTAB TTS_FT3
011C 216 : TERMTAB TTS_FT4
011C 217 : TERMTAB TTS_FT5
011C 218 : TERMTAB TTS_FT6
011C 219 : TERMTAB TTS_FT7
011C 220 : TERMTAB TTS_FT8
011C 221 TERMTAB LA36, TTS_LA36
0122 222 TERMTAB LA12, TTS_LA12
0128 223 TERMTAB LA34, TTS_LA34
012E 224 TERMTAB LA38, TTS_LA38
0134 225 TERMTAB LA12, TTS_LA12
013A 226 TERMTAB LA100, TTS_LA100
0141 227 TERMTAB LA24, TTS_LA24
0147 228 TERMTAB LQP02, TTS_LQP02
014E 229 TERMTAB VT55, TTS_VT55
00 0154 230 .BYTE 0
4E 57 4F 4E 4B 4E 55 00' 0155 231
07 0155 232 UNKNOWN_TT: .ascii /UNKNOWN/
015D 233
015D 234 .IF NDF RTPAD ; IF CTDRIVER...
015D 235
015D 236 CTP$AB_SENSEBUF:

```



```

0A 015D 237
00 015D 238          .BYTE          CTP$C_MT_READ_CHAR      ; message type
015E 239          .BYTE          0                        ; flags
015F 240
015F 241
015F 242          ; Item list, for now, request everything.
015F 243          ;
015F 244
00000000 015F 245 char_type = physical
015F 246
015F 247          SENSETAB      ch$c_ph_in_speed          ; INPUT SPEED
0161 248          SENSETAB      ch$c_ph_out_speed         ; OUTPUT SPEED
0163 249          SENSETAB      ch$c_ph_char_size        ; CHARACTER SIZE
0165 250          SENSETAB      ch$c_ph_parity_enable    ; PARITY_ENABLE
0167 251          SENSETAB      ch$c_ph_parity_type     ; PARITY_TYPE
0169 252          SENSETAB      ch$c_ph_modem_present   ; MODEM_PRESENT
016B 253          SENSETAB      ch$c_ph_autobaud        ; AUTOBAUD_DETECT
016D 254          :::          SENSETAB      ch$c_ph_manage_guar ; MANAGEMENT_GUARANTEED
016D 255          :::          SENSETAB      ch$c_ph_switch1  ; SWITCH_CHAR_1
016D 256          :::          SENSETAB      ch$c_ph_switch2  ; SWITCH_CHAR_2
016D 257          :::          SENSETAB      ch$c_ph_manage_ena ; MANAGEMENT_ENABLED
016D 258
00000100 016D 259 char_type = logical
016D 260
016D 261          SENSETAB      ch$c_lg_mode_writing      ; MODE WRITING ALLOWED
016F 262          SENSETAB      ch$c_lg_term_bits       ; TERMINAL_TYPE
0171 263          SENSETAB      ch$c_lg_term_type      ; TERMINAL_SUBTYPE
0173 264          SENSETAB      ch$c_lg_output_flow    ; OUTPUT_FLOW_CONTROL
0175 265          SENSETAB      ch$c_lg_page_stop     ; OUTPUT_PAGE_STOP
0177 266          SENSETAB      ch$c_lg_flow_char_pass ; FLOW_CHAR_PASS_THRU
0179 267          SENSETAB      ch$c_lg_input_flow    ; INPUT_FLOW_CONTROL
017B 268          SENSETAB      ch$c_lg_loss_notif     ; LOSS_NOTIFICATION
017D 269          SENSETAB      ch$c_lg_line_width    ; LINE_WIDTH
017F 270          SENSETAB      ch$c_lg_page_length   ; PAGE_LENGTH
0181 271          SENSETAB      ch$c_lg_stop_length   ; STOP_LENGTH
0183 272          SENSETAB      ch$c_lg_cr_fill       ; CR_FILL
0185 273          SENSETAB      ch$c_lg_lf_fill       ; LF_FILL
0187 274          SENSETAB      ch$c_lg_wrap          ; WRAP
0189 275          SENSETAB      ch$c_lg_hor_tab       ; HORIZONTAL_TAB
018B 276          SENSETAB      ch$c_lg_verf_tab      ; VERTICAL_TAB
018D 277          SENSETAB      ch$c_lg_form_feed    ; FORM_FEED
018F 278
00000200 018F 279 char_type = cterm
018F 280
018F 281          SENSETAB      ch$c_ct_ignore_input      ; IGNORE_INPUT
0191 282          :::          SENSETAB      ch$c_ct_char_attrib ; CHAR_ATTRIBUTES
0191 283          SENSETAB      ch$c_ct_ctrl_o_pass    ; CONTROL_O_PASS_THRU
0193 284          SENSETAB      ch$c_ct_raise_input    ; RAISE_INPUT
0195 285          SENSETAB      ch$c_ct_normal_echo    ; NORMAL_ECHO
0197 286          SENSETAB      ch$c_ct_input_esc     ; INPUT_ESCAPE_ENABLE
0199 287          :::          SENSETAB      ch$c_ct_output_esc  ; OUTPUT_ESCAPE_ENABLE
0199 288          SENSETAB      ch$c_ct_input_count   ; INPUT_COUNT_STATE
019B 289          SENSETAB      ch$c_ct_auto_prompt   ; AUTO_PROMPT
019D 290          SENSETAB      ch$c_ct_error_processing ; ERROR_PROCESSING
019F 291
00000042 019F 292 CTP$K_SENSEBUF == .-CTP$AB_SENSEBUF ; Size
019F 293

```

```

CTD
Sym
AUT
AUT
CHS
CHS
CHS
CHS
CHS
CHS
CHS
CHS
CHS
CHS
CHS
CHS
CHS
CHS
CHS
CHS
CHS
CHS
CHS
CHS
CHS
CHS
CHS
CHS
CHS
CHS
CHS
CHA
CHA
CHA
CHA
CON
CR
CT$
CTE
CTP
CTP
CTP
CTP
CTP
CTP
CTP
CT
DC$
ERR
FLO

```

CTDSENSE
V04-000

- CTDRIVER SENSE MODE PROCESSING L 3
DECLARATIONS

16-SEP-1984 02:24:25 VAX/VMS Macro V04-00
5-SEP-1984 03:14:35 [RTPAD.SRC]CTDSENSE.MAR;1

Page 7
(2)

019F 294 .ENDC
019F 295 .nlist meb
019F 296

CTD
Sym
TT8
TT8
TT8
TT8
TT8
TT8
TT8
TT8
TT8
TT8
TT8
TT8
TT8
TT8
TT8
TT8
TT8
TT8
TT8
TT8
TT8
TT8
TT8
TT8
TT8
TT8
TT8
TT8
TT8
TT8
TT2
TT
TT
TT
TT
UCB
UNK
VER
WRA

PSE

¡AB
\$\$\$

```

019F 298 .SBTTL CT_POST_SENSE - Map TSA into VMS
019F 299
019F 300 :
019F 301 : Must map entire set of characteristics returned into VMS sense mode data
019F 302 :
019F 303 :
019F 304 : Input:
019F 305 :     R2 - Address of CTP
019F 306 :     R9 - Address of 12 byte buffer
019F 307 :     R10 - Address of 8 byte buffer
019F 308 :
019F 309 : Output:
019F 310 :     0(R9) - first longword of sense mode data
019F 311 :     4(R9) - first longword of sense mode characteristics
019F 312 :     8(R9) - second longword of sense mode characteristics
019F 313 :
019F 314 :     0(R10) - first longword of status
019F 315 :     4(R10) - second longword of status
019F 316 :
019F 317 : Characteristics returned: (R9)
019F 318 :
019F 319 :     +-----+-----+-----+
019F 320 :     | page width | type | class |
019F 321 :     +-----+-----+-----+
019F 322 :     | length | characteristics |
019F 323 :     +-----+-----+-----+
019F 324 :     | characteristics |
019F 325 :     +-----+-----+-----+
019F 326 :
019F 327 : IOSB: (R10)
019F 328 :     +-----+-----+-----+
019F 329 :     | R speed | T speed | status |
019F 330 :     +-----+-----+-----+
019F 331 :     | 0 | parity | LF fill | CR fill |
019F 332 :     +-----+-----+-----+
019F 333 :
019F 334 :
019F 335 :
019F 336 .IF DF RTPAD
019F 337 CT_CHAR_MSG:: ; RTPAD ENTRY
019F 338 .IFF
019F 339 CT_POST_SENSE:: ; CTDRIVER ENTRY
019F 340 .ENDC
019F 341
019F 342 MOVZWL CTP$W_MSGSIZE(R2),R1 ; Fetch size of buffer
019F 343 MOVAB CTP$W_CH_PARAM(R2),R2 ; Set address of first characteristic
7E 52 2C A2 9E 01A3 344 ADDL3 R1,R2,-(SP) ; Save end address on stack
019F 345 SUBL2 #3,(SP) ; *** fudge, check this **
019F 346
019F 347 MOVB #DC$_TERM,TT_BUF(R9) ; Set terminal class
69 42 8F 90 01AE 348 MOVW #SS$_NORMAL,TT_IOSB(R10); Set status
019F 349
019F 350 10$:
019F 351 CMPL R2,(SP) ; Hit end of list yet?
019F 352 BGEQ 25$ ; Branch if so
019F 353 MOVZWL (R2)+,R6 ; Get parameter
57 FE3F CF 9E 01BA 354 MOVAB SENSE_TABLE,R7 ; Get table address

```

Pha

Ini
Com
Pas
Sym
Pas
Sym
Pse
Cro
Ass

The
866
The
815
19

Mac

-\$2
-\$2
-\$2
-\$2
TOT

165

The

MAC

```

87 56 B1 01C2 355 20$:
    OD 13 01C2 356      CMPW   R6,(R7)+      ; Compare to table entry
    87 B5 01C5 357      BEQL   30$           ; Branch if match found
    F7 18 01C7 358      TSTW   (R7)+         ; Advance pointer past routine
    01C9 359      BGEQ   20$           ; Loop if greater than or equal to zero
    01CB 360
    01CB 361      ; PANIC           ; 'invalid sense mode returned'
    01CB 362      MINOR_ERROR       ; Increment error count, then exit
    01CF 363 25$:
    8E D5 01CF 364      TSTL   (SP)+         ; throw away end address
    01C8 31 01D1 365      BRW    POST_SENSE_EXIT ; exit code
    01D4 366      ; Dispatch to routine
    01D4 367
    01D4 368
    01D4 369 30$:
56 000001E5'EF 9E 01D4 370      MOVAB  SENSE_ROUTINES,R6    ; Base
    57 67 3C 01DB 371      MOVZWL (R7),R7          ; get routine address
    57 56 C0 01DE 372      ADDL2  R6,R7          ; Add offset to base for routine address
    67 16 01E1 373      JSB    (R7)          ; jsb to routine
    D0 11 01E3 374      BRB    10$          ; Do next parameter
    01E5 375
    01E5 376      ;
    01E5 377      ; All these routines have R0,R1,R4,R6-R8 as scratch
    01E5 378      ;
    01E5 379

```

```

01E5 381 SENSE_ROUTINES:
01E5 382
01E5 383 ; physical
01E5 384
01E5 385 INPUT_SPEED:
50 82 B0 01E5 386 MOVW (R2)+,R0 ; physical 1
03 AA 0F 10 01E8 387 BSBB GET_SPEED ; Get speed
03 AA 50 90 01EA 388 MOVB RO,TT_IOSB+3(R10) ; map into TT$C_BAUD_xxxx
05 01EE 389 RSB ; set receive speed
01EF 390 ; Return
01EF 391 OUTPUT_SPEED:
50 82 B0 01EF 392 MOVW (R2)+,R0 ; physical 2
05 10 01F2 393 BSBB GET_SPEED ; Get speed
02 AA 50 90 01F4 394 MOVB RO,TT_IOSB+2(R10) ; map into TT$C_BAUD_xxxx
05 01F8 395 RSB ; set transmit speed
01F9 396 ; Return
01F9 397 ; Local routine to map speed into TT$C_BAUD rates
01F9 398 ;
01F9 399 ; Input:
01F9 400 ; RO - baud rate in BPS
01F9 401 ;
01F9 402 ; Output:
01F9 403 ; RO - Baud rate in TT$C_BAUD_xxx terms
01F9 404 ;
51 FE99 CF 9E 01F9 405 GET_SPEED:
01F9 406 MOVAB SENSE_BAUD,R1 ; Get table
01FE 407 110$:
81 50 B1 01FE 408 CMPW RO,(R1)+ ; compare
08 15 0201 409 BLEQ 120$ ;
81 95 0203 410 TSTB (R1)+ ; Advance
F7 12 0205 411 BNEQ 110$ ; Branch if not zero
50 10 90 0207 412 MOVB #TT$C_BAUD_19200,R0 ; Assume 19200 if > 19200
05 020A 413 RSB ; Return
50 61 90 020B 414 120$: MOVB (R1),R0 ; Fetch baud rate symbol value
05 020E 415 RSB ; Return
020F 416
020F 417 CHARACTER_SIZE:
020F 418 .IF DF RTI D ; physical 3
020F 419 BICL #TT$M_EIGHTBIT,- ;
020F 420 TT_CHAR1(R9) ; Set characteristic
020F 421 .ENDC
82 07 B1 020F 422 CMPW #7,(R2)+ ; Character size
00008000 08 18 0212 423 BGEQ 10$ ; exit if less than or equal to 7
04 A9 C8 0214 424 BISL #TT$M_EIGHTBIT,- ;
021A 425 TT_CHAR1(R9) ; Set characteristic
05 021C 426 10$: RSB ; Return
021D 427
021D 428 PARITY_ENABLE:
021D 429 .IF DF RTPAD ; physical 4
021D 430 BICB #TT$M_PARITY,- ;
021D 431 TT_IOSB+6(R10) ; Set parity enabled
021D 432 .ENDC
50 82 90 021D 433 MOVB (R2)+,R0 ; Get boolean
05 50 E9 0220 434 BLBC RO,10$ ; Branch if not enabled
40 8F 88 0223 435 BISB #TT$M_PARITY,- ;
06 AA 05 0226 436 TT_IOSB+6(R10) ; Set parity enabled
0228 437 10$: RSB ; Return

```

```

0229 438
0229 439 PARITY_TYPE: ; physical 5
0229 440 .IF DF RTPAD
0229 441 BICB #TTSM_ODD,-
0229 442 TT_10SB+6(R10) ; Assume even parity
0229 443 .ENDC
50 82 3C 0229 444 MOVZWL (R2)+,R0 ; Get parity number
50 02 B1 022C 445 CMPW #ch$c_parity_odd,R0 ; Odd?
80 05 12 022F 446 BNEQ 10$ ; If not, exit
06 AA 88 0231 447 BISB #TTSM_ODD,-
0234 448 TT_10SB+6(R10) ; Set odd parity
05 0236 449 10$: RSB ; Return
0237 450
0237 451 MODEM_PRESENT: ; physical 6
0237 452 .IF DF RTPAD
0237 453 BICL #<TTSM_MODEM!-
0237 454 TTSM_REMOTE>,-T_CHAR1(R9) ; Set modem and remote bits
0237 455 .ENDC
50 82 90 0237 456 MOVB (R2)+,R0 ; Get Boolean
08 50 E9 023A 457 BLBC R0,10$ ; Branch if not enabled
04 A9 00202000 8F C8 023D 458 BISL #<TTSM_MODEM!-
0245 459 TTSM_REMOTE>,-TT_CHAR1(R9) ; Set modem and remote bits
05 0245 460 10$: RSB ; Return
0246 461
0246 462 AUTOBAUD_DETECT: ; physical 7
0246 463 .IF DF RTPAD
0246 464 BICL #TT2SM_AUTOBAUD,-
0246 465 TT_CHAR2(R9) ; Set autobaud
0246 466 .ENDC
50 82 90 0246 467 MOVB (R2)+,R0 ; Get Boolean
04 50 E9 0249 468 BLBC R0,10$ ; Branch if not enabled
08 02 C8 024C 469 BISL #TT2SM_AUTOBAUD,-
08 A9 024E 470 TT_CHAR2(R9) ; Set autobaud
05 0250 471 10$: RSB ; Return
0251 472
0251 473 MANAGEMENT_GUARANTEED: ; physical 8
50 82 90 0251 474 MOVB (R2)+,R0 ; Get Boolean
00 50 E9 0254 475 BLBC R0,10$ ; Branch if not enabled
C5 0257 476 10$: RSB ; Return
0258 477
0258 478 SWITCH_CHAR_1: ; physical 9
013A 30 0258 479 BSBW IGNORE_STRING ; Ignore this string data
05 025B 480 RSB ; Return
025C 481
025C 482 SWITCH_CHAR_2: ; physical 10
0136 30 025C 483 BSBW IGNORE_STRING ; Ignore this string data
05 025F 484 RSB ; Return
0260 485

```

```

0260 487 ; logical
0260 488
50 82 90 0260 489 MODE_WRITING_ALLOWED: ; logical 1
00 50 E9 0260 490 MOVB (R2)+,R0 ; Get Boolean
05 0263 491 BLBC R0,10$ ; Branch if not enabled
0266 492 10$: RSB ; Return
0267 493
0267 494 TERMINAL_TYPE: ; logical 2
0267 495
0267 496 ; bit mask is:
0267 497 : 0 - Unknown/known
0267 498 : 1 - scope/hardcopy
0267 499
50 82 3C 0267 500 MOVZWL (R2)+,R0 ; Get characteristic
00001000 8F CB 026A 501 BISL #TT$M_SCOPE,- ;
04 A9 0270 502 TT_CHAR1(R9) ; Set scope
08 50 01 E0 0272 503 BBS #CTP$V.CH_SCOPE,R0,10$ ; Branch if scope
00001000 8F CA 0276 504 BICL #TT$M_SCOPE,- ;
04 A9 027C 505 TT_CHAR1(R9) ; Clear scope
027E 506
027E 507 ; *** unknown?
027E 508
04 50 00 E0 027E 509 10$: BBS #CTP$V.CH_KNOWN,R0,20$ ; Branch if known
01 A9 00 90 0282 510 MOVB #TT$_UNKNOWN,TT_BUF+1(R9) ; Set terminal type
05 0286 511 20$: RSB ; Return
0287 512
0287 513 TERMINAL_SUBTYPE: ; logical 3
56 82 9A 0287 514 MOVZBL (R2)+,R6 ; Get length of string
01 12 028A 515 BNEQ 10$ ; Continue if not 0
05 028C 516 RSB ; Return
028D 517 10$:
57 52 D0 028D 518 MOVL R2,R7 ; Get address of string
52 56 C0 0290 519 ADDL2 R6,R2 ; Update pointer
040C 8F BB 0293 520 PUSHR #*M<R2,R3,R10> ; Save registers
58 FE2F CF 9E 0297 521 MOVAB CT$AB_TERM_TABLE,R8 ; Address of table
54 D4 029C 522 CLRL R4 ; Zero length
029E 523 20$:
58 54 C0 029E 524 ADDL2 R4,R8 ; Get next entry
01 A9 88 90 02A1 525 MOVB (R8)+,TT_BUF+1(R9) ; Set terminal type
0E 13 02A5 526 BEQL 30$ ; branch if end of list
54 88 9A 02A7 527 MOVZBL (R8)+,R4 ; Get length
54 56 91 02AA 528 CMPB R6,R4 ; compare lengths
68 67 EF 12 02AD 529 BNEQ 20$ ; not equal, loop
E9 12 29 02AF 530 CMPC3 R6,(R7),(R8) ; Compare
02B3 531 BNEQ 20$ ; Loop if not equal
02B5 532
02B5 533 ; Terminal type match
02B5 534
02B5 535 ASSUME TT$_UNKNOWN EQ 0 ; assume for end of table code
02B5 536 30$:
040C 8F BA 02B5 537 POPR #*M<R2,R3,R10> ; Restore registers
02B9 538
05 02B9 539 RSB ; Return
02BA 540
02BA 541
02BA 542 OUTPUT_FLOW_CONTROL: ; logical 4
02BA 543 .IF DF-RTPAD

```

```

02BA 544 BICL #TTSM TTSYNC,-
02BA 545 TT_CHAR1(R9) ; Set tt sync
50 82 90 02BA 546 .ENDC
04 50 E9 02BA 547 MOVB (R2)+,R0 ; Get Boolean
20 C8 02BD 548 BLBC R0,10$ ; Branch if not enabled
04 A9 02C0 549 BISL #TTSM TTSYNC,-
05 02C2 550 TT_CHAR1(R9) ; Set tt sync
02C4 551 10$: RSB ; Return
02C5 552
02C5 553 OUTPUT_PAGE_STOP: ; logical 5
02C5 554 .IF DF RTPAD
02C5 555 BICL #TTSM HOLDScreen,-
02C5 556 TT_CHAR1(R9) ; Set hold screen
50 82 90 02C5 557 .ENDC
08 50 E9 02C5 558 MOVB (R2)+,R0 ; Get Boolean
00004000 8F C8 02C8 559 BLBC R0,10$ ; Branch if not enabled
04 A9 02CB 560 BISL #TTSM HOLDScreen,-
05 02D1 561 TT_CHAR1(R9) ; Set hold screen
02D3 562 10$: RSB ; Return
02D4 563
05 82 90 02D4 564 FLOW_CHAR_PASS_THRU: ; logical 6
00 50 E9 02D4 565 MOVB (R2)+,R0 ; Get Boolean
02D7 566 BLBC R0,10$ ; Branch if not enabled
05 02DA 567 10$: RSB ; Return
02DB 568
02DB 569 INPUT_FLOW_CONTROL: ; logical 7
02DB 570 .IF DF RTPAD
02DB 571 BICL #TTSM HOSTSYNC,-
02DB 572 TT_CHAR1(R9) ; Set hostsync
50 82 90 02DB 573 .ENDC
04 50 E9 02DB 574 MOVB (R2)+,R0 ; Get Boolean
10 C8 02DE 575 BLBC R0,10$ ; Branch if not enabled
04 A9 02E1 576 BISL #TTSM HOSTSYNC,-
05 02E3 577 TT_CHAR1(R9) ; Set hostsync
02E5 578 10$: RSB ; Return
02E6 579
50 82 90 02E6 580 LOSS_NOTIFICATION: ; logical 8
00 50 E9 02E6 581 MOVB (R2)+,R0 ; Get Boolean
05 02E9 582 BLBC R0,10$ ; Branch if not enabled
02EC 583 10$: RSB ; Bell on loss data?
02ED 584
02 A9 82 80 02ED 585 LINE_WIDTH: ; logical 9
05 02ED 586 MOVW (R2)+,TT_BUF+2(R9) ; Line width
02F1 587 RSB ; Return
02F2 588
07 A9 82 33 02F2 589 PAGE_LENGTH: ; logical 10
05 02F2 590 CVTWB (R2)+,TT_CHAR1+3(R9) ; Page length
02F6 591 RSB ; Return
02F7 592
05 82 B5 02F7 593 STOP_LENGTH: ; logical 11
05 02F7 594 TSTW (R2)+ ; Ignore for now
02F9 595 RSB ; Return
02FA 596
02FA 597 CR_FILL: ; logical 12
02FA 598 .IF DF RTPAD
02FA 599 BICL #TTSM CRFILL,-
02FA 600 TT_CHAR1(R9)

```



```

04 AA 82 33 02FA 601 .ENDC
08 13 02FA 602 CVTWB (R2)+,TT_IOSB+4(R10)
00000400 8F C8 02FE 603 BEQL 10$ ; Branch if zero
04 A9 05 0300 604 BISL #TTSM CRFILL,-
0306 605 TT_CHAR1(R9)
0308 606 10$: RSB ; Return
0309 607 ; logical 13
0309 608 LF FILL:
0309 609 .IF DF RTPAD
0309 610 BICL #TTSM LFFILL,-
0309 611 TT_CHAR1(R9)
0309 612 .ENDC
05 AA 82 33 0309 613 CVTWB (R2)+,TT_IOSB+5(R10)
08 13 030D 614 BEQL 10$ ; Branch if zero
00000800 8F C8 030F 615 BISL #TTSM LFFILL,-
04 A9 05 0315 616 TT_CHAR1(R9)
0317 617 10$: RSB ; Return
0318 618 ; logical 14
0318 619 WRAP:
0318 620 .IF DF RTPAD
0318 621 BICL #TTSM WRAP,-
0318 622 TT_CHAR1(R9) ; set wrap
0318 623 .ENDC
82 03 B1 0318 624 CMPW #3,(R2)+ ; Get wrap (value 1 to 4)
08 18 031B 625 BGEQ 10$ ; if 3 or less, no wrap
00000200 8F C8 031D 626 BISL #TTSM WRAP,-
04 A9 05 0323 627 TT_CHAR1(R9) ; set wrap
0325 628 10$: RSB ; Return
0326 629 ; logical 15
0326 630 HORIZONTAL TAB:
0326 631 .IF DF RTPAD
0326 632 BICL #TTSM MECHTAB,-
0326 633 TT_CHAR1(R9) ; Set mechtab
0326 634 .ENDC
82 01 B1 0326 635 CMPW #1,(R2)+ ; mechtab?
08 12 0329 636 BNEQ 10$ ; Branch if no
00000100 8F C8 032B 637 BISL #TTSM MECHTAB,-
04 A9 05 0331 638 TT_CHAR1(R9) ; Set mechtab
0333 639 10$: RSB ; Return
0334 640 ; logical 16
82 B5 0334 641 VERTICAL TAB: ; ignore
05 05 0336 642 TSTW (R2)+ ; Return
0337 643 RSB ; logical 17
0337 644 FORM FEED:
0337 645 .IF DF RTPAD
0337 646 BICL #TTSM MECHFORM,-
0337 647 TT_CHAR1(R9) ; Set mechform
0337 648 .ENDC
82 01 B1 0337 649 CMPW #1,(R2)+ ; mech form?
08 12 033A 650 BNEQ 10$ ; Branch if no
00080000 8F C8 033C 651 BISL #TTSM MECHFORM,-
04 A9 05 0342 652 TT_CHAR1(R9) ; Set mechform
0344 653 10$: RSB ; Return
0344 654

```

```

0345 656 ; cterm
0345 657
50 82 90 0345 658 IGNORE_INPUT: ; cterm 1
00 50 E9 0345 659 MOVB (R2)+,R0 ; Get Boolean
05 0348 660 BLBC R0,10$ ; Branch if not enabled
034C 661 10$: RSB ; Return
034C 662
034C 663 CHAR_ATTRIBUTES: ; cterm 2
034C 664
034C 665 .IF DF RTPAD
034C 666
034C 667 MOVAB OUTBAND_NEW,R0 ; Get temporary mask
034C 668 MOVZBL (R2)+,RT ; get character
034C 669 MOVZBL (R2)+,R6 ; Get mask
034C 670 MOVZBL (R2)+,R7 ; Get attribute
034C 671 CLRL R4 ; Clear
034C 672 BBSS R1,R4,5$ ; turn character into mask
034C 673 5$:
034C 674 EXTZV #CTPSV_CH_00,-
034C 675 #CTPSS_CH_00,R6,R8 ; Fetch isolate mask
034C 676 CMPB #CTPSM_CH_00,R8 ; Must be full mask
034C 677 BNEQ 100$ ; exit *** - other things to handle here
034C 678
034C 679 EXTZV #CTPSV_CH_00,-
034C 680 #CTPSS_CH_00,R7,R8 ; Fetch attributes
034C 681 CMPB #CTPSC_CH_CANCEL,R8 ; Cancel?
034C 682 BNEQ 10$ ; br if no
034C 683 BICL R4,OOB_EXCLUDE(R0) ; clear them all...
034C 684 BICL R4,OOB_INCLUDE(R0)
034C 685 BICL R4,OOB_ABORT(R0)
034C 686 BRB 100$
034C 687 10$:
034C 688 CMPB #CTPSC_CH_ICLEAR,R8 ; immediate clear?
034C 689 BEQL 20$ ; br if yes
034C 690 CMPB #CTPSC_CH_DCLEAR,R8 ; deferred clear?
034C 691 BNEQ 30$ ; br if no
034C 692 20$:
034C 693 ;
034C 694 ; set up abort out of band
034C 695 ;
034C 696 BISL R4,OOB_ABORT(R0) ; Set abort flag
034C 697 BRB 100$
034C 698 30$:
034C 699 CMPB #CTPSC_CH_HELLO,R8 ; hello?
034C 700 BNEQ 100$ ; no, (sanity check really)
034C 701 BBC #CTPSV_CH_I,R6,40$ ; Branch if include not specified
034C 702 BBC #CTPSV_CH_I,R7,40$ ; Branch if include not required
034C 703 BISL R4,OOB_INCLUDE(R0) ; Set include bit
034C 704 BRB 100$
034C 705 40$:
034C 706 BISL R4,OOB_EXCLUDE(R0) ; set exclude bit
034C 707 ;
034C 708 ; Handle out-of-band discard (D)
034C 709 ;
034C 710 100$:
034C 711 BBC #CTPSV_CH_D,R6,200$ ; Skip if not in select mask
034C 712 BICL R4,OOB_DISCARD(R0) ; Assume no discard output

```

```

034C 713          BBC      #CTPSV_CH_D,R7,200$      ; Skip if not in select mask
034C 714          BISL     R4,00B_DISCARD(R0)      ; Set discard output
034C 715 200$:
034C 716          ;
034C 717          ; Handle control character echoing      (EE)
034C 718          ;
034C 719          EXTZV    #CTPSV_CH_EE,-
034C 720          #CTPSS_CH_EE,R6,R8              ; Fetch isolate mask
034C 721          CMPB     #3,R8                    ; specified?
034C 722          BNEQ     300$                      ; nope
034C 723          BICL     R4,00B_ECHO(R0)          ; assume no echo (like ^T)
034C 724          EXTZV    #CTPSV_CH_EE,-
034C 725          #CTPSS_CH_EE,R7,R8              ; Fetch real data
034C 726          CMPB     #CTPSC_CH_ECHONONE,R8    ; echo none?
034C 727          BEQL     300$                      ; yes, continue
034C 728          CMPB     #CTPSC_CH_ECHOSTANDARD,R8 ; echo STANDARD?
034C 729          BNEQ     300$                      ; branch if other...
034C 730          BISL     R4,00B_ECHO(R0)          ; set standard echo (like ^C)
034C 731 300$:
034C 732          ;
034C 733          ; *** code not done for:
034C 734          ;         enable/disable special character (F)
034C 735          ;
034C 736          RSB                          ; Exit
034C 737          ;
034C 738 .IFF      ; CTDRIVER
034C 739          ;
52  03  C0 034C 740          ADDL     #3,R2              ; Skip (should never really get here)
05  034F 741          RSB
0350 742          ;
0350 743 .ENDC
0350 744          ;
50  82  90 0350 745 CONTROL_O_PASS_THRU:              ; cterm 3
00  50  E9 0350 746          MOVB     (R2)+,R0          ; Get Boolean
05  0353 747          BLBC     R0,10$                ; Branch if not enabled
0356 748 10$:      RSB                          ; Return
0357 749          ;
0357 750 RAISE INPUT:                                ; cterm 4
0357 751 .IF DF RTPAD
0357 752          BICL     #TTSM_LOWER,-
0357 753          TT_CHAR1(R9)                    ; Set convert lower
0357 754 .ENDC
50  82  90 0357 755          MOVB     (R2)+,R0          ; Get Boolean
08  50  E8 035A 756          BLBS     R0,10$                ; Branch if not enabled
00000080 8F  C8 035D 757          BISL     #TTSM_LOWER,-
04  A9  05 0363 758          TT_CHAR1(R9)                    ; Set convert lower
0365 759 10$:      RSB                          ; Return
0366 760          ;
0366 761 NORMAL_ECHO:                                ; cterm 5
0366 762 .IF DF RTPAD
0366 763          BICL     #TTSM_NOECHO,-
0366 764          TT_CHAR1(R9)                    ; Set noecho
0366 765 .ENDC
50  82  90 0366 766          MOVB     (R2)+,R0          ; Get Boolean
04  50  E8 0369 767          BLBS     R0,10$                ; Branch if enabled (note opposite sense)
04  02  C8 036C 768          BISL     #TTSM_NOECHO,-
04  A9  05 036E 769          TT_CHAR1(R9)                    ; Set noecho

```

```

05 0370 770 10$: RSB ; Return
    0371 771
    0371 772 INPUT_ESCAPE_ENABLE: ; cterm 6
    0371 773 OUTPUT_ESCAPE_ENABLE: ; cterm 7
    0371 774 .IF DF RTPAD
    0371 775 BICL #TTSM_ESCAPE,-
    0371 776 TT_CHAR1(R9)
    0371 777 .ENDC
50 82 90 0371 778 MOVB (R2)+,R0 ; Get Boolean
04 50 E9 0374 779 BLBC R0,10$ ; Branch if not enabled
04 08 C8 0377 780 BISL #TTSM_ESCAPE,-
04 A9 0379 781 TT_CHAR1(R9)
05 037B 782 10$: RSB ; Return
    037C 783
    037C 784 INPUT_COUNT_STATE: ; cterm 8
82 B5 037C 785 TSTW (R2)+ ; Ignore
05 037E 786 RSB
    037F 787
    037F 788 AUTO_PROMPT: ; cterm 9
    037F 789 .IF DF RTPAD
    037F 790 BICL #TTSM_SCRIPT,-
    037F 791 TT_CHAR1(R9)
    037F 792 .ENDC
50 82 90 037F 793 MOVB (R2)+,R0 ; Get Boolean
08 50 E9 0382 794 BLBC R0,10$ ; Branch if not enabled
00000040 8F C8 0385 795 BISL #TTSM_SCRIPT,-
04 A9 038B 796 TT_CHAR1(R9)
05 038D 797 10$: RSB ; Return
    038E 798
    038E 799 ERROR_PROCESSING: ; cterm 10
50 82 90 038E 800 MOVB (R2)+,R0 ; Get Boolean
00 50 E9 0391 801 BLBC R0,10$ ; Branch if not enabled
05 0394 802 10$: RSB ; Return
    0395 803
    0395 804
    0395 805 IGNORE_STRING:
50 82 9A 0395 806 MOVZBL (R2)+,R0 ; Get length of string
52 50 C0 0398 807 ADDL2 R0,R2 ; Add to address
05 039B 808 RSB
    039C 809

```

CTDSENSE
V04-000

- CTDRIVER SENSE MODE PROCESSING J 4
CT_POST_SENSE - Map TSA into VMS

16-SEP-1984 02:24:25 VAX/VMS Macro V04-00
5-SEP-1984 03:14:35 [RTPAD.SRC]CTDSENSE.MAR;1

Page 18
(7)

CTD
V04

```
      039C      811 POST_SENSE_EXIT:
      039C      812
05     039C      813          RSB
      039D      814
      039D      815 .end
```

CTDSENSE
Symbol table

- CTD RIVER SENSE MODE PROCESSING K 4

16-SEP-1984 02:24:25 VAX/VMS Macro V04-00
5-SEP-1984 03:14:35 [RTPAD.SRC]CTDSENSE.MAR;1

Page 19
(7)

CTD
V04

AUTOBAUD DETECT	00000246	R	02	FORM FEED	00000337	R	02
AUTO_PROMPT	0000037F	R	02	GET SPEED	000001F9	R	02
CHSC_CTERM	= 00000002			HORIZONTAL TAB	00000326	R	02
CHSC_CT_AUTO_PROMPT	= 00000009			IGNORE_INPOT	00000345	R	02
CHSC_CT_CTRL0_PASS	= 00000003			IGNORE_STRING	00000395	R	02
CHSC_CT_ERROR_PROCESSING	= 0000000A			INPUT_COUNT_STATE	0000037C	R	02
CHSC_CT_IGNORE_INPUT	= 00000001			INPUT_ESCAPE_ENABLE	00000371	R	02
CHSC_CT_INPUT_COUNT	= 00000008			INPUT_FLOW_CONTROL	000002DB	R	02
CHSC_CT_INPUT_ESC	= 00000006			INPUT_SPEED	000001E5	R	02
CHSC_CT_NORMAL_ECHO	= 00000005			LF FILL	00000309	R	02
CHSC_CT_RAISE_INPUT	= 00000004			LINE WIDTH	000002ED	R	02
CHSC_LG_CR_FILL	= 0000000C			LOGICAL	= 00000100		
CHSC_LG_FLOW_CHAR_PASS	= 00000006			LOSS NOTIFICATION	000002E6	R	02
CHSC_LG_FORM_FEED	= 00000011			MANAGEMENT GUARANTEED	00000251	R	02
CHSC_LG_HOR_TAB	= 0000000F			MODEM PRESENT	00000237	R	02
CHSC_LG_INPOT_FLOW	= 00000007			MODE WRITING_ALLOWED	00000260	R	02
CHSC_LG_LF_FILL	= 0000000D			NORMAL ECHO	00000366	R	02
CHSC_LG_LINE_WIDTH	= 00000009			OUTPUT_ESCAPE_ENABLE	00000371	R	02
CHSC_LG_LOSS_NOTIF	= 00000008			OUTPUT_FLOW_CONTROL	000002BA	R	02
CHSC_LG_MODE_WRITING	= 00000001			OUTPUT_PAGE_STOP	000002C5	R	02
CHSC_LG_OUTPUT_FLOW	= 00000004			OUTPUT_SPEED	000001EF	R	02
CHSC_LG_PAGE_LENGTH	= 0000000A			PAGE LENGTH	000002F2	R	02
CHSC_LG_PAGE_STOP	= 00000005			PARITY_ENABLE	0000021D	R	02
CHSC_LG_STOP_LENGTH	= 0000000B			PARITY_TYPE	00000229	R	02
CHSC_LG_TERM_BITS	= 00000002			PHYSICAL	= 00000000		
CHSC_LG_TERM_TYPE	= 00000003			POST SENSE EXIT	0000039C	R	02
CHSC_LG_VERT_TAB	= 00000010			RAISE_INPUT	00000357	R	02
CHSC_LG_WRAP	= 0000000E			SENSE_BAUD	00000096	R	02
CHSC_LOGICAL	= 00000001			SENSE_ROUTINES	000001E5	R	02
CHSC_PARITY_ODD	= 00000002			SENSE_TABLE	00000000	R	02
CHSC_PHYSICAL	= 00000000			SSS_NORMAL	= 00000001		
CHSC_PH_AUTOBAUD	= 00000007			STOP_LENGTH	000002F7	R	02
CHSC_PH_CHAR_SIZE	= 00000003			SWITCH_CHAR_1	00000258	R	02
CHSC_PH_IN_SPEED	= 00000001			SWITCH_CHAR_2	0000025C	R	02
CHSC_PH_MODEM_PRESENT	= 00000006			TERMINAL_SUBTYPE	00000287	R	02
CHSC_PH_OUT_SPEED	= 00000002			TERMINAL_TYPE	00000267	R	02
CHSC_PH_PARITY_ENABLE	= 00000004			TTSC_BAUD_110	= 00000003		
CHSC_PH_PARITY_TYPE	= 00000005			TTSC_BAUD_1200	= 00000008		
CHARACTER_SIZE	0000020F	R	02	TTSC_BAUD_134	= 00000004		
CHAR_ATTRIBUTES	0000034C	R	02	TTSC_BAUD_150	= 00000005		
CHAR_COUNT	= 0000000B			TTSC_BAUD_1800	= 00000009		
CHAR_TYPE	= 00000200			TTSC_BAUD_19200	= 00000010		
CONTROL_O_PASS_THRU	00000350	R	02	TTSC_BAUD_2000	= 0000000A		
CR_FILL	000002FA	R	02	TTSC_BAUD_2400	= 0000000B		
CT\$AB_TERM_TABLE	000000CA	RG	02	TTSC_BAUD_300	= 00000006		
CTERM	= 00000200			TTSC_BAUD_3600	= 0000000C		
CTPSAB_SENSEBUF	0000015D	RG	02	TTSC_BAUD_4800	= 0000000D		
CTPSC_RT_READ_CHAR	= 0000000A			TTSC_BAUD_50	= 00000001		
CTPSK_SENSEBUF	= 00000042	G		TTSC_BAUD_600	= 00000007		
CTPSV_CH_KNOWN	= 00000000			TTSC_BAUD_7200	= 0000000E		
CTPSV_CH_SCOPE	= 00000001			TTSC_BAUD_75	= 00000002		
CTPSW_CH_PARAM	= 0000002C			TTSC_BAUD_9600	= 0000000F		
CTPSW_MSGSIZE	= 00000028			TTSM_CRFILL	= 00000400		
CT_POST_SENSE	0000019F	RG	02	TTSM_EIGHTBIT	= 00008000		
DC\$ TERM	= 00000042			TTSM_ESCAPE	= 00000008		
ERROR_PROCESSING	0000038E	R	02	TTSM_HOLDSCREEN	= 00004000		
FLOW_CHAR_PASS_THRU	000002D4	R	02	TTSM_HOSTSYNC	= 00000010		

CTDSENSE
Symbol table

- CTDRIVER SENSE MODE PROCESSING L 4

16-SEP-1984 02:24:25 VAX/VMS Macro V04-00
5-SEP-1984 03:14:35 [RTPAD.SRC]CTDSENSE.MAR;1

Page 20
(7)

CTD
V04

```

TISM_LFFILL      = 00000800
TISM_LOWER      = 00000080
TISM_MECHFORM   = 00080000
TISM_MECHTAB    = 00000100
TISM_MODEM      = 00200000
TISM_NOECHO     = 00000002
TISM_ODD        = 00000080
TISM_PARITY     = 00000040
TISM_REMOTE    = 00002000
TISM_SCOPE     = 00001000
TISM_SCRIPT     = 00000040
TISM_TTSYNC    = 00000020
TISM_WRAP      = 00000200
TIS_CA100      = 00000025
TIS_LA12       = 00000024
TIS_LA24       = 00000025
TIS_LA34       = 00000022
TIS_LA36       = 00000020
TIS_LA38       = 00000023
TIS_LQ02       = 00000026
TIS_UNKNOWN    = 00000000
TIS_VK100      = 00000002
TIS_VT05       = 00000001
TIS_VT100      = 00000060
TIS_VT101      = 00000061
TIS_VT102      = 00000062
TIS_VT105      = 00000063
TIS_VT125      = 00000064
TIS_VT131      = 00000065
TIS_VT132      = 00000066
TIS_VT173      = 00000003
TIS_VT200_SERIES = 0000006E
TIS_VT52       = 00000040
TIS_VT55       = 00000041
TISM_AUTOBAUD  = 00000002
TT_BUF        = 00000000
TT_CHAR1      = 00000004
TT_CHAR2      = 00000008
TT_IOSB       = 00000000
UCBSW_ERRCNT  = 00000082
UNKNOWN TT    = 00000155 R 02
VERTICAL_TAB  = 00000334 R 02
WRAP          = 00000318 R 02
    
```

↑-----↑
! Psect synopsis !
↑-----↑

PSECT name	Allocation	PSECT No.	Attributes
. ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$AB\$\$	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$115_DRIVER	0000039D (925.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.06	00:00:00.47
Command processing	141	00:00:00.52	00:00:01.57
Pass 1	420	00:00:10.03	00:00:20.11
Symbol table sort	0	00:00:01.75	00:00:03.42
Pass 2	137	00:00:02.14	00:00:03.20
Symbol table output	20	00:00:00.11	00:00:00.33
Psect synopsis output	1	00:00:00.01	00:00:00.01
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	753	00:00:14.62	00:00:29.11

The working set limit was 1800 pages.
86638 bytes (170 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1619 non-local and 35 local symbols.
815 source lines were read in Pass 1, producing 16 object records in Pass 2.
19 pages of virtual memory were used to define 18 macros.

! Macro library statistics !

Macro library name	Macros defined
-\$255\$DUA28:[SHRLIB]REM.MLB;1	0
-\$255\$DUA28:[RTPAD.OBJ]RTPAD.MLB;1	2
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	1
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	8
TOTALS (all libraries)	11

1655 GETS were required to define 11 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:CTDSENSE/OBJ=OBJ\$:CTDSENSE MSRC\$:CTDSENSE/UPDATE=(ENH\$:CTDSENSE)+EXECMLS/LIB+LIB\$:RTPAD/LIB+SHRLIB\$:REM/LIB

0333 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

A grid of 10 columns and 10 rows of small, faint text screens, likely representing a software installation or configuration process. The screens contain various system messages, prompts, and data. Some screens are more legible than others, showing titles like:

- CTDSENSE LIS
- CTDUMSPEC LIS
- CTSENSERT LIS
- RSTSRT LIS
- CTERMRT LIS
- DTE_DF03 LIS
- CTSETRT LIS
- CTOSET LIS

The text is generally too small to read in detail, but the layout suggests a sequential or parallel installation of software components.