

TTTTTTTTT TTTTTTTTT TT TT TT TT TT TT TT TT TT TT	TTTTTTTTT TTTTTTTTT TT TT TT TT TT TT TT TT TT TT	YY YY YY YY YY YY YY YY YY YY YY	YY YY YY YY YY YY YY YY YY YY YY	SSSSSSSS SSSSSSSS SS SS SS SS SSSSSS SSSSSS SS SS SS SSSSSSSS SSSSSSSS	TTTTTTTTT TTTTTTTTT TT TT TT TT TT TT TT TT TT TT	RRRRRRR RRRRRRR RR RR RR RR RRRRRRR RRRRRRR RR RR RR RR RR RR	SSSSSSSS SSSSSSSS SS SS SS SS SSSSSS SSSSSS SS SS SS SSSSSSSS SSSSSSSS	TTTTTTTTT TTTTTTTTT TT TT TT TT TT TT TT TT TT TT	PPPPPPP PPPPPPP PP PP PP PP PPPPPPP PPPPPPP PP PP PP PP PP PP
--	--	--	--	--	--	--	--	--	--

....
....
....
....

LL LL LL LL LL LL LL LL LL LL LLLLLLLLL LLLLLLLLL	IIIIII IIIIII II II II II II II II II II II IIIIII IIIIII	SSSSSSSS SSSSSSSS SS SS SS SS SSSSSS SSSSSS SS SS SS SS SSSSSSSS SSSSSSSS
--	--	--

(2)	255
(3)	281
(4)	317
(15)	740
(16)	944
(17)	977
(18)	1042
(19)	1172
(22)	1242
(24)	1409

Declarations
TTY\$STARTIO - START I/O OPERATION ON TERMINAL
START IO ACTION ROUTINES
TTY\$WRITSTARTIO - Starts or queues a write operation
TTY\$STARTOUTPUT - START OUTPUT OPERATION ON UNIT
TTY\$GETNXTWRITE - Start next write or restart read
TTY\$WRITEDONE - Complete a write operation
TTY\$WRITEPOST - QUEUE A WRITE COMPLETION
TTY\$READONE - READ OPERATION DONE
End of module

```

0000 1 .TITLE TTYSTRSTP - Terminal driver start/stop I/O routines
0000 2 .IDENT 'V04-000'
0000 3
0000 4
0000 5 *****
0000 6
0000 7 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
0000 8 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
0000 9 * ALL RIGHTS RESERVED. *
0000 10
0000 11 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
0000 12 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
0000 13 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
0000 14 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
0000 15 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
0000 16 * TRANSFERRED. *
0000 17
0000 18 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
0000 19 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
0000 20 * CORPORATION. *
0000 21
0000 22 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
0000 23 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
0000 24
0000 25
0000 26 *****
0000 27
0000 28 **
0000 29 FACILITY:
0000 30
0000 31 VAX/VMS TERMINAL DRIVER
0000 32
0000 33 ABSTRACT:
0000 34
0000 35 THIS MODULE CONTAINS ROUTINES PERTAINING TO STARTING AND COMPLETING
0000 36 I/O REQUESTS.
0000 37
0000 38 AUTHOR:
0000 39
0000 40 R.HEINEN 10-OCT-1977
0000 41
0000 42 Revision history:
0000 43
0000 44 V03-030 MIRO450 MICHAEL I. ROSENBLUM 27-JUN-1984
0000 45 Add code to the free linefeed logic to account for PC_NOCRLF.
0000 46 Fix problem that causes the first linefeed typed on a
0000 47 read with no prompt to not be echoed.
0000 48
0000 49 V03-029 RKS0029 RICK SPITZ 10-APR-1984
0000 50 Enhance virtual terminal connect action routine to
0000 51 perform an implicit set mode operation.
0000 52
0000 53 V03-028 MIRO370 Michael I. Rosenblum 20-Mar-1984
0000 54 Put code in to fix problems with lines and prompts causing
0000 55 wrap. Fix bug that would cause FMS programs to crash the
0000 56 system.
0000 57

```

0000	58	:	V03-027	RKS0027	RICK SPITZ	05-MAR-1984
0000	59	:			Enhance write post completion to handle the case	
0000	60	:			of a write completion with no current PUCB.	
0000	61	:				
0000	62	:	V03-026	MIR0310	Michael I. Rosenblum	09-Feb-1984
0000	63	:			Fix bugs.	
0000	64	:			make sure setting nomodem on a modem terminal shuts down	
0000	65	:			the line.	
0000	66	:				
0000	67	:	V03-025	MIR0300	Michael I. Rosenblum	30-Jan-1984
0000	68	:			Add input fallback	
0000	69	:				
0000	70	:	V03-024	MIR0085	Michael I. Rosenblum	26-Aug-1983
0000	71	:			Remove references to DCL_OUTBND and DCL_CTRLC.	
0000	72	:				
0000	73	:	V03-023	MIR0082	Michael I. Rosenblum	19-Aug-1983
0000	74	:			Make autxoff mode work with passall and ttsync.	
0000	75	:			Fix pas thru to remain enabled after a read completes.	
0000	76	:				
0000	77	:	V03-022	MIR0080	Michael I. Rosenblum	28-Jul-1983
0000	78	:			Move newline code into write done rather than TTYFDT	
0000	79	:			Reposition routines in the module.	
0000	80	:				
0000	81	:	V03-021	MIR0070	Michael I. Rosenblum	13-jul-1983
0000	82	:			Fix bug that would cause TTY\$DISCONNECT to be called twice.	
0000	83	:			if a SETMODE with the HANGUP modifier was issued.	
0000	84	:				
0000	85	:	V03-020	MIR0051	Michael I. Rosenblum	23-Jun-1983
0000	86	:			Fix missing literals in connect and disconnect code.	
0000	87	:			Check write active bit in getnextwrite to insure that	
0000	88	:			The write queue is not reordered.	
0000	89	:				
0000	90	:	V03-019	RKS0019	RICK SPITZ	7-JUN-1983
0000	91	:			ADD CONNECT/DISCONNECT ACTION ROUTINES.	
0000	92	:			ENHANCE WRITE DONE FORK PROCESS TO ALWAYS USE REQCOM	
0000	93	:			IF THE WRITE IRP IS POINTED TO BY UCBSL_IRP	
0000	94	:			MAKE SURE LUCB IS NOT DETACHED AT THE ALTERNATE WRITE ENTRY.	
0000	95	:			REMOVE CTRL Y HANGUP CHECK, AS IT IS STILL DONE IN FDT.	
0000	96	:				
0000	97	:	V03-018	RKS0018	RICK SPITZ	16-MAY-1983
0000	98	:			MOVE SEGMENTS OF CHARACTERISTICS FDT CODE TO TTYSTRSTP	
0000	99	:			TO ALLOW CLEAN DISCONNECT OF DISCONNECTED TERMINALS.	
0000	100	:			RESTORE LUCB FROM LUCB IN READ/WRITE DONE.	
0000	101	:				
0000	102	:	V03-017	MIR0050	Michael I. Rosenblum	11-May-1983
0000	103	:			Remove code that special cased broadcasts. Allow the	
0000	104	:			data returned by timeout errors to be stored in the recall	
0000	105	:			buffer. Make write post complete broadcasts.	
0000	106	:				
0000	107	:	V03-016	MIR0030	Michael I. Rosenblum	30-Mar-1983
0000	108	:			Integrate Read verification with the standard driver	
0000	109	:			Add support for alternate frame sizes.	
0000	110	:				
0000	111	:	V03-015	MIR0029	Michael I. Rosenblum	22-Mar-1983
0000	112	:			Add field to the iosb when itemlist reads are used.	
0000	113	:				
0000	114	:	V03-014	RKS0014	RICK SPITZ	14-MAR-1983

```

0000 115 :
0000 116 :
0000 117 :
0000 118 :
0000 119 :
0000 120 :
0000 121 :
0000 122 :
0000 123 :
0000 124 :
0000 125 :
0000 126 :
0000 127 :
0000 128 :
0000 129 :
0000 130 :
0000 131 :
0000 132 :
0000 133 :
0000 134 :
0000 135 :
0000 136 :
0000 137 :
0000 138 :
0000 139 :
0000 140 :
0000 141 :
0000 142 :
0000 143 :
0000 144 :
0000 145 :
0000 146 :
0000 147 :
0000 148 :
0000 149 :
0000 150 :
0000 151 :
0000 152 :
0000 153 :
0000 154 :
0000 155 :
0000 156 :
0000 157 :
0000 158 :
0000 159 :
0000 160 :
0000 161 :
0000 162 :
0000 163 :
0000 164 :
0000 165 :
0000 166 :
0000 167 :
0000 168 :
0000 169 :
0000 170 :
0000 171 :

```

ADD SUPPORT FOR LOGICAL UCB. NOTE THAT THE DRIVER SWITCHES TO PHYSICAL UCB CONTEXT AT STARTIO ENTRY. IT RESTORES LOGICAL UCB CONTEXT PRIOR TO RETURNING TO THE SYSTEM.

V03-013 MIR8026 Michael I. Rosenblum 14-Mar-1983
Fix bug in partail escape sequence processing.

V03-012 MIR5026 Michael I. Rosenblum 10-Mar-1983
Fix security whole with command recall and the password by not allowing noecho strings to be stored in the recall buffer.

V03-011 MIR1024 Michael I. Rosenblum 09-Mar-1983
Fix code in getnxtwrite to look at the read packet rather than UCBSW_BCNT to find the number of characters that have been read so far.

V03-010 MIR0026 Michael I. Rosenblum 01-Mar-1983
Add code to save the results of the last read.

V03-009 MIR0024 Michael I. Rosenblum 28-Jan-1983
Update code to use the new read packet format

V03-008 MIR0023 Michael I. Rosenblum 24-Jan-1983
Read buffer was used after it was deallocated if a Cancel was issued while EDITREAD state was in affect. Changed READONE code to clear the edit read states when a read is completed.

V03-007 MIR0016 Michael I. Rosenblum 4-Jan-1983
Change TTY\$STARTOUTPUT to use the UCB OUTYPE field to determine the necessary action when TTY\$GETNEXTCHAR is called. This change should illiminate the checking the volitale condition code bits that previously had the function of OUTYPE. For compatibility purposes only we are setting the correct condition codes.

V03-006 MIR0015 Michael I. Rosenblum 20-Dec-1982
Change TTY\$V_ST_UNSQL and TTY\$V_ST_GETAHD to TTY\$V_FD_UNSQL and TTY\$V_FD_GETAHD, to reflect changes in the fork dispatcher also change PORT_DISCONNECT call to refer to CLASS_MODEM_DIS. Change all port calls to call the Class driver jacket routines.

V03-005 MIR0013 Michael I. Rosenblum 16-Dec-1982
Fix up references to new ucb structure

V03-004 MIR0011 Michael I. Rosenblum 18-Nov-1982
Remove code that implimented HOLDSCREEN.

V03-003 MIR0010 Michael I. Rosenblum 09-Nov-1982
Move the address of the terminator mask, and the length of the prompt string from the IRP into the terminal read buffer.

V03-002 ROW0077 Ralph O. Weber 27-MAR-1982
Change TTY\$WRITEDONE to insure that eventhough UCBSW_IT_CURSOR

```

0000 172 : can now be bigger than UCBSW DEVBUFSIZ, i.e. eventhough our
0000 173 : internal cursor position marker can virtually be beyond the
0000 174 : right-hand edge of the screen, the cursor-position value
0000 175 : returned in IOSB will never exceed the width of the screen.
0000 176 :
0000 177 : V03-001 JLV0202 Jake VanNoy 23-MAR-1982
0000 178 : Change MODHANGUP from NOMOD to PRIV_TO_MOD in Set
0000 179 : Mode/Char logic.
0000 180 : Correct alternate class name lookup.
0000 181 :
0000 182 : V02-045 RKS0045 RICK SPITZ 22-FEB-1982
0000 183 : Repair diagnostic function code logic.
0000 184 :
0000 185 : V02-044 RKS0044 RICK SPITZ 16-FEB-1982
0000 186 : Enhance broadcast logic to allow delay prior to
0000 187 : forcing output. Move setting of controls pending
0000 188 : to STOP2 timeout. This way user ^s can be distinguished
0000 189 : from terminal xoff.
0000 190 : Save R3 prior to forking to create typeahead on read.
0000 191 :
0000 192 : V02-043 RKS0043 RICK SPITZ 11-FEB-1982
0000 193 : Zero fork byte in TWP to allow DMA of broadcast.
0000 194 : Prevent XON characteristic from being permantly set.
0000 195 :
0000 196 : V02-042 RKS0042 Rick Spitz 8-FEB-1982
0000 197 : Repair Alternate typeahead logic to allow setting
0000 198 : Permanent from users terminal.
0000 199 : Allocate typeahead buffer when starting read, if not already
0000 200 : done. This is needed for lines which are used for communications
0000 201 : on DMF-32 async lines.
0000 202 :
0000 203 : V02-041 ROW0066 Ralph O. Weber 31-JAN-1982
0000 204 : Enhance alternate class driver setup to relocate address in
0000 205 : alternate class driver vector table. Correct use of
0000 206 : unrelateable .ASCID directive.
0000 207 :
0000 208 : V02-040 RKS0040 RICK SPITZ 24-JAN-1982
0000 209 : USE INPUT VALUE FOR READ FIELD OFFSET.
0000 210 : ADD LOGIC TO BIND TO ALTERNATE DRIVER.
0000 211 :
0000 212 : V02-039 RKS0039 RICK SPITZ 15-DEC-1981
0000 213 : FIX MAINTENANCE DISPATCH LOGIC.
0000 214 : DISALLOW SETTING ALT TYPEAHEAD IF ONE ALREAY EXISTS.
0000 215 : REMOVE LOGIO REQUIREMENT FOR PARITY ENABLE.
0000 216 : FIX WRTSTARTIO RETURN ADDRESSING.
0000 217 : ALLOW NOECHO READ TO NOT BLOCK WRITES.
0000 218 : ADD WRITE POST ROUTINE TO REPLACE INSPOST LOGIC, THIS
0000 219 : CORRECTS RACE CONDITION IN HALF DUPLEX WRITE COMPLETIONS.
0000 220 : ADD SUPPORT FOR ALTERNATE CLASS DRIVER.
0000 221 :
0000 222 : V02-038 JLV0126 Jake VanNoy 1-Dec-1981
0000 223 : Add local echo logic and set speed privilege checking.
0000 224 :
0000 225 : V02-037 JLV0102 Jake VanNoy 27-Oct-1981
0000 226 : Changed TTYDEFS to $TTYDEFS.
0000 227 :
0000 228 : V02-036 JLV0070 Jake VanNoy 28-Aug-1981

```

```
0000 229 : Added UCBSL_TT_DEVP1 checking and no refresh on broadcast.
0000 230 :
0000 231 : V02-035 RKS035 RICK SPITZ 26-AUG-1981
0000 232 : ADD MAINT ENABLE BIT
0000 233 :
0000 234 : V02-034 RKS034 RICK SPITZ 20-AUG-1981
0000 235 : ADD SUPPORT FOR ESCAPE MODIFIER ON READ.
0000 236 :
0000 237 : V02-033 RKS033 RICK SPITZ 12-AUG-1981
0000 238 : RESET DMA ABORT STATE IN WRITE DONE LOGIC.
0000 239 : RESET CONTROLS STATE FOR MAINTENANCE FUNCTIONS.
0000 240 :
0000 241 : V02-032 RKS032 RICK SPITZ 27-JULY-1981
0000 242 : SEVERAL ENHANCEMENTS HAVE BEEN ADDED TO SUPPORT REVISIONS
0000 243 : TO THE UCB STRUCTURE INCLUDING SPLIT SPEED, AND QUADWORD STATE
0000 244 : AND DEVDEPEND FIELDS.
0000 245 : SUPPORT FOR DIAGNOSTIC FUNCTIONS AND ENHANCED MODEM PROCESSING
0000 246 : HAS BEEN ADDED. SEVERAL CHANGES TO SUPPORT THE CLASS/PORT
0000 247 : STRUCTURE AS WELL AS ENHANCEMENTS TO ALLOW TERMINAL
0000 248 : INITIATED CONTROL S AND Q DURING BROADCAST HAVE ALSO BEEN
0000 249 : ADDED.
0000 250 :
0000 251 : V02-031 RKS031 RICK SPITZ 26-FEB-1981
0000 252 : REMOVE V2.0 AUDIT TRAILS
0000 253 :
```



```
0000 255          .SBTTL Declarations
0000 256
0000 257 :
0000 258 : EXTERNAL SYMBOLS
0000 259 :
0000 260 : $ARBDEF          : DEFINE ACCESS RIGHTS BLOCK
0000 261 : $CADEF           : DEFINE CONDITIONAL ASSEMBLY PARAMETERS.
0000 262 : $CRBDEF         : DEFINE CRB
0000 263 : $DPTDEF        : DEFINE DPT OFFSETS
0000 264 : $IODEF         : DEFINE I/O FUNCTION CODES
0000 265 : $IPLDEF        : DEFINE IPL'S
0000 266 : $IRPDEF        : DEFINE IRP
0000 267 : $PRDEF         : DEFINE PROCESSOR REGISTERS
0000 268 : $PRVDEF        : DEFINE PRIVILEGE MASK BITS
0000 269 : $SSDEF         : Define system status codes
0000 270 : $TTDEF         : DEFINE TERMINAL CHARACTERISTICS
0000 271 : $TT2DEF        : DEFINE TERMINAL CHARACTERISTICS
0000 272 : $TTYDEF        : DEFINE TERMINAL DRIVER SYMBOLS
0000 273 : $UCBDEF        : DEFINE UCB
0000 274 : $VECDEF        : DEFINE CRB VECTOR OFFSETS
0000 275 : $TTYMACS       : DEFINE TERMINAL MACROS
0000 276 : $TTYDEFS       : DEFINE TERMINAL DEFINITIONS
0000 277 : $TTYMODEM      : DEFINE TERMINAL MODEM DEFINITIONS
00000000 278
00000000 279          .PSECT $$$115_DRIVER, LONG : DEFINE NON-PAGED PSECT
```

```

0000 281 .SBTTL TTY$STARTIO - START I/O OPERATION ON TERMINAL
0000 282 :++
0000 283 : TTY$STARTIO - START I/O OPERATION ON TERMINAL
0000 284 :
0000 285 : FUNCTIONAL DESCRIPTION:
0000 286 :
0000 287 : THIS ROUTINE IS ENTERED WHEN THE UNIT IS IDLE AND THERE IS A PACKET TO PROCESS.
0000 288 :
0000 289 : INPUTS:
0000 290 :
0000 291 : I/O PACKET FORMATTED AS DESCRIBED IN TTYFDT.
0000 292 :
0000 293 : R3 = I/O PACKET ADDRESS
0000 294 : R5 = LOGICAL UCB ADDRESS
0000 295 :
0000 296 : OUTPUTS:
0000 297 :
0000 298 : NONE
0000 299 :--
0000 300 TTY$STARTIO:: ; START TERMINAL I/O
0000 301 :
0000 302 : MOVE TO PHYSICAL UCB CONTEXT. THIS INVOLVES DUPLICATING
0000 303 : MANIPULATIONS TO THE LOGICAL UCB DONE BY IOCSINITIATE
0000 304 :
150 00A0 C5 D0 0000 305 MOVL UCBSL_TL_PHYUCB(R5),R0 ; GET PHYSICAL UCB ADDRESS
58 A0 53 D0 0005 306 MOVL R3,UCBSL_IRP(R0) ; COPY IRP ADDRESS TO PHYS UCB
78 A0 2C A3 7D 0009 307 MOVQ IRP$S_SVAPTE(R3),UCBSL_SVAPTE(R0)
64 A0 0048 8F AA 000E 308 BICW #UCBSM_CANCEL!UCBSM_TIMEOUT,UCBSW_STS(R0)
55 50 D0 0014 309 MOVL R0,R5 ; SWITCH TO PHYSICAL UCB
0017 310
54 20 A3 FFE6' 30 0017 311 BSBW TTY$LOCK ; SETUP IPL AND REGISTERS
FFCO 8F AB 001A 312 BICW3 #^C<IOSM_FCODE>,IRPSW_FUNC(R3),R4; GET INTERNAL FUNCTION CODE
0021 313 CASE R4,TYPE=B,<DO_READ,DO_WRITE,DO_SETM,DO_SETC,DO_HANGUP,-
0021 314 DO_MAINT,DO_HANGUP,DO_CONNECT,DO_DISCONNECT>
0037 315
  
```

```

0037 317 .sbttl START_IO ACTION ROUTINES
0037 318
0037 319 ; CONNECT THIS PUCB TO A DETACHED LUCB
0037 320
0037 321 DO_CONNECT:
0037 322 BBC #IOSV TT DISCON,-
0039 323 IRPSW_FUNC(R3),10$ ;SKIP UNLESS DISCONNECT SPECIFIED
48 A5 08 20 A3 E1 0037 324 10$: BICL #TT2SM_DISCONNECT,UCBSL_DEVDEPND2(R5); FORCE HANGUP TO COMMAND PROCE
003C 325
51 0084 C5 D0 0044 326
51 13 0044 327
004B 328
004B 329
004F 330
54 0A BB 004F 331 PUSHR #^M<R1,R3> ; SAVE IRP ADDRESS AND LUCB
02 DO 0051 332 MOVL #TTYSV_FD_DISCONNECT,R4 ; SCHEDULE DISCONNECT COMMAND PUCB
FFA9 30 0054 333 BSBW TT$CRE_FORK
0A BA 0057 334 POPR #^M<R1,R3> ; SAVE IRP ADDRESS AND LUCB
0059 335
0059 336 ; NOW CONVERT IRP INTO SET MODE AND FILL IN VALUES FROM
0059 337 ; THE TARGET LUCB
0059 338
40 A3 44 A1 7D 0059 339 MOVB UCBSL_DEVDEPEND(R1),IRPSQ_TT_STATE(R3)
39 A3 41 A1 90 005E 340 MOVW UCBSB_DEVTYPE(R1),IRPSL_MEDIA+1(R3) ; TERMINAL TYPE
3A A3 42 A1 B0 0063 341 MOVW UCBSW_DEVBUFSIZ(R1),IRPSL_MEDIA+2(R3) ; WIDTH
4C A3 D4 0068 342 CLRL IRPSL_TT_PRMPRT(R3)
009C C3 D4 006B 343 CLRL IRPSL_VAC5(R3)
006F 344
40 A3 00200000 8F CA 006F 345 BICL #TTSM_MODEM,IRPSQ_TT_STATE(R3) ; TRACK MODEM TO BE SAME AS
08 44 A5 15 E1 0077 346 BBC #TT$V_MODEM,UCBSL_DEVDEPEND(R5),20$ ;
40 A3 00200000 8F C8 007C 347 BISL #TTSM_MODEM,IRPSQ_TT_STATE(R3) ;
0084 348 20$:
44 A3 00020000 8F CA 0084 349 BICL #TT2SM_DISCONNECT,IRPSQ_TT_STATE+4(R3) ; TRACK DISCONNECT TO BE SAM
08 48 A5 11 E1 008C 350 BBC #TT2SV_DISCONNECT,UCBSL_DEVDEPND2(R5),22$;
44 A3 00020000 8F C8 0091 351 BISL #TT2SM_DISCONNECT,IRPSQ_TT_STATE+4(R3) ;
0099 352 22$:
0099 353
0099 354
0125 31 0099 355 BRW DO_SET ; NOW INVOKE SET MODE ACTION ROUTINE
009C 356
50 0908 8F 3C 009C 357 25$: MOVZWL #SS$ NOSUCHDEV,R0 ; INDICATE DEVICE NOT AVAILABLE
0691 31 00A1 358 BRW TTYS$DONE

```

```
00A4 360
00A4 361
00A4 362 : DISCONNECT COMMAND LUCB FROM PUCB.
00A4 363 : IF NOT DETACHED, HANGUP SIGNALLED TO COMMAND PROCESS
00A4 364
00A4 365 DO_DISCONNECT:
54 53 DD 00A4 366 PUSHL R3 ; SAVE IRP
02 DO 00A6 367 MOVL #TTY$V_FD_DISCONNECT,R4 ; SCHEDULE DISCONNECT ON THAT PUCB
FF54' 30 00A9 368 BSBW TTY$CRE_FORK
53 8ED0 00AC 369 POPL R3 ; RESTORE IRP
50 01 3C 00AF 370 MOVZWL #SS$ NORMAL,R0
0680 31 00B2 371 BRW TTY$DONE
```

```
00B5 373
00B5 374
00B5 375 ; PROCESS HANGUP FUNCTION. THIS ROUTINE FORCES A MODEM HANGUP
00B5 376
00B5 377 DO_HANGUP:
38 BB 00B5 378 PUSHR #*M<R3,R4,R5> ; SAVE REGISTERS
FF46' 30 00B7 379 BSBW CLASS MODEM DIS ; DISCONNECT UNIT
38 BA 00BA 380 POPR #*M<R3,R4,R5> ; RESTORE REGITERS
0363 31 00BC 381 BRW DO_EXIT
00BF 382
```

```

00BF 384
00BF 385 ; PROCESS MAINTENANCE FUNCTIONS
00BF 386
00BF 387 DO_MAINT:
54 41 44 A5 E0 00BF 388 BBS #TTSV_MODEM,- ; DISALLOW IF MODEM LINE
0118 C5 D0 00C1 389 UCBSL_DEVDEPEND(R5),30$
24 20 04 EC 00C4 390 MOVL UCBSL_TT_PORT(R5),R4 ; ACCESS PORT VECTOR
06 07 EF 00C9 391 BBS #IOSV_SET_MODEM,- ; BRANCH IF SET MODEM FUNCTION
50 20 A3 00CB 392 IRPSW_FUNC(R3),20$
012A C5 50 88 00CE 393 EXTZV #IOSV_LOOP,#<IOSV_LOOP_EXT-IOSV_LOOP+1>,-
50 50 88 00D1 394 IRPSW_FUNC(R3),R0 ; GET MAINT SUBMODIFIERS
012A C5 50 D4 00D4 395 BISB R0,UCBSB_TT_MAINT(R5) ; PASS TO PORT
FF22' 30 00D9 396 CLRL R0 ; ASSUME ERROR, FOR NULL POST ROUTIN
7F 8F 8A 00DB 397 BSBW TTYSMAINT ; INVOKE PORT DRIVER TO DO FUNCTION
012A C5 00E1 398 BICB #^C<UCBSM_TT_DSBL>,- ; RESET ALL BUT DISABLE
1E 50 E9 00E4 400 BLBC R0,30$ ; FAILURE
50 0118 C5 D0 00E7 401 MOVL UCBSL_TT_PORT(R5),R0 ; GET PORT VECTOR ADDRESS
FF11' 30 00EC 402 BSBW TTYSRESUME ; RESET ANY CONTROLS STATE
0330 31 00EF 403 BRW DO_EXIT ; SUCCESS
52 3A A3 3C 00F2 404 20$: MOVZWL IRPSL_MEDIA+2(R3),R2 ; PROCESS SET MODEM SIGNALS
52 E5 8F 8A 00F6 406 BICB #^C<TTSM_DS_DTR!- ; GET SET/RESET MODEM MASK
00FA 407 TTSM_DS_SECTX!- ; CLEAR ALL BUT MODEM OUTPUT
00FA 408 TTSM_DS_RTS>,R2 ; BITS
53 DD 00FA 409 PUSHL R3 ; SAVE VOLITAL REGISTER
FF01' 30 00FC 410 BSBW TTYSDS_SET ; SET /RESET SPECIFIED SIGNALS
53 8ED0 00FF 411 POPL R3 ; RESTORE REGISTER
031D 31 0102 412 BRW DO_EXIT ; SUCCESS EXIT
50 2C 3C 0105 413 30$: MOVZWL #SS$ _ABORT,R0 ; ERROR EXIT
51 D4 0108 414 CLRL R1
0628 31 010A 416 BRW TTYSDONE
  
```

```

010D 418 :
010D 419 : READ OPERATION
010D 420 :
010D 421 DO_READ:
54 78 A5 D0 010D 422      MOVL   UCBSL_SVAPTE(R5),R4      ; GET THE ADDRESS OF THE READ PACKET
0111 423
04 62 40 A3 C8 0111 424      BISL   IRPSQ_TT_STATE(R3), (R2) ; Set the read state bits.
04 A2 44 A3 C8 0115 425      BISL   IRPSQ_TT_STATE+4(R3),4(R2)
011A 426
38 A3 D4 011A 427      CLRL   IRPSL_MEDIA(R3) ; Set up storage for the read
011D 428 ; terminator.
04 AA 011D 429      BICW   #UCBSM_TT_NOTIF,- ; Set the 'user has not been
68 A5 011F 430      UCBSW_DEVSTS(R5) ; notified' bit.
0B E1 0121 431      BBC    #IOSV_PURGE,- ; Branch forward if purge type-
03 20 A3 0123 432      IRPSW_FUNC(R3),10$ ; ahead not requested.
FED7' 30 0126 433      BSBW   TTY$PURGE_AHEAD ; Otherwise, purge buffer.
0129 434 ; the write completes.
0129 435 10$:
00E4 C5 D5 0129 436      TSTL   UCBSL_TT_TYPAHD(R5) ; Type ahead buffer allocated?
73 13 012D 437      BEQL   30$ ; Not yet
012F 438
FECE' 30 012F 439 12$: BSBW   TTY$SETUP_READ ; Set up the UCB for a read
0132 440 ; operation.
0132 441
0132 442 :
0132 443 : CHECK FOR LINE FEED NEEDED
0132 444 :
0132 445 :
0132 446      IF STATE - ; Skip if passall, or
0132 447      <PASALL>,25$
0136 448      IF NOT STATE NOECHO,14$ ; NO ECHO THEN
013A 449      CLR STATE EDITING ; NO EDITING
013E 450 14$: IF STATE -
013E 451      <NC,WRAP>,20$ ; if already did line feed.
0145 452
0145 453      IF NOT STATE NOECHO,15$ ; Branch if echo
26 48 A5 E1 0149 454      BBC    #TT2$V_LOCALECHO,- ; Branch if not local echo
014B 455      UCBSL_DEVDEPND2(R5),20$
014E 456
00FC C5 B5 014E 457 15$: TSTW   UCBSW_TT_CURSOR(R5) ; CURSOR AT 0?
20 12 0152 458      BNEQ   20$ ; If no, send no line feed.
00FF C5 91 0154 459      CMPB   UCBSB_TT_LASTC(R5),- ; Was the last character also a
0D 0158 460      #TTY$C_CR ; carriage return?
19 12 0159 461      BNEQ   20$ ; No. Don't send free linefeed.
015B 462      IF NOT STATE SKIPLF,17$ ; NO SKIP LINEFEED THEN BYPASS NOCRLF CHECK
OF 0122 C5 07 E0 015F 463      BBS    #TTY$V_PC_NOCRLF,UCBSW_TT_PRTCTL(R5),20$;SO JUST ECHO THE CHARACTER
0165 464 17$: SET_STATE <SENDF$ ; SEND A LINE FEED FIRST
0168 465      CLR_STATE <SKIPLF>
016C 466      IF NOT STATE PROMPT,20$ ; DO WE HAVE A
0170 467      SET_STATE <SKIPLF> ; SEND A LINE FEED FIRST
0174 468 20$: IF_STATE RDVERIFY,25$ ; THIS ISN'T NECESSARY IF READ VERIFY
0178 469
54 78 A5 D0 0178 470      MOVL   UCBSL_SVAPTE(R5),R4 ; GET THE READ PACKET ADDRESS
7E 00FC C5 3C 017C 471      MOVZWL UCBSW_TT_CURSOR(R5),-(SP); SAVE THE CURSOR POSITION FOR ECHOING
02 44 A4 B1 0181 472      CMPW   TTY$W_RB_MODE(R4),#TTY$K_ER_ECHLINE; IS THIS A READ WITH INITIAL
0185 473 ; OFFSET.
0D 12 0185 474      BNEQ   21$ ; NO THEN USE NORMAL

```

```

00FC C5 012B C5 9B 0187 475
          018E 476
3A A4 012B C5 9B 018E 477
          32 A4 B4 0194 478 21$:
          FE66' 30 0197 479
00FC C5 8E F7 019A 480
          0390 31 019F 481 25$:
          01A2 482 30$:
          01A2 483
          01A2 484
          53 DD 01A2 485
          53 D4 01A4 486
54 01 D0 01AC 487
          FE54' 30 01A9 488
          53 BF D0 01AC 489
          FF7D 31 01AF 490
          01B2 491
  
```

```

MOVZBW UCBSB_TT_OLDPZORG(R5),UCBSW_TT_CURSOR(R5); YES THEN USE THE
          ; STORED INITIAL CURSOR POSITION
MOVZBW UCBSB_TT_OLDPZORG(R5),TTY$W_RB_CPZORG(R4);
CLRW TTY$W_RB_LINREST(R4); NO EXTRA CHARACTERS
BSBW FIND BOL NOCLEAR; find the wrapping and
RVTLW (SP)T,UCBSW_TT_CURSOR(R5); RESTORE THE CURSOR POSITION FOR ECHOING
BRW TTY$STARTOUTPUT; Go start the read.

; Allocate typeahead buffer

PUSHL R3; SAVE IRP ADDRESS
CLRL R3; INDICATE NO DATA
MOVL #TTY$V_FD_GETAHD,R4; ASK FOR TYPEAHD FORK
BSBW TTY$CRE_FORK; GO ALLOCATE BUFFER
POPL R3; RESTORE IRP ADDRESS
BRW 12$; CONTINUE PROCESSING
  
```



```

01B2 493
01B2 494 ; SET MODE OPERATION -
01B2 495
01B2 496 DO_SETM:
000C 31 01B2 497 BRW DO_SET
01B5 498
01B5 499 ; DO SET CHARACTERISTICS
01B5 500
01B5 501 DO_SETC:
32 A3 0C B1 01B5 502 CMPW #12,IRPSW_BCNT(R3) ; DO PRIVILEGED SET
15 01B9 503 BLEQ DO_SET ; CHECK PARAMETERS
00C8 C5 DO 01BB 504 MOVL UCBSL_TT_DECHA1(R5),- ; ALL SPECIFIED
44 A3 01BF 505 IRPSQ_TT_STATE+4(R3) ; INIT DEFAULT IF NOT SPECIFIED
01C1 506
01C1 507 ; PROCESS CHANGE OF CHARACTERISTICS AND MODE
01C1 508 ; CHANGE BASIC MODE BITS IN UCBSL_DEVDEPEND
01C1 509
01C1 510 DO_SET:
54 44 A5 40 A3 CD 01C1 511 XORL3 IRPSQ_TT_STATE(R3),UCBSL_DEVDEPEND(R5),R4; GET MODIFIED BITS
08 54 0D E5 01C7 512 BBCC #TT$V_REMOTE,R4,8$ ; DISALLOW CLEARING REMOTE BIT
40 A3 00002000 8F CA 01CB 513 BICL #TT$M_REMOTE,IRPSQ_TT_STATE(R3)
01D3 514 ; DISALLOW SETTING REMOTE BIT
01D3 515 8$:
44 A5 54 CA 01D3 516 BICL R4,UCBSL_DEVDEPEND(R5) ; CLEAR THE CHANGED BITS
44 A5 40 A3 C8 01D7 517 BISL IRPSQ_TT_STATE(R3),UCBSL_DEVDEPEND(R5);
41 A5 39 A3 90 01DC 518 MOVB IRPSL_MEDIA+1(R3),UCBSB_DEVTYPE(R5); INSERT NEW TERMINAL TYPE
01E1 519
01E1 520 :
01E1 521 :
50 44 A3 DO 01E1 522 MOVL IRPSQ_TT_STATE+4(R3),R0 ; GET SECOND DEVDEPEND WORD
51 48 A5 50 CD 01E5 523 XORL3 R0,UCBSL_DEVDEPN2(R5),R1 ; GET MODIFIED BITS
50 00000200 8F CA 01EA 524 BICL #<TT2$M_DCL_MAILBX>,R0 ; REMOVE DCL SPECIFIC BITS
01F1 525
24 51 06 E1 01F1 526 BBC #TT2$V_DMA,R1,12$ ; SKIP IF DMA NOT CHANGED
14 50 06 E1 01F5 527 BBC #TT2$V_DMA,R0,10$ ; BRANCH IF TURNING DMA OFF
OE 0122 C5 02 E1 01F9 528 BBC #TTY$V_PC_DMAAVL,UCBSW_TT_PRTCTL(R5),10$ ; DONT IF FEATURE NOT AVAIL
50 00000040 8F C8 01FF 529 BISL #TT2$M_DMA,R0 ; SET DMA CHARACTERISTIC ON
0122 C5 02 AB 0206 530 BISW #TTY$M_PC_DMAENA,UCBSW_TT_PRTCTL(R5) ; ENABLE IN PORT
11 020B 531 BRB 12$
50 0122 C5 02 AA 020D 532 10$: BICW #TTY$M_PC_DMAENA,UCBSW_TT_PRTCTL(R5) ; DISABLE DMA IN PORT
50 00000040 8F CA 0212 533 BICL #TT2$M_DMA,R0 ; RESET DMA CHARACTERISTIC
0219 534
0219 535 12$: NOMOD ALTYPEAHD ; DISALLOW CHANGING TYPE AHEAD
0224 536
11 48 A5 E0 0224 537 BBS #TT2$V_MODHANGUP,-
0226 538 UCBSL_DEVDEPN2(R5),15$ ; BRANCH IF MODIFY HANGUP ALLOWED
0229 539
0229 540 PRIV_TO_MOD HANGUP ; REQUIRE PRIV TO MODIFY HANGUP
023A 541 15$: PRIV_TO_MOD SETSPEED ; REQUIRE PRIV TO MODIFY SET SPEED
024B 542 PRIV_TO_MOD SECURE ; REQUIRE PRIVS TO MODIFY SECURE SERVER
025C 543 PRIV_TO_MOD MODHANGUP ; REQUIRE PRIV TO MODIFY MODHANGUP
026D 544
06 50 05 E1 026D 545 BBC #TT2$V_XON,R0,20$ ; BRANCH IF NO XON REQUESTED
50 20 CA 0271 546 BICL #TT2$M_XON,R0 ; RESET XON BIT.
FD89' 30 0274 547 BSBW TTY$RESUME ; CALL RESUME
0277 548
0277 549 20$:

```

```

48 A5 50 D0 0277 550      MOVL   R0,UCBSL_DEVDEPND2(R5) ; SET SECOND DEVDEPENDENT WORD
          0278 551      :
          027B 552      : SET UP WIDTH
          027B 553      :
42 A5 3A A3 B0 027B 554      MOVW   IRPSL_MEDIA+2(R3),UCBSW_DEVBUFSIZ(R5); INSERT NEW CARRIAGE WIDTH
          0280 555      :
          0280 556      : SET UP SPEED
          0280 557      :
51 4C A3 3C 0280 558      MOVZWL  IRPSW_TT_PRMPRT(R3),R1 ; GET NEW SPEED
          1E 13 0284 559      BEQL   30$ ; IF EQL THEN NO CHANGE
          0286 560      :
          0286 561      : SET SPEED PRIVILEGE CHECK
          0286 562      :
00F4 C5 51 91 0286 563      CMPB   R1,UCBSW_TT_SPEED(R5) ; IS LOW ORDER BYTE OF SPEED CHANGING?
          12 13 028B 564      BEQL   28$ ; BRANCH IF NOT
          028D 565      :
          08 E1 028D 566      BBC    #TT2$V_SETSPEED,-
58 B3 0D 48 A5 028F 567      UCBSL_DEVDEPND2(R5),28$ ; BRANCH IF SET SPEED ALLOWED
          00400080 8F D3 0292 568      BITL   #<<1@PRV$V LOG IO>! - ; DOES PROCESS HAVE LOG_IO
          029A 569      <1@PRV$V PHY IO>>, - ; OR PHY IO PRIVILEGE?
          029A 570      @IRPSL_ARB(R3) ; CHECK ACCESS RIGHTS BLOCK
          03 12 029A 571      BNEQ   28$ ; BRANCH IF PRIVILEGED
          019D 31 029C 572      BRW    NOPRIV_EXIT ; PRIV FAILURE
          029F 573      :
          029F 574      :
          029F 575      : PROCESS PARITY SETTINGS
          029F 576      :
          00F4 C5 51 B0 029F 577 28$: MOVW   R1,UCBSW_TT_SPEED(R5) ; INSERT LINE SPEED
          12 009C C3 05 E1 02A4 578 30$: BBC    #TT$V_ALTRPAR,IRPSL_VAL5(R3),35$; BR IF PARITY SHOULD NOT BE ALTERED
50 009C C3 3F 8F 8B 02AA 579      BICB3  #^C<TT$M PARITY!TT$M ODD>,IRPSL_VAL5(R3),R0; RESET BITS
          00F8 C5 C0 8F 8A 02B1 580      BICB   #TT$M PARITY!TT$M ODD,UCBSB_TT_PARITY(R5); CLEAR CURRENT PARITY
          00F8 C5 50 88 02B7 581      BISB   R0,UCBSB_TT_PARITY(R5) ; INSERT NEW VALUE
          02BC 582      :
          02BC 583      : SET UP CHARACTER SIZE AND STOP BITS
          02BC 584      :
          02BC 585 35$:
          08 009C C3 04 E0 02BC 586      BBS    #TT$V_ALTFRAME,IRPSL_VAL5(R3),36$; DOES THE USER WANT A NEW FRPAM SI
          21 00F8 C5 02 E1 02C2 587      BBC    #UCBSV_TT_USERFRAME,UCBSB_TT_PARITY(R5),37$; DID THE USER SPECIFY
          02C8 588      : -THE FRAME SIZE?
          02C8 589      BRB    42$ ; YES THEN DON'T BOTHER IT
          00 00F8 C5 02 E5 02CA 590 36$: BBCC  #UCBSV_TT_USERFRAME,UCBSB_TT_PARITY(R5),38$
          009C C3 FFFFFFF0 8F CB 02D0 591 38$: BICL3  #^C<^XOF>,IRPSL_VAL5(R3),R0; GET THE NEW FRAME SIZE
          00F8 C5 02 03 50 F0 02DA 592      BEQL   37$ ; 0 SPECIFIED THEN CLEAR USER FRAME
          1C 00F8 C5 02 E3 02DC 593      INSV   R0,#UCBSV_TT_LEN,#2,UCBSB_TT_PARITY(R5); SET THE
          02E3 594      : -PARITY CORRECTLY
          02E9 595      BBBS   #UCBSV_TT_USERFRAME,UCBSB_TT_PARITY(R5),42$; AND SET USER FRAME
          02E9 596      : -SPECIFIED THEN CONTINUE ON
          11 00F8 C5 06 E1 02E9 597 37$: BBC    #TT$V_PARITY,-
          0F E0 02EB 598      UCBSB_TT_PARITY(R5),40$ ; IF NO PARITY, USE 8 BIT
          0C 44 A5 02F1 600      BBS    #TT$V_EIGHTBIT,-
          00F8 C5 18 8A 02F4 601      UCBSL_DEVDEPND(R5),40$ ; USE 8 BIT SIZE
          00F8 C5 10 88 02F9 602      BICB   #UCBSM_TT_LEN,UCBSB_TT_PARITY(R5) ; RESET CHARACTER FRAME
          05 11 02FE 603      BISB   #^X10,UCBSB_TT_PARITY(R5) ; SET 7 BIT CHARACTER FRAME
          0300 604 40$: BRB    42$
          00F8 C5 18 88 0300 605      BISB   #UCBSM_TT_LEN,UCBSB_TT_PARITY(R5) ; SET 8 BIT CHARACTER FRAME
          0305 606 42$:

```

```

11 009C C3 0A E1 0305 607 BBC #TTSV_ALTDISPAR,IRPSL_VAL5(R3),41$ ; CHECK FOR DISABLE PARITY E
   00F8 C5 02 CA 030B 608 BICL #UCBSM TT DISPARERR,UCBSB TT PARITY(R5) ; CLEAR DISMISS
06 009C C3 09 E1 0310 609 BBC #TTSV_DISPARERR,IRPSL_VAL5(R3),41$ ; DOES HE WANT IT SET
00 00F8 C5 01 E2 0316 610 BBSS #UCBSV TT DISPARERR,UCBSB TT PARITY(R5),41$ ; YES THEN SET IT
07 009C C3 08 E0 031C 611 41$: BBS #TTSV_TWOSTOP,IRPSL_VAL5(R3),43$ ; DOES HE WANT TWO STOP BITS
   04 00F4 C5 91 0322 612 CMPB UCBSW_TT_SPEED(R5),#4 ; SPEED <= 150 BAUD?
   07 14 0327 613 BGTR 44$ ; NO
   00F8 C5 20 88 0329 614 43$: BISB #UCBSM_TT_STOP,UCBSB_TT_PARITY(R5) ; FLAG 2 STOP BITS
   05 11 032E 615 BRB 46$
   00F8 C5 20 8A 0330 616 44$: BICB #UCBSM_TT_STOP,UCBSB_TT_PARITY(R5) ; FLAG 1 STOP BIT
   0335 618
   0335 619 : PROCESS FILL DATA
   0335 620
   0335 621 46$:
   0335 622 CLRL R0 ; ASSUME NEW VALUE IS 0
CA 44 A5 0A E1 0337 623 BBC #TTSV_CRFILL,UCBSL_DEVDEPEND(R5),50$; CR FILL ON?
   50 4E A3 9A 033C 624 MOVZBL IRPSW_TT_PRMP+2(R3),R0 ; GET NEW VALUE
   02 54 0A E0 0340 625 BBS #TTSV_CRFILL,R4,50$ ; CHANGE?
   0C 13 0344 626 BEQL 60$ ; IF NEQ EQL 0 THEN NO CHANGE
   00F6 C5 50 90 0346 627 50$: MOVB R0,UCBSB_TT_CRFILL(R5) ; RESET VALUE
   05 12 034B 628 BNEQ 60$ ; IF NEQ THEN OK
00 44 A5 0A E5 034D 629 BBCC #TTSV_CRFILL,UCBSL_DEVDEPEND(R5),60$; SET OFF
   50 D4 0352 630 60$: CLRL R0 ; ASSUME NEW VALUE IS 0
0A 44 A5 0B E1 0354 631 BBC #TTSV_LFFILL,UCBSL_DEVDEPEND(R5),65$; LF FILL ON?
   50 4F A3 9A 0359 632 MOVZBL IRPSW_TT_PRMP+3(R3),R0 ; GET NEW VALUE
   02 54 0B E0 035D 633 BBS #TTSV_LFFILL,R4,65$ ; CHANGE?
   0C 13 0361 634 BEQL 75$ ; IF NEQ EQL 0 THEN NO CHANGE
   00F7 C5 50 90 0363 635 65$: MOVB R0,UCBSB_TT_LFFILL(R5) ; RESET VALUE
   05 12 0368 636 BNEQ 75$ ; IF NEQ THEN OK
00 44 A5 0B E5 036A 637 BBCC #TTSV_LFFILL,UCBSL_DEVDEPEND(R5),75$; SET OFF
   036F 638 :
   036F 639 : CHECK FOR CHANGE IN STATUS OF MODEM
   036F 640 :
   036F 641 75$:
   12 54 15 E1 036F 642 BBC #TTSV_MODEM,R4,80$ ; NO CHANGE IN MODEM STATUS
   1F BB 0373 643 PUSHR #M<R0,R1,R2,R3,R4> ; SAVE VOLITAL REGISTERS
   51 00 9A 0375 644 MOVZBL #MODEM$C_INIT,R1 ; ASSUME MODEM INIT
44 A5 15 E0 0378 645 BBS #TTSV_MODEM,UCBSL_DEVDEPEND(R5),- ; IT IS INIT
   03 037C 646 MOVZBL #MODEM$C_SHUTDWN,R1 ; NO, SHUT DOWN
   51 01 9A 037D 647 77$:
   0380 648 77$:
   FC7D' 30 0380 649 BSBW TRANSITION_NOCHECK ; DECLARE MODEM TRANSITION
   1F BA 0383 650 POPR #M<R0,R1,R2,R3,R4>
   0385 651
   0385 652 80$:
   0385 653 :
   0385 654 : enable or disable AUTO XON AND XOFF
   0385 655 :
0122 C5 0040 8F A8 0385 656 BISW #TTY$M_PC_XOFENA,UCBSW_TT_PRTCTL(R5); TURN ON AUTO XOFF
   05 44 A5 00 E0 038C 657 BBS #TTSV_PASSALL,UCBSL_DEVDEPEND(R5),84$; IS THIS PASSALL? YES THEN
   07 44 A5 05 E0 0391 658 BBS #TTSV_TTSYNC,UCBSL_DEVDEPEND(R5),85$; IF TT SYNC IS SET THEN LEAVE I
0122 C5 0040 8F AA 0396 659 84$: BICW #TTY$M_PC_XOFENA,UCBSW_TT_PRTCTL(R5); TURN OFF AUTO XOFF
   039D 660 85$:
   039D 661 :
   039D 662 : COPY OVER PASSALL AND NOECHO TO CURRENT STATE
   039D 663 :

```

04	54	A2	02	02	54	FO	03A3	665	INSV	R4,#TTYSV ST_PASALL,#2,4(R2);	INSERT IN STATE VECTOR							
54	44	A5	01	03	EE	03A9	666	EXTV	#TT2SV ESCAPE,#1,UCBSL_DEVDEPND(R5),R4	; GET CURRENT SETTING								
04	A2	01	0B	54	FO	03AF	667	INSV	R4,#TTYSV ST_ESCAPE,#T,4(R2)	; UPDATE IN STATE								
		04	48	A5	12	E1	03B5	668	BBC	#TT2SV PASTHRU,UCBSL_DEVDEPND2(R5),98\$; IN PASS THRU MODE							
							03BA	669	SET_STATE	PASACL								
							03BE	670	98\$:									
							03BE	671	:									
							03BE	672	:	INIT THE UNIT TO CHANGE THE SPEED AND PARITY								
							03BE	673	:									
						FC3F'	30	03BE	674	BSBW TTY\$SET_LINE	; INIT LINE SPEED AND PARITY							
							03C1	675	:									
							03C1	676	:	IF MULTI IS SET THEN THE I/O MUST BE STARTED								
							03C1	677	:									
							03C1	678	100\$:	IF NOT_STATE MULTI,110\$; BR IF MULTI NO SET							
						016A	30	03C5	679	BSBW TTY\$STARTOUTPUT	; START THE MULTIPLE OUTPUT							
							03C8	680	:									
							03C8	681	:	CHECK FOR SET CHARACTERISTICS AND RETURN IOSB DATA								
							03C8	682	:									
							03C8	683	110\$:	MOVL UCBSL_IRP(R5),R3	; GET CURRENT PACKET ADDRESS							
						03	20	A3	06	00	ED	03CC	684	CMPZV	#IRPSV_FCODE,#IRPSS_FCODE,IRPSW_FUNC(R3),#TTYSC_FC_SETC;	SET CHAR?		
												12	03D2	685	BNEQ	120\$; IF NEQ THEN NO	
													03D4	686	MOVL	UCBSW_TT_SPEED(R5),UCBSW_TT_DESPEE(R5);	RESET PERM SPEED	
													03DB	687	BBC	#TT2SV AUTOBAUD,-		
													03DD	688		UCBSL_DEVDEPND2(R5),115\$; BRANCH IF NOAUTOBAUD	
													03E0	689	MOVZBW	#TTSC_BAUD 9600,-		
													03E2	690		UCBSW_TT_DESPEE(R5)	; SET PERMANENT 9600 BAUD FOR AUTOBAUD	
													03E5	691	115\$:	MOVB	UCBSB_TT_PARITY(R5),UCBSB_TT_DEPARI(R5);	RESET PERM PARITY
													03EC	692	MOVL	UCBSB_DEVTYPE(R5),UCBSB_TT_DETYPE(R5);	RESET TYPE AND WIDTH	
													03F2	693	BICL3	#TTSM_REMOTE,UCBSL_DEVDEPND(R5),UCBSL_TT_DECHAR(R5);	RESET PERM CHA	
													03FD	694	MOVL	UCBSL_DEVDEPND2(R5),UCBSL_TT_DECHA1(R5)	; UPDATE SECOND CHAR WORD	
													0403	695				
													0403	696	BBC	#TT2SV_ALTYPEAHD,IRPSQ_TT_STATE+4(R3),120\$;	SKIP IF ALTERNATE	
													0408	697			; TYPEAHEAD NOT SPECIFIED	
													04C8	698	BBSS	#TT2SV_ALTYPEAHD,UCBSL_TT_DECHA1(R5),120\$;	ONLY ALLOW SETTING	
													040E	699			; AS PERM CHARACTERISTIC	
													040E	700	120\$:			
													040E	701				
													040E	702	:		UPDATE NEWLY WRITTEN FIELDS WHICH ARE MAINTAINED	
													040E	703	:		IN BOTH THE LOGICAL ANY PHYSICAL UCB	
													040E	704	:			
													040E	705	MOVL	UCBSL_TT_LOGUCB(R5),R0	; GET LOGICAL UCB ADDRESS	
													0413	706	MOVL	UCBSL_DEVDEPND(R5),UCBSL_DEVDEPND(R0)	; UPDATE CHARACTERISTICS	
													0418	707	MOVL	UCBSL_DEVDEPND2(R5),UCBSL_DEVDEPND2(R0)	; 'UCBSL_TT_DEVDP1'	
													041D	708	MOVL	UCBSB_DEVCLASS(R5),UCBSB_DEVCLASS(R0)	; CLASS,TYPE,BUFSIZE	
													0422	709				
													0422	710	:		THIS ROUTINE COMPLETES SET AND SENSE CHARACTERISTICS OPERATIONS	
													0422	711	:		AND RETURNS STATUS VALUES IN THE IOSB	
													0422	712	:			
													0422	713	DO_EXIT:			
													0422	714	MOVL	UCBSW_TT_SPEED-2(R5),R0;	RETURN SPEED	
													0427	715	BICL3	#C<<UCBSM_TT_PARTY!UCBSM_TT_ODD>>@16>,UCBSB_TT_PARITY-2(R5),R1;		
													0431	716	MOVW	UCBSB_TT_CRFIL(R5),R1	; INSERT FILL DATA	
													0436	717	MOVW	#SS\$ NORMAL,R0	; SET STATUS	
													0439	718	BRW	TTYSDONE		
													043C	719				
													043C	720	NOPRIV_EXIT:			

TTYSTRSTP
V04-000

D 7
- Terminal driver start/stop I/O routine 16-SEP-1984 02:18:30 VAX/VMS Macro V04-00
START_IO ACTION ROUTINES 5-SEP-1984 04:17:09 [TTDRVR.SRC]TTYSTRSTP.MAR;1 Page 18
(13)

50	24	D0	043C	721	MOVL	#SS\$_NOPRIV,R0	:	SET NO PRIV
	51	D4	043F	722	CLRL	R1	:	CLEAR R1
02F1	31	0441	723	BRW	TTYSDONE			
		0444	724					

TT
VO

```

0444 726 :
0444 727 : WRITE OPERATION
0444 728 :
0444 729 DO_WRITE: ;
0444 730 :
0444 731 : Control only comes here in the case of half duplex writes. Full
0444 732 : duplex writes use the TTY$WRSTARTIO entry point.
0444 733 :
53 2C A3 DO 0444 734 MOVL IRP$L_SVAPE(R3), R3 ; Get addr of write block
00000532'EF 9F 0448 735 PUSHAB TTY$STARTOUTPUT ; if write is started, control
; will return to STARTOUTPUT
0040 30 044E 736 BSBW WRTSTARTIO ; if queued, control will return
05 0451 738 RSB ; here. Return to caller.

```

```

0452 740      .SBTTL TTY$WRTSTARTIO - Starts or queues a write operation
0452 741
0452 742      :++
0452 743      :
0452 744      : Functional description:
0452 745      :
0452 746      : If called from an FDI routine (or from EXE$BRDCST),
0452 747      : TTY$WRTSTARTIO first raises to device IPL, and then calls the
0452 748      : internal routine. All other code enters through the WRTSTARTIO
0452 749      : entry point.
0452 750      :
0452 751      : If a write is occurring, the routine queues the write buffer.
0452 752      : If a read is occurring, but the buffer header specifies
0452 753      : write-breakthrough, the routine starts the write.
0452 754      : If a read is occurring, but no characters have been received
0452 755      : yet, the routine starts the write.
0452 756      : Otherwise, the routine queues the write buffer.
0452 757      :
0452 758      : To start the write operation, the routine writes the address
0452 759      : of the buffer in UCBSL_IT_WRTBUF, sets and clears various
0452 760      : state bits, and returns.
0452 761      :
0452 762      : To queue the buffer, the routine inserts the buffer address at
0452 763      : the end of the queue unless the header specifies write-
0452 764      : breakthrough. In the latter case, the buffer address is inserted
0452 765      : after the last write-breakthrough request in the queue.
0452 766      :
0452 767      : Returning from WRTSTARTIO is odd. The routine assumes that 0(SP)
0452 768      : is the address to return to if the write is to start now.
0452 769      : If the routine instead queues the write, the routine pops this
0452 770      : start-write address of the stack, and returns to the real
0452 771      : caller.
0452 772      :
0452 773      : Inputs:
0452 774      :
0452 775      : R3      - address of the write buffer
0452 776      : R5      - address of the UCB
0452 777      :
0452 778      : Implicit inputs:
0452 779      :
0452 780      : The write buffer consists of a header, and an optional message
0452 781      : buffer. For broadcast messages, the message buffer is absent.
0452 782      : TTY$WB_FR3 is 0 for a normal broadcast and 1 for and ANSI
0452 783      : broadcast or one that specified norefresh.
0452 784      :
0452 785      : Outputs:
0452 786      :
0452 787      : If the broadcast message is rejected, the TTY$WB_END field
0452 788      : of the write packet is zeroed.
0452 789      :
0452 790      : R0      - preserved
0452 791      : R1      - scratch
0452 792      : R2      - address of UCB state bits
0452 793      : R3      - address of buffer
0452 794      : R4      - if packet is started, address of IRP or 0
0452 795      :           - if packet is queued, scratch
0452 796      : R5      - address of UCB
  
```

```

0452 797 :
0452 798 : Implicit outputs:
0452 799 :
0452 800 : Buffer may be entered in queue.
0452 801 :
0452 802 : If write operation is started,
0452 803 : UCBSL_TT_WRTBUF - address of buffer
0452 804 : UCBSQ_TT_STATE - write bit, and other bits from IRP are set
0452 805 : control-0 may be canceled
0452 806 : UCBSL_DEVDEPEND - mailbox may be enabled
0452 807 :
0452 808 :--
0452 809 :
0452 810 TTY$WRTSTARTIO: : Start or queue write.
51 00A0 C5 D0 0452 811 MOVL UCBSL_TL_PHYUCB(R5),R1 : GET PUCB ADDRESS
: 22 13 0457 812 BEQL 30$ : NONE CURRENTLY EXISTS
: 55 51 D0 0459 813 MOVL R1,R5 : SWITCH TO PUCB CONTEXT
: 24 A3 D5 045C 814 TSTL TTY$L_WB_IRP(R3) : Is this a broadcast?
: OC 13 045F 815 BEQL 20$ : YES, SPECIAL CASE
: FB9C' 30 0461 816 10$: : Acceptable packet.
00000532'EF 9F 0461 817 BSBW TTY$LOCK : Raise to DIPL, get states.
: 25 10 046A 818 PUSHAB TTY$STARTOUTPUT : Set up return address to
: 05 046A 819 : start the output.
: 046A 820 BSBW WRTSTARTIO : Start or queue the packet.
: 046C 821 RSB : If packet queued, control
: 046D 822 : returns here, so return to
: 046D 823 : caller.
: 046D 824 : INTERNAL BROADCAST PACKET.
: 046D 825 : CHECK FOR DISABLE
: 046D 826 20$:
44 A5 00020001 8F D3 046D 827 BITL #TTSM_PASSALL!- : Test for passall and/or
: EA 13 046E 828 TTSM_NOBRDCST,- : nobroadcast modes set in the
: 20 A3 D4 046E 829 UCBSL_DEVDEPEND(R5) : term's UCB (ignore NOECHO).
: 047A 830 BEQL 10$ : Continue if not set.
: 047A 831 25$: CLRL TTY$L_WB_END(R3) : Zero end address to indicate
: 047A 832 : failure to EXE$BRDCST.
: 047B 833 RSB : And return to EXE$BRDCST.
: 047B 834 :
: 047B 835 : LUCB CURRENTLY DETACHED
: 047B 836 : COMPLETE THE WRITE
: 047B 837 : ASSUME IRP$L_IOST1+4 EQ IRP$L_IOST2
: 047B 838 30$:
: 51 24 A3 D0 047B 839 MOVL TTY$L_WB_IRP(R3),R1 : GET IRP ADDRESS
: F6 13 047F 840 BEQL 25$ : INTERNAL BROADCAST, REJECT IT.
: 53 51 D0 0481 841 MOVL R1,R3 : GET IRP ADDRESS
: 38 A3 7C 0484 842 CLRQ IRP$L_IOST1(R3) : INIT IOSB RETURN
: 00000000'GF 80 0487 843 MOVW #SS$NORMAL.IRP$L_IOST1(R3)
: 17 048B 844 JMP G^COM$POST
: 0491 845 :
: 0491 846 :
: 0491 847 :
: 0491 848 WRTSTARTIO: : Checks for start or queue.
: 0491 849 :
: 0491 850 :
: 0491 851 : Inputs:
: 0491 852 :
: 0491 853 : R2 - address of state bits longword

```


				0491	854	:	R3	- address of write packet	
				0491	855	:	R5	- address of the device's UCB	
				0491	856	:			
				0491	857	:	0(SP)	- address to return if write is queued	
				0491	858	:	4(SP)	- address to return to if write is started	
				0491	859	:			
				0491	860	:			
		01	BB	0491	861		PUSHR	#*M<R0>	: Save a register.
50	00CC	C5	9E	0493	862		MOVAB	UCBSL_TT_WFLINK(R5),R0	: Get address of write queue.
54	24	A3	D0	0498	863		MOVL	TTY\$LB_WB_IRP(R3),R4	: Get address of IRP.
				049C	864		IF_STATE	-	: If writing is in progress,
				049C	865			WRITE,QUEUE_PKT	: just queue the packet.
		54	D5	04A0	866		TSTL	R4	: See if the packet has an IRP.
		42	13	04A2	867		BEQL	START_PKT	: If not, just start the packet.
				04A4	868		IF_NOT_STATE	=	: If not in a read state,
				04A4	869			READ,START_PKT	: go ahead and start the packet.
				04AB	870		IF_STATE	-	: If noecho read, no blocking
				04AB	871			NOECHO,START_PKT	: so start the write
		09	E0	04AC	872		BBS	#IOSV_BREAKTHRU,-	
35	20	A4		04AE	873			IRPSW_FUNC(R4),START_PKT	: Start if break thru write.
51	78	A5	D0	04B1	874		MOVL	UCBSL_SVAPTE(R5),R1	: get the read packet address
	3C	A1	B5	04B5	875		TSTW	TTY\$WB_TXTOFF(R1)	: and check if we have started typing
		2C	13	04B8	876		BEQL	START_PKT	: go start the packet.
		19	11	04BA	877		BRB	QUEUE_LAST	: Otherwise, queue packet at end
				04BC	878				: of queue.
				04BC	879				
				04BC	880		QUEUE_PKT:		
		54	D5	04BC	881		TSTL	R4	: If an IRP is associated,
		15	12	04BE	882		BNEQ	QUEUE_LAST	: queue packet at end of queue.
54	50	D0		04C0	883		MOVL	R0,R4	: Make a copy of queue head.
				04C3	884				
				04C3	885		10\$:		
51	64	D0		04C3	886		MOVL	TTY\$LB_WB_FLINK(R4),R1	: Get first queue entry.
50	51	D1		04C6	887		CMPL	R1,R0	: See if at end of queue.
	0A	13		04C9	888		BEQL	QUEUE_LAST	: If yes, put at end of queue.
	24	A1	D5	04CB	889		TSTL	TTY\$LB_WB_IRP(R1)	: Else, see if this entry has an
				04CE	890				: associated IRP.
		09	12	04CE	891		BNEQ	INSERT_PKT	: If yes, branch to insert
				04D0	892				: packet before it.
54	51	D0		04D0	893		MOVL	R1,R4	: Otherwise, go on to next
	EE	11		04D3	894		BRB	10\$: entry in queue.
				04D5	895				
				04D5	896		QUEUE_LAST:		: Queue at end of queue.
54	04	A0	D0	04D5	897		MOVL	TTY\$LB_WB_BLINK(R0),R4	: Get back pointer.
				04D9	898				
				04D9	899		INSERT_PKT:		
		63	0E	04D9	900		INSQUE	TTY\$LB_WB_FLINK(R3),-	: Insert new packet in the
		64		04DB	901			TTY\$LB_WB_FLINK(R4)	: queue.
		01	BA	04DC	902		POPR	#*M<R0>	: Remove saved register and
51	6E	D0		04DE	903		MOVL	(SP),R1	: get queued address
5E	08	C0		04E1	904		ADDL	#8,SP	: clean stack
	61	17		04E4	905		JMP	(R1)	: return to queued address
				04E6	906				
				04E6	907		START_PKT:		: Start the packet.
00D4	C5	53	D0	04E6	908		MOVL	R3,UCBSL_TT_WRTBUF(R5)	: Point to packet from UCB.
				04EB	909		SET_STATE	-	: Set the write state.
				04EB	910			WRITE	

TTYSTRSTI
V04-000

L 7

- Terminal driver start/stop I/O routine 16-SEP-1984 02:18:30 VAX/VMS Macro V04-00 Page 26
TTY\$GETNXTWRITE - Start next write or re 5-SEP-1984 04:17:09 [TTDRVR.SRC]TTYSTRSTP.MAR;1 (17)

	05	0582	1034		RSB		; Return to GETNEXTCHAR.
		0583	1035				
		0583	1036	20\$:			
FA7A'	30	0583	1037		BSBW	TTY\$RESTARTIO	; Restart the read, if any.
		0586	1038				
	05	0586	1039	30\$:			; Joint read/write return.
		0586	1040		RSB		; Return.

TT
Ps

PS
--
\$A
\$\$

Ph
--
In
Co
Pa
Sy
Pa
Sy
Ps
Cr
As

Th
15
Th
14
56

Ma
--
-s
-s
TO
27
Th
MA

```

0587 1042      .SBTTL  TTY$WRITEDONE - Complete a write operation
0587 1043
0587 1044      :++
0587 1045      : TTY$WRITEDONE - WRITE OPERATION DONE
0587 1046      :
0587 1047      : FUNCTIONAL DESCRIPTION:
0587 1048      :
0587 1049      :     This routine creates a fork process to complete the write, and
0587 1050      :     checks for another write packet to start up.
0587 1051      :
0587 1052      : INPUTS:
0587 1053      :
0587 1054      :     R2 = ADDRESS OF THE UNIT STATE VECTOR
0587 1055      :     R5 = UCB ADDRESS
0587 1056      :
0587 1057      :     TTY$W_WB_STATUS - status of operation
0587 1058      :     TTY$W_WB_BCNT  - number of bytes transferred
0587 1059      :
0587 1060      : OUTPUTS:
0587 1061      :
0587 1062      :     R2,R5 ARE PRESERVED.
0587 1063      :
0587 1064      :--
0587 1065
0587 1066      TTY$WRITEDONE::          ; Complete write operation.
0587 1067
0587 1068      :
0587 1069      : This routine used to start by clearing a whole raft of state bits.
0587 1070      : I only turn off write-related bits, and I do that in TTY$GETNXTWRITE.
0587 1071      : The bits I no longer modify are:
0587 1072      :
0587 1073      :     READ, DEL, XON, EOL, PROMPT, CTRLR, NOFLTR, ESC, ESC_0, and
0587 1074      :     BADESC
0587 1075      :
0587 1076
0587 1077
0587 1078      24  BB  0587 1078      PUSHR  #^M<R2,R5>          ; Save state and UCB address.
0589 1079      14$:
0589 1080
0589 1081      53  00D4 C5  D0  0589 1081      MOVL   UCBSL_TT_WRTBUF(R5),R3 ; Get address of write buffer.
058E 1082
058E 1083      :
058E 1084      : NEW LINE MODIFIER
058E 1085      :
058E 1086      54  24 A3  D0  058E 1086      MOVL   TTY$W_WB_IRP(R3),R4    ; Get address of associated IRP.
0592 1087      0C  13      BEQL   10$
0594 1088      07  20 A4  0A  E1  0594 1088      BBC    #IOSV_NEWLINE,IRP$W_FUNC(R4),10$; NO NEWLINE THEN DON'T ADD A THING
0599 1089
05A0 1090      10$:
05A0 1091      CLR_STATE -          ; Clear the write bits.
05A0 1092      <WRITE,WRTALL>
05A8 1093      9D  AF  9F  05A8 1093      PUSHAB TTY$GETNXTWRITE      ; Return address after queuing fork
05AB 1094
05AB 1095      WRITEPOST:
05AB 1096
05AB 1097      0B  A5  90  05AB 1097      MOVB   UCBSB_FIPL(R5),-      ; Set up fork IPL in the buffer
05AE 1098      0B  A3      05AE 1098      TTY$B_WB_FIPL(R3)          ; block.

```

```

54 24 A3 D0 05B0 1099      MOVL  TTY$WB_IRP(R3),R4      ; Get address of associated IRP.
      08 12 05B4 1100      BNEQ  5$                    ; one there then continue
      06 90 05B6 1101      MOVB  #IPL$_QUEUEAST,-      ; for broadcast fork to ipl6
      0B A3 05B8 1102      TTY$WB_FIPL(R3)
      7E D4 05BA 1103      cLrL  -(sp)-                ; no irp then no process to fork to
      03 11 05BC 1104      brb   7$
      05BE 1105 5$:
      0C A4 DD 05BE 1106      PUSHL IRP$PID(R4)           ; FORK ON THE PID OF THE IO OWNER
      55 53 DO 05C1 1107 7$:  MOVL  R3,R5                    ; Setup fork block address.
00000000'EF 16 05C4 1108      JSB   TTY$SYNCH            ; Create a fork process.
      05CA 1109
      05CA 1110
      05CA 1111 : This is the write completion fork process. Registers are as follows:
      05CA 1112 :
      05CA 1113 : R4 - address of IRP
      05CA 1114 : R5 - address of write buffer (TWP)
      05CA 1115 :
      05CA 1116 :
      53 54 DO 05CA 1117      MOVL  R4,R3                    ; Need IRP in R3 for I/O post.
      54 13 05CD 1118      BEQL  100$
      05CF 1119
      05CF 1120      .IF DF CAS_MEASURE_IOT
      05CF 1121 :
      05CF 1122 : ACCUMULATE STATISTICS ON NUMBER OF CHARACTERS AND I/OS TO TERMINALS.
      05CF 1123 :
      05CF 1124      BSBB  TTSTATS                    ; CALL STATISTICS ROUTINE.
      05CF 1125
      05CF 1126      .ENDC
      05CF 1127
      05CF 1128 :
      05CF 1129 : NOTE: IRP$MEDIA = IRP$IOST1
      05CF 1130 : NOTE: terminal position is 0-based; interface position is 1-based
      05CF 1131 :
      05CF 1132 :
      54 55 DO 05CF 1133      MOVL  R5,R4                    ; Put buffer address in R4.
      38 A3 3C 05D2 1134      MOVZWL IRP$MEDIA(R3),-      ; number of lines output for the
      3C A3 05D5 1135      IRP$IOST2(R3)              ; write QIO, and zero other values
      28 A4 DO 05D7 1136      MOVL  TTY$WB_STATUS(R4),-  ; move status and count of bytes
      38 A3 05DA 1137      IRP$IOST1(R3)              ; transferred into IOSB
      05DC 1138
      55 1C A3 DO 05DC 1139      MOVL  IRP$UCB(R3),R5        ; Reg. in LUCB address.
      55 00A0 C5 DO 05E0 1140      MOVL  UCBSL_TL_PHYUCB(R5),R5 ; Switch to Physical context
      1D 13 05E5 1141      BEQL  15$                    ; Disconnect has occurred!
      00FC C5 B1 05E7 1142      CMPW  UCBSW_TT_CURSOR(R5),-  ; Is cursor marker beyond the right-
      42 A5 05EB 1143      UCBSW_DEVBUFSIZ(R5)      ; hand edge of screen?
      09 1E 05ED 1144      BGEQU 12$                    ; Branch if cursor has gone too far.
      00FC C5 A1 05EF 1145      ADDW3 UCBSW_TT_CURSOR(R5),-  ; Else return cursor column position,
      3E A3 01 05F3 1146      #1,IRP$IOST2+2(R3)      ; adjusted for zero offset, in IOSB.
      05 11 05F6 1147      BRB   14$                    ; Continue building IOSB.
      42 A5 B0 05F8 1148 12$:  MOVW  UCBSW_DEVBUFSIZ(R5),-  ; If necessary, return cursor column
      3E A3 05FB 1149      IRP$IOST2+2(R3)      ; position in IOSB as right-hand edge.
      00FE C5 81 05FD 1150 14$:  ADDB3 UCBSW_TT_LINE(R5),-  ; move line position into IOSB
      3F A3 01 0601 1151      #1,IRP$IOST2+3(R3)      ;
      0604 1152
      08 A4 B0 0604 1153 15$:  MOVW  TTY$WB_SIZE(R4),-  ; Move size of buffer into IRP
      30 A3 0607 1154      IRP$BOFF(R3)          ; to record quota used.
      55 1C A3 DO 0609 1155      MOVL  IRP$UCB(R3),R5        ; Restore logical UCB address
  
```

58 A5	53	L1	060D	1156		CMP	R3,UCB\$\$_IRP(R5)	:	Is this the current write
	06	13	0611	1157		BEQ	30\$:	blocking the i/o queue?
			0613	1158					
C0000000	'GF	17	0613	1159	20\$:	JMP	G^COM\$POST	:	Full duplex: complete write
			0619	1160					
			0619	1161	30\$:			:	Half duplex:
50	38 A3	7D	0619	1162		MOVQ	IRP\$\$_MEDIA(R3),R0	:	Load IOST1 and IOST2 in R0,R1
			061D	1163		REQCOM		:	Complete request and get next
			0623	1164				:	entry in system queue.
			0623	1165	100\$:				
53	20 A5	D0	0623	1166		MOVL	TTY\$\$_WB_END(R5),R3	:	GET THE ADDRESS OF THE LAST CHARACTER
	2C B5	17	0627	1167		JMP	@TTY\$\$_WB_RETADDR(R5)	:	Want fork process to gain control
			062A	1168				:	string for fork process use.
			062A	1169					
			062A	1170					


```

062A 1172      .SBTTL  TTY$WRITEPOST - QUEUE A WRITE COMPLETION
062A 1173
062A 1174      :++
062A 1175      : TTY$WRITEPOST - QUEUE A WRITE COMPLETION
062A 1176      :
062A 1177      : FUNCTIONAL DESCRIPTION:
062A 1178      :
062A 1179      :     THIS ROUTINE FORKS ON A TWP TO COMPLETE A QUEUED WRITE OPERATION
062A 1180      :     BOTH HALF AND FULL DUPLEX.
062A 1181      :
062A 1182      : INPUTS:
062A 1183      :
062A 1184      :     R4 = TWP ADDRESS
062A 1185      :     R5 = UCB ADDRESS
062A 1186      :
062A 1187      :     UCB$W_BOFF - STATUS OF OPERATION
062A 1188      :
062A 1189      : OUTPUTS:
062A 1190      :
062A 1191      :     R0,R1,R2,R3,R5 ARE PRESERVED
062A 1192      :--
062A 1193
062A 1194      TTY$WRITEPOST::
062A 1195
53   28   BB 062A 1196      PUSHR  #^M<R3,R5>      : SAVE REGISTERS
      54   DO 062C 1197      MOVL   R4,R3        : TWP ADDRESS
2A   A3   B4 062F 1198      CLRW  TTY$WB BCNT(R3) : NONE TRANSFERED
7C   A5   B0 0632 1199      MOVW  UCB$W_BOFF(R5),- : SAVE COMPLETION STATUS
28   A3   0635 1200      TTY$WB STATUS(R3)
      FF71 30 0637 1201      BSBW  WRITEPOST      : QUE THE FORK
      28   BA 063A 1202      POPR  #^M<R3,R5>
      05   063C 1203      RSB
      063D 1204
      063D 1205
  
```

```

063D 1207
063D 1208 .IF DF CAS_MEASURE_IOT
063D 1209 :
063D 1210 : Subroutine to accumulate statistics on the number of
063D 1211 : the number of characters read and written to terminals
063L 1212 :
063D 1213 TSTATS:BLBC G^PMS$GL_DOSTATS,40$ : IF FLAG SET, BYPASS STATISTICS CODE
063D 1214 MOVZWL IRP$W_BCNT(R3),R1 : GET # CHARACTERS TRANSFERRED.
063D 1215 DIVL3 #5,R1,RO : STATISTICS ARE KEPT IN INCREMENTS
063D 1216 : OF 5 CHARACTERS.
063D 1217 CMPL #9,RO : LAST ENTRY IN TABLE IS FOR I/OS
063D 1218 : OF >= 45 CHARACTERS.
063D 1219 BGEQ 10$
063D 1220 MOVL #9,RO
063D 1221 10$: CMPZV #IRP$V_FCODE,#IRP$S_FCODE,IRP$W_FUNC(R3),#TTY$C_FC_READ
063D 1222 : CHECK IF JUST FINISHED A READ OR WRITE.
063D 1223 BNEQ 20$ : BRANCH FOR WRITE
063D 1224 :
063D 1225 : COMPILE STATISTICS FOR READ
063D 1226 :
063D 1227 INCL G^PMS$AL_READTBL[RO] : INCREMENT APPROPRIATE RANGE.
063D 1228 INCL G^PMS$GL_TREADS : INCREMENT READ COUNT
063D 1229 ADDL2 R1,G^PMS$GL_READCNT : INCREMENT TOTAL COUNT FOR CHARACTERS
063D 1230 BRB 40$
063D 1231 :
063D 1232 : COMPILE STATISTICS FOR WRITE
063D 1233 :
063D 1234 20$: INCL G^PMS$AL_WRITETBL[RO] : INCREMENT APPROPRIATE RANGE.
063D 1235 INCL G^PMS$GL_TWRTES : INCREMENT WRITE COUNT
063D 1236 ADDL2 R1,G^PMS$GL_WRTCNT : INCREMENT TOTAL COUNT FOR CHARACTERS
063D 1237 : WRITTEN.
063D 1238 40$: RSB : RETURN TO CALLER.
063D 1239
063D 1240 .ENDC
  
```

```

063D 1242 .SBTTL TTY$READONE - READ OPERATION DONE
063D 1243 :++
063D 1244 : TTY$READONE - READ I/O OPERATION DONE
063D 1245 :
063D 1246 : FUNCTIONAL DESCRIPTION:
063D 1247 :
063D 1248 : THIS ROUTINE IS ENTERED TO COMPLETE THE CURRENT READ OPERATION.
063D 1249 : THE ACTION IS TO RESET THE STATE OF THE UNIT TO REFLECT THE CHANGE AND TO
063D 1250 : FORK ON THE IRP TO COMPLETE THE PROCESSING.
063D 1251 :
063D 1252 : INPUTS:
063D 1253 :
063D 1254 : R2 = ADDRESS OF THE UNIT STATE VECTOR
063D 1255 : R5 = UCB ADDRESS
063D 1256 :
063D 1257 : UCBSW_BOFF = STATUS WORD
063D 1258 : UCBSW_BCNT = COUNT OF TRANSFER
063D 1259 :
063D 1260 : IRP$L_MEDIA(CURRENT PACKET) = TERMINATOR AND TERMINATOR SIZE
063D 1261 :
063D 1262 : OUTPUTS:
063D 1263 :
063D 1264 :
063D 1265 : NONE
063D 1266 :--
063D 1267 .enable lsb
063D 1268 TTY$READONE:: : READ I/O DONE
03 44 A5 12 E1 063D 1269 BBC #TT$V_READSYNC,UCBSL_DEVDEPEND(R5),10$; BR IF NOT READSYNC
68 A5 02 AA 0642 1270 BSBW TTY$XOFF : SEND XOFF
0645 1271 10$: BICW #UCBSM_TT_TIMO,UCBSW_DEVSTS(R5); CLEAR TIMEOUT ENABLED
0649 1272 :
0649 1273 : SET UP ERRORS ON ESCAPE SEQUENCES
0649 1274 :
0649 1275 : IF NOT_STATE ESC,15$ : IF NOT ESCAPE THEN BR
7C A5 01FC 8F B0 064D 1276 MOVW #SS$_PARTESCAPE,UCBSW_BOFF(R5); ASSUME PARTIAL ESCAPE SEQUENCE
54 78 A5 D0 0653 1277 MOVL UCBSL_SVAPTE(R5),R4 : GET THE ADDRESS OF THE READ PACKET
53 58 A5 D0 0657 1278 MOVL UCBSL_IRP(R5),R3 : ADDRESS CURRENT PACKET
53 3A A3 9B 065B 1279 MOVZBW IRP$L_MEDIA+2(R3),R3 : MAKE THE ESCAPE SEQUENCE COUNT A WORD
3C A4 53 A2 065F 1280 SUBW R3,TTY$W_RB_TXTOFF(R4) : SUBTRACT OUT TERMINATOR LENGTH
7C A5 3C B0 0663 1281 15$: IF NOT_STATE BADESC,20$ : ESCAPE SYNTAX CORRECT?
0667 1282 MOVW #SS$_BADESCAPE,UCBSW_BOFF(R5); SET STATUS FOR IMPROPER ESCAPE SEQ
066B 1283 :
066B 1284 : RESET PASSALL AND NOECHO IF MODES
066B 1285 :
066B 1286 20$:
066B 1287 :
066B 1288 .IF DF CAS_MEASURE_IOT
066B 1289 :
066B 1290 BLBC G^PM$GL_DOSTATS,25$ : IF FLAG SET, BYPASS STATISTICS CODE
066B 1291 BBC #TT$V_PASSALL,UCBSL_DEVDEPEND(R5),25$; BR IF NOT PASSALL
066B 1292 INCL G^PM$GL_PASSALL : INCREMENT PASSALL COUNT
066B 1293 :
066B 1294 .ENDC
066B 1295 :
54 44 A5 FC 8F 8B 066B 1296 25$: BICB3 #C<TT$M_PASSALL!TT$M_NOECHO>,UCBSL_DEVDEPEND(R5),R4;
04 A2 02 02 54 FO 0671 1297 INSV R4,#TT$V_ST_PASSALL,#2,4(R2) : RESET PASSALL AND NOECHO
54 44 A5 01 03 EE 0677 1298 EXTW #TT$V_ESCAPE,#1,UCBSL_DEVDEPEND(R5),R4 : GET CURRENT SETTING

```

```

04 A2 01 0B 54 F0 067D 1299      INSV   R4,#TTY$V ST_ESCAPE,#1,4(R2)      ; UPDATE IN STATE
      04 48 A5 12 E1 0683 1300      BBC    #T12$V PASTHRU,UCBSL_DEVDEPND2(R5),98$ ; IN PASS THRU MODE
      0688 1301      SET_STATE PASACL
      068C 1302 98$:
      068C 1303
      068C 1304      CLR_STATE - ; Clear state bits relevant to
      068C 1305      <READ,DEL,EOL,- ; a read (this does not include
      068C 1306      PROMPT,CTRLR,NOFLTR,- ; the write bits).
      068C 1307      ESC,ESC O,BADESC,PRE,TERMNORM,-
      068C 1308      REFRSH,EDITREAD,SKIPCRLF,RDVERIFY,ECHAES,-
      068C 1309      MULTI,RECALL,OVERSTRIKE,EDITING,QUOTING,BACKSPACE>
      53 58 A5 D0 0699 1310      MOVL   UCBSL_IRP(R5),R3 ; ADDRESS CURRENT PACKET
      54 78 A5 D0 069D 1311      MOVL   UCBSL_SVAPTE(R5),R4 ; GET THE ADDRESS OF THE READ PACKET
012B C5 3A A4 90 06A1 1312      MOVSB  TTY$W_RB_CPZORG(R4),UCBSB_TT_OLDCPZORG(R5); SAVE BECAUSE
      06A7 1313      ; WE MAY CONTINUE THIS READ
      3C A3 0C A3 D0 06A7 1314      MOVL   IRPSL_PID(R3),IRPSL_TT_TERM(R3); SAVE DATA FOR FORK BLOCK
      4C A3 0B A3 90 06AC 1315      MOVSB  IRPSB_RMOD(R3),IRPSB_TT_PRMT(R3);
      0B A3 0B A5 90 06B1 1316      MOVSB  UCBSB_FIPL(R5),IRPSB_RMOD(R3); SET FORK IPL
      30 A3 7C A5 B0 06B6 1317      MOVWB  UCBSW_BOFF(R5),IRPSW_BOFF(R3); SAVE STATUS
      32 A3 3C A4 B0 06BB 1318      MOVWB  TTY$W_RB_TXTOFF(R4),IRPSW_BCNT(R3); SAVE TRANSFER SIZE
      55 53 24 BB 06C0 1319      PUSHR  #*M<R2,R5>
      53 10 A3 7D 06C5 1321      MOVL   R3,R5 ; SET UP FORK BLOCK ADDRESS
      FE7B CF 9F 06C9 1322      MOVQB  IRPSL_AST(R3),R3 ; GET PARAMS FOR FORK
      3C A5 DD 06CD 1323      PUSHAB W*TTY$GETNXTWRITE ; FORCE RETURN TO RESTORE UCB ADDRESS
      00000000'EF 16 06D0 1324      PUSHL  IRPSL_TT_TERM(R5) ; USE THE PID OF THE ISSUING PROCESS
      06D6 1325      JSB    TTY$SYNCR ; CREATE FORK PROCESS
      06D6 1326      ; I/O DONE FORK PROCESS
      53 55 D0 06D6 1328      MOVL   R5,R3 ; RESTORE PACKET AND UCB ADDRESSES
      06D9 1329
      06D9 1330      .IF DF CAS_MEASURE_IOT
      06D9 1331      ; ACCUMULATE STATISTICS ON NUMBER OF CHARACTERS AND I/OS TO TERMINALS.
      06D9 1332
      06D9 1333      BSBW  TTSTATS ; CALL STATISTICS ROUTINE.
      06D9 1334
      06D9 1335      .ENDC
      55 1C A3 D0 06D9 1338      MOVL   IRPSL_UCB(R3),R5 ; GET LUCB ADDRESS FROM IRP
      55 00A0 C5 D0 06DD 1339      MOVL   UCBSL_TL_PHYUCB(R5),R5 ; SWITCH TO PHYSICAL USB CONTEXT
      50 30 A3 D0 06E2 1340      MOVL   IRPSW_BOFF(R3),R0 ; GET STATUS AND TRANSFER SIZE
      54 3A A3 9B 06E6 1341      MOVZBW IRPSL_MEDIA+2(R3),R4 ; GET THE BYTE COUNT OF THE TERMINATOR
      32 A3 54 A0 06EA 1342      ADDW   R4,IRPSW_BCNT(R3) ; CALC TOTAL TRANSFER SIZE
      54 2C A3 D0 06EE 1343      MOVL   IRPSL_SVAPTE(R3),R4 ; GET ADDRESS OF THE BUFFER BLOCK
      21 20 A3 0F E1 06F2 1344      BBC    #IOSV_EXTEND,IRPSW_FUNC(R3),27$
      39 A3 FF 8F 90 06F7 1345      MOVSB  #*XOFF,IRPSL_MEDIA+1(R3); INFORCE RESERVED FIELD
      17 40 A3 0A E0 06FC 1346      BBS    #TTY$V ST_RDVERIFY,IRPSB_TT_STATE(R3),27$; DON'T DO IT FOR READ VERI
      51 30 A4 3C 0701 1347      MOVZWL TTY$W_RB_CINOFF(R4),R1 ; get the offset to this character
      2C A4 51 C0 0705 1348      ADDL   R1,TTY$W_RB_LIN(R4) ; ADD IN THE OFFSET TO GET AN ADDRESS
      51 3C A4 3C 0709 1349      MOVZWL TTY$W_RB_TXTOFF(R4),R1 ; GET THE LENGTH OF THE READ
      51 51 64 C0 070D 1350      ADDL   TTY$W_RB_TXT(R4),R1 ; AND GET THE ADDRESS OF THE LAST CHARACTER
      51 2C A4 C2 0710 1351      SUBL   TTY$W_RB_LIN(R4),R1 ; SUBTRACT OUT THE ADDRESS OF THE
      3B A3 51 90 0714 1352      MOVSB  R1,IRPSL_MEDIA+3(R3) ; LAST CHARACTER AND THIS IS THE
      0718 1353      ; OFFSET FROM THE END OF THE READ
      0718 1354      ; TO THE LAST POSITION.
      51 38 A3 D0 0718 1355 27$:      MOVL   IRPSL_MEDIA(R3),R1 ; SET TERMINATOR DATA

```


TTYSTRSTP
V04-000

- Terminal driver start/stop I/O routine 16-SEP-1984 02:18:30 VAX/VMS Macro V04-00 Page 35
TTY\$READONE - READ OPERATION DONE 5-SEP-1984 04:17:09 [TTDRVR.SRC]TTYSTRSTP.MAR;1 (24)

07A3 1408
07A3 1409 .SBTTL End of module
07A3 1410
07A3 1411 .END

TT
VC

TTYSTRSTP
Symbol table

CLASS_MODEM_DIS	*****	X	02	SS\$ ABORT	= 0000002C		
CNT	= 00000001			SS\$ BADESCAPE	= 0000003C		
COM\$POST	*****	X	02	SS\$ NOPRIV	= 00000024		
DO_CONNECT	00000037	R	02	SS\$ NORMAL	= 00000001		
DO_DISCONNECT	000000A4	R	02	SS\$ NOSUCHDEV	= 00000908		
DO_EXIT	00000422	R	02	SS\$ PARTESCAPE	= 000001FC		
DO_HANGUP	000000B5	R	02	SS\$ TIMEOUT	= 0000022C		
DO_MAINT	000000BF	R	02	START_PKT	000004E6	R	02
DO_READ	0000010D	R	02	T	= 00000005		
DO_SET	000001C1	R	02	TRANSITION NOCHECK	*****	X	02
DO_SETC	000001B5	R	02	TT\$C_BAUD 9600	= 0000000F		
DO_SETM	000001B2	R	02	TT\$M_DS_DTR	= 00000002		
DO_WRITE	00000444	R	02	TT\$M_DS_RTS	= 00000010		
END_FALL	00000735	R	02	TT\$M_DS_SECTX	= 00000008		
F	= 00000000			TT\$M_MBX_DSABL	= 00010000		
FIND BOL NOCLEAR	*****	X	02	TT\$M_MODEM	= 00200000		
INSERT_PKT	000004D9	R	02	TT\$M_NOBRDCST	= 00020000		
IOSM_FCODE	= 0000003F			TT\$M_NOECHO	= 00000002		
IOSV_BREAKTHRU	= 00000009			TT\$M_ODD	= 00000080		
IOSV_CANCTRLO	= 00000006			TT\$M_PARITY	= 00000040		
IOSV_ENABLMBX	= 00000007			TT\$M_PASSALL	= 00000001		
IOSV_EXTEND	= 0000000F			TT\$M_REMOTE	= 00002000		
IOSV_LOOP	= 00000007			TT\$V_ALTDISPAR	= 0000000A		
IOSV_LOOP_EXT	= 0000000C			TT\$V_ALTFRAME	= 00000004		
IOSV_NEWLINE	= 0000000A			TT\$V_ALTRPAR	= 00000005		
IOSV_PURGE	= 0000000B			TT\$V_CRFILL	= 0000000A		
IOSV_SET MODEM	= 0000000A			TT\$V_DISPARRR	= 00000009		
IOSV_TT DISCON	= 0000000C			TT\$V_EIGHTBIT	= 0000000F		
IOCSREQCOM	*****	X	02	TT\$V_ESCAPE	= 00000003		
IPL\$ QUEUEAST	= 00000006			TT\$V_LFFILL	= 0000000B		
IRP\$B_RMOD	= 0000000B			TT\$V_MODEM	= 00000015		
IRP\$L_ARB	= 00000058			TT\$V_PARITY	= 00000006		
IRP\$L_AST	= 00000010			TT\$V_PASSALL	= 00000000		
IRP\$L_IOST1	= 00000038			TT\$V_READSYNC	= 00000012		
IRP\$L_IOST2	= 0000003C			TT\$V_REMOTE	= 0000000D		
IRP\$L_MEDIA	= 00000038			TT\$V_TTSYNC	= 00000005		
IRP\$L_PID	= 0000000C			TT\$V_TWOSTOP	= 00000008		
IRP\$L_SVAPTE	= 0000002C			TT2\$M_ALTYPEAHD	= 00000080		
IRP\$L_TT TERM	= 0000003C			TT2\$M_DCL MAILBX	= 00000200		
IRP\$L_UCB	= 0000001C			TT2\$M_DISCONNECT	= 00020000		
IRP\$L_VALS	= 0000009C			TT2\$M_DMA	= 00000040		
IRP\$Q_TT STATE	= 00000040			TT2\$M_XON	= 00000020		
IRP\$S_FCODE	= 00000006			TT2\$V_ALTYPEAHD	= 00000007		
IRP\$V_FCODE	= 00000000			TT2\$V_AUTOBAUD	= 00000001		
IRP\$W_BCNI	= 00000032			TT2\$V_DISCONNECT	= 00000011		
IRP\$W_BOFF	= 00000030			TT2\$V_DMA	= 00000006		
IRP\$W_FUNC	= 00000020			TT2\$V_EDITING	= 0000000C		
IRP\$W_TT PRMPT	= 0000004C			TT2\$V_FALLBACK	= 0000000E		
MODEM\$C_INIT	= 00000000			TT2\$V_HANGUP	= 00000002		
MODEM\$C_SHUTDWN	= 00000001			TT2\$V_LOCALECHO	= 00000000		
NOPRIV_EXIT	0000043C	R	02	TT2\$V_MODHANGUP	= 00000003		
PORT_STARTIO	= 00000000			TT2\$V_PASTHRU	= 00000012		
PRV\$V_LOG_IO	= 00000007			TT2\$V_SECURE	= 00000010		
PRV\$V_PHY_IO	= 00000016			TT2\$V_SETSPEED	= 00000008		
QUEUE_LAST	000004D5	R	02	TT2\$V_XON	= 00000005		
QUEUE_PKT	000004BC	R	02	TTY\$A_INPFALL	*****	X	02
READ\$DONE	00000730	R	02	TTY\$A_TA_RCL	= 00000018		

TTYSTRSTP
Symbol table

```

TTYSB_WB_FIPL      = 0000000B
TTYSRE_FORK        = ***** X 02
TTYSB_CR           = 0000000D
TTYSB_FC_SETC     = 00000003
TTYSB_DONE        = 00000735 R 02
TTYSB_DS_SET      = ***** X 02
TTYSB_GETNEXTCHAR = ***** X 02
TTYSB_GETNXTWRITE = 00000548 RG 02
TTYSB_ER_ECHLINE  = 00000002
TTYSB_TA_RCLLEN   = 00000100
TTYSB_LOCK        = ***** X 02
TTYSB_RB_LIN      = 0000002C
TTYSB_RB_TXT      = 00000000
TTYSB_WB_BLINK    = 00000004
TTYSB_WB_END      = 00000020
TTYSB_WB_FLINK    = 00000000
TTYSB_WB_FR3      = 00000010
TTYSB_WB_IRP      = 00000024
TTYSB_WB_RETADDR  = 0000002C
TTYSB_MAINT       = ***** X 02
TTYSB_PC_DMAENA   = 00000002
TTYSB_PC_XOFENA   = 00000040
TTYSB_ST_BACKSPACE = 00000020
TTYSB_ST_BADESC   = 00000100
TTYSB_ST_CTRL0    = 00000001
TTYSB_ST_CTRLR    = 00040000
TTYSB_ST_DEL      = 00000002
TTYSB_ST_ECHAES   = 02000000
TTYSB_ST_EDITING  = 00100000
TTYSB_ST_EDITREAD = 00000200
TTYSB_ST_EOL      = 00000100
TTYSB_ST_ESC      = 00000080
TTYSB_ST_ESC_O    = 00004000
TTYSB_ST_MULTI    = 00000040
TTYSB_ST_NL       = 00000200
TTYSB_ST_NOECHO   = 00000008
TTYSB_ST_NOFLTR   = 00000040
TTYSB_ST_OVERSTRIKE = 00800000
TTYSB_ST_PASALL   = 00000004
TTYSB_ST_PRE      = 04000000
TTYSB_ST_PROMPT   = 00000020
TTYSB_ST_QUOTING  = 00400000
TTYSB_ST_RDVERIFY = 00000400
TTYSB_ST_READ     = 00001000
TTYSB_ST_RECALL   = 00000800
TTYSB_ST_RECONNECT = 10000000
TTYSB_ST_REFRSH   = 00000400
TTYSB_ST_SEDLF    = 00000010
TTYSB_ST_SKIPCRLF = 00080000
TTYSB_ST_SkipLF   = 00002000
TTYSB_ST_TERMORM  = 01000000
TTYSB_ST_WRAP     = 00008000
TTYSB_ST_WRITE    = 00000080
TTYSB_ST_WRTALL   = 00000010
TTYSB_PORGE_AHEAD = ***** X 02
TTYSB_READONE     = 0000063D RG 02
TTYSB_RESTARTIO   = ***** X 02

```

```

TTYSB_RESUME      = ***** X 02
TTYSB_SETUP_READ  = ***** X 02
TTYSB_SETUP_LINE  = ***** X 02
TTYSB_STARTIO     = 00000000 RG 02
TTYSB_STARTOUTPUT = 00000532 RG 02
TTYSB_SYNC        = ***** X 02
TTYSB_FD_DISCONNECT = 00000002
TTYSB_FD_GETAHD   = 00000001
TTYSB_PC_DMAAVL   = 00000002
TTYSB_PC_NOCRLF   = 00000007
TTYSB_ST_ESCAPE   = 0000000B
TTYSB_ST_PASALL   = 00000002
TTYSB_ST_RDVERIFY = 0000000A
TTYSB_ST_BACKSPACE = 00000005
TTYSB_ST_BADESC   = 00000028
TTYSB_ST_CTRL0    = 00000020
TTYSB_ST_CTRLR    = 00000032
TTYSB_ST_DEL      = 00000021
TTYSB_ST_ECHAES   = 00000039
TTYSB_ST_EDITING  = 00000034
TTYSB_ST_EDITREAD = 00000009
TTYSB_ST_EOL      = 00000008
TTYSB_ST_ESC      = 00000027
TTYSB_ST_ESC_O    = 0000002E
TTYSB_ST_MULTI    = 00000006
TTYSB_ST_NL       = 00000029
TTYSB_ST_NOECHO   = 00000023
TTYSB_ST_NOFLTR   = 00000026
TTYSB_ST_OVERSTRIKE = 00000037
TTYSB_ST_PASALL   = 00000022
TTYSB_ST_PRE      = 0000003A
TTYSB_ST_PROMPT   = 00000025
TTYSB_ST_QUOTING  = 00000036
TTYSB_ST_RDVERIFY = 0000000A
TTYSB_ST_READ     = 0000000C
TTYSB_ST_RECALL   = 0000000B
TTYSB_ST_RECONNECT = 0000003C
TTYSB_ST_REFRSH   = 0000002A
TTYSB_ST_SEDLF    = 00000004
TTYSB_ST_SKIPCRLF = 00000033
TTYSB_ST_SkipLF   = 0000002D
TTYSB_ST_TERMORM  = 00000038
TTYSB_ST_WRAP     = 0000002F
TTYSB_ST_WRITE    = 00000007
TTYSB_ST_WRTALL   = 00000024
TTYSB_WRTEDONE    = 00000587 RG 02
TTYSB_WRITEPOST   = 0000062A RG 02
TTYSB_WRTSTARTIO  = 00000452 RG 02
TTYSB_RB_CPZORG   = 0000003A
TTYSB_RB_LINOFF   = 00000030
TTYSB_RB_LINREST  = 00000032
TTYSB_RB_MODE     = 00000044
TTYSB_RB_SIZE     = 00000008
TTYSB_RB_TXTOFF   = 0000003C
TTYSB_TA_RCLSZ    = 00000014
TTYSB_WB_BCNT     = 0000002A
TTYSB_WB_SIZE     = 00000008

```


TTYSTRSTP
Symbol table

TTY\$W_WB_STATUS	= 00000028		X1	= 00000002
TTY\$XOFF	= *****	X 02	Z0	= 00000000
UCBSB_DEVCLASS	= 00000040		Z1	= 00000000
UCBSB_DEVTYPE	= 00000041			= 00000000
UCBSB_FIPL	= 0000000B			
UCBSB_TT_CRFILL	= 000000F6			
UCBSB_TT_DEPARI	= 000000EC			
UCBSB_TT_DETYPE	= 000000F0			
UCBSB_TT_LASTC	= 000000FF			
UCBSB_TT_LFFILL	= 000000F7			
UCBSB_TT_LINE	= 000000FE			
UCBSB_TT_MAINT	= 0000012A			
UCBSB_TT_OLDCPZORG	= 0000012B			
UCBSB_TT_OUTYPE	= 0000010B			
UCBSB_TT_PARITY	= 000000F8			
UCBSL_DEVDEPEND	= 00000044			
UCBSL_DEVDEPND2	= 00000048			
UCBSL_IRP	= 00000058			
UCBSL_PDT	= 00000084			
UCBSL_SVAPTE	= 00000078			
UCBSL_TL_PHYUCB	= 000000A0			
UCBSL_TT_DECHA1	= 000000C8			
UCBSL_TT_DECHAR	= 000000C4			
UCBSL_TT_LOGUCB	= 000000C0			
UCBSL_TT_PORT	= 00000118			
UCBSL_TT_TYPAHD	= 000000E4			
UCBSL_TT_WBLINK	= 000000D0			
UCBSL_TT_WFLINK	= 000000CC			
UCBSL_TT_WRTBUF	= 000000D4			
UCBSM_CANCEL	= 00000008			
UCBSM_TIMEOUT	= 00000040			
UCBSM_TT_DISPARERR	= 00000002			
UCBSM_TT_DSBL	= 00000080			
UCBSM_TT_LEN	= 00000018			
UCBSM_TT_NOTIF	= 00000004			
UCBSM_TT_ODD	= 00000080			
UCBSM_TT_PARTY	= 00000040			
UCBSM_TT_STOP	= 00000020			
UCBSM_TT_TIMO	= 00000002			
UCBSV_INT	= 00000001			
UCBSV_TT_DISPARERR	= 00000001			
UCBSV_TT_LEN	= 00000003			
UCBSV_TT_USERFRAME	= 00000002			
UCBSW_BOFF	= 0000007C			
UCBSW_DEVBUFSIZ	= 00000042			
UCBSW_DEVSTS	= 00000068			
UCBSW_STS	= 00000064			
UCBSW_TT_CURSOR	= 000000FC			
UCBSW_TT_DESPEE	= 000000E8			
UCBSW_TT_PRTCTL	= 00000122			
UCBSW_TT_SPEED	= 000000F4			
W0	= 00001F60			
W1	= 07DC45E2			
WRITEPOST	= 000005AB	R	02	
WRTSTARTIO	= 00000491	R	02	
X	= 000007DC			
XO	= 00000000			

! 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
\$\$\$115_DRIVER	000007A3 (1955.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.03	00:00:00.85
Command processing	118	00:00:00.33	00:00:03.16
Pass 1	611	00:00:18.41	00:01:04.16
Symbol table sort	0	00:00:02.66	00:00:10.55
Pass 2	264	00:00:03.99	00:00:13.95
Symbol table output	34	00:00:00.19	00:00:00.19
Psect synopsis output	2	00:00:00.02	00:00:00.67
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1066	00:00:25.65	00:01:33.54

The working set limit was 2100 pages.
152906 bytes (299 pages) of virtual memory were used to buffer the intermediate code.
There were 130 pages of symbol table space allocated to hold 2384 non-local and 89 local symbols.
1411 source lines were read in Pass 1, producing 18 object records in Pass 2.
56 pages of virtual memory were used to define 53 macros.

! Macro library statistics !

Macro library name	Macros defined
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	22
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	10
TOTALS (all libraries)	32

2769 GETS were required to define 32 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:TTYSTRSTP/OBJ=OBJ\$:TTYSTRSTP MSRC\$:TTYSTRSTP/UPDATE=(ENH\$:TTYSTRSTP)+EXECMLS/LIB

0404 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

The image displays a grid of 144 terminal windows arranged in 12 rows and 12 columns. Each window contains text-based data, likely system logs or diagnostic messages. The text is dense and appears to be a mix of system status reports, error codes, and file names. Some windows are larger than others, suggesting they are active or selected. The overall appearance is that of a multi-user terminal session on a VAX/VMS system. Several windows are clearly labeled with titles such as:

- TTYSYNCH LIS (top right)
- TTYSTRSTP LIS (middle left)
- TTYSUB LIS (middle bottom)
- YCDTRVER LIS (bottom right)

The text within the windows is too small to read in detail, but it appears to be a mix of system status reports, error codes, and file names. The overall appearance is that of a multi-user terminal session on a VAX/VMS system.