


```

LL          TTTTTTTTTT DDDDDDDD RRRRRRRR I I I I I VV      VV  EEEEEEEEE E RRRRRRRR
LL          TTTTTTTTTT DDDDDDDD RRRRRRRR I I I I I VV      VV  EEEEEEEEE E RRRRRRRR
LL          TT          DD      DD  RR      RR  I I      I I  EE          RR      RR
LL          TT          DD      DD  RR      RR  I I      I I  EE          RR      RR
LL          TT          DD      DD  RR      RR  I I      I I  EE          RR      RR
LL          TT          DD      DD  RRRRRRRR I I      I I  EEEEEEEEE RRRRRRRR
LL          TT          DD      DD  RRRRRRRR I I      I I  EEEEEEEEE RRRRRRRR
LL          TT          DD      DD  RR      RR  I I      I I  EE          RR      RR
LL          TT          DD      DD  RR      RR  I I      I I  EE          RR      RR
LL          TT          DD      DD  RR      RR  I I      I I  EE          RR      RR
LL          TT          DD      DD  RR      RR  I I      I I  EE          RR      RR
LLLLLLLLLL TT          DDDDDDDD RR      RR  I I I I I VV      VV  EEEEEEEEE RRR      RR
LLLLLLLLLL TT          DDDDDDDD RR      RR  I I I I I VV      VV  EEEEEEEEE RRR      RR

```

```

LL          I I I I I SSSSSSSS
LL          I I I I I SSSSSSSS
LL          I I      SS
LL          I I      SS
LL          I I      SS
LL          I I      SSSSSS
LL          I I      SSSSSS
LL          I I      SS
LL          I I      SS
LL          I I      SS
LL          I I      SS
LLLLLLLLLL I I I I I SSSSSSSS
LLLLLLLLLL I I I I I SSSSSSSS

```

L
V

(4)	179	Global Definitions
(5)	216	Local Definitions
(18)	813	Driver Prologue Table
(19)	879	Local Storage
(22)	1025	Port Vector
(25)	1142	LT\$CTRL INIT - Controller Initialization
(26)	1245	LT\$HISTORY - Save history in the buffer
(27)	1297	LT\$STRETRY - Entry to start driver
(28)	1371	LT\$GETFFI - Get the FFI block and initialize FFI interface
(29)	1438	LT\$SHUTENTRY - Shutdown entry point
(30)	1570	TQE TIMER BUILDING AND WAITING ROUTINES
(31)	1636	LT\$SHUT DONE - Datalink shutdown complete entry
(32)	1660	LT\$ASYNCERR - Process async errors from datalink
(33)	1672	LT\$SETENTRY - Set a new multicast message
(34)	1768	LT\$UNIT INIT - Unit Initialization
(35)	1836	LT\$UCB INIT - Initialize a terminal UCB
(36)	1875	LT\$STARTIO - Start I/O routine
(37)	2127	LT\$FLUSH_DATA - Flush buffered output data and drain TT buffer
(38)	2150	LT\$ABORT - Abort (flush) buffered output data
(38)	2176	LT\$FLOW_CHANGE - Change flow control
(39)	2197	LT\$XON - Resume input stream into class driver
(39)	2247	LT\$XOFF - Stop input stream into class driver
(40)	2306	LT\$FFI_RCV_MSG - FAST Interface Receive Complete routine
(41)	2491	PROCESS RECEIVED MESSAGE
(42)	2554	CIRCUIT STATE TRANSITIONS
(43)	2613	PROCESS RECEIVED SLOT DATA
(44)	2737	PROCESS SLOT CREDITS RECEIVED
(45)	2837	BUILD A STOP SLOT
(46)	2882	PROCESS OUTPUT SLOT DATA
(48)	3138	CONSUME CREDIT WHEN FILLING IN SLOT FIELD
(49)	3222	SEND REPLY RESPONSE TO RECEIVED MESSAGE
(50)	3297	LT\$CREATECSB - Create a CSB
(51)	3350	LT\$NEW_CIRCUIT - Get Concentrator State Block
(52)	3555	LT\$DEALLOCBSB - Deallocate a CSB
(53)	3661	LT\$CREATEUCB - Create an LT UCB
(54)	3699	LT\$DESTROYUCB - Destroy an LT UCB
(55)	3747	LT\$ALC_UCB - Allocate New LT UCB
(56)	3894	LT\$HANGUP_UCB - Cause Terminal Hangup On UCB
(57)	3937	LT\$DISCONNECT - Port Disconnect
(58)	4001	LT\$KILLUCB - Delete a dried up UCB
(59)	4043	LT\$SIDELINEUCB - Set a UCB aside
(60)	4074	LT\$CIRCDEAD - Declare circuit dead
(61)	4113	LT\$STOPCIRC - Transmit stop message
(62)	4195	LT\$SET_TIMER - Start Timer
(63)	4259	LT\$TICK - Timer Service Routine
(64)	4476	LT\$XMIT - Transmit a message
(65)	4612	LT\$XMT_FFIDONE - FFI Transmit done routine
(66)	4681	LT\$FFIPOSTDONE - Garbage transmit posting routine

;JAY0002
-1

```

0000 1 .TITLE LTDRIVER - Local Area Terminal Port Driver
0000 .1 .IDENT 'V04-002'
0000 3
0000 4 :*****
0000 5 :*
0000 6 :* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7 :* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8 :* ALL RIGHTS RESERVED.
0000 9 :*
0000 10 :* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 :* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 :* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 :* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 :* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 :* TRANSFERRED.
0000 16 :*
0000 17 :* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 :* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 :* CORPORATION.
0000 20 :*
0000 21 :* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 :* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23 :*
0000 24 :*
0000 25 :*****
0000 26
0000 27 : Do not remove items from this list. Add X when item is done.
0000 28 : to be done:
0000 29 :
0000 30 : - LATCP should be allowed to specify the Ethernet device.
0000 31 : - fix slot stop and circuit stop code for correct message format
0000 32 : including reason code and text.
0000 33 : - transmit two slots/message from host transmit loop.
0000 34 : - allocate and be able to send additional buffers based about the servers
0000 35 : value of XTRABFRs (in START message).
0000 36 : - send credits in DATA_B slots.
0000 37 :

```



```

0000 39
0000 40
0000 41 :*****
0000 42 :*
0000 43 :* Copyright (c) 1982, 1984
0000 44 :* by digital equipment corporation, maynard, mass.
0000 45 :*
0000 46 :* this software is furnished under a license and may be used and copied
0000 47 :* only in accordance with the terms of such license and with the
0000 48 :* inclusion of the above copyright notice. this software or any other
0000 49 :* copies thereof may not be provided or otherwise made available to any
0000 50 :* other person. no title to and ownership of the software is hereby
0000 51 :* transferred.
0000 52 :*
0000 53 :* the information in this software is subject to change without notice
0000 54 :* and should not be construed as a commitment by digital equipment
0000 55 :* corporation.
0000 56 :*
0000 57 :* digital assumes no responsibility for the use or reliability of its
0000 58 :* software on equipment which is not supplied by digital.
0000 59 :*
0000 60 :*****
0000 61
0000 62 +-
0000 63 Facility:
0000 64
0000 65 VAX/VMS Terminal Driver
0000 66
0000 67 Abstract:
0000 68
0000 69 Local Area Terminal Port Driver
0000 70
0000 71 This is an alternate port driver for use with the VMS terminal
0000 72 driver. It forms the host end of the Local Area Terminal (LAT)
0000 73 protocol and communicates with a LAT concentrator to provide
0000 74 terminal communications on a local area network.
0000 75
0000 76 This driver performs IO using a datalink driver via the FFI
0000 77 mechanism. Its connection with the datalink driver is controlled
0000 78 by a program called LATCP.
0000 79
0000 80
0000 81
0000 82 Authors:
0000 83
0000 84 Bruce Mann
0000 85 Darrell Duffy
0000 86 Joe Marchesani
0000 87 Rich Bailey
0000 88
0000 89 Revision history:
0000 90
0000 .1 V04-002 JAY0002 John A. Ywoskus 17-Dec-1984
0000 .2 Add CPU dispatch table to determine max slots/circuit.
0000 .3 Fix odd length reject slot format.
0000 .4
0000 .5 V04-001 JAY0001 John A. Ywoskus 28-Sep-1984

```

: JAY0002
: JAY0002
: JAY0002
: JAY0002
: JAY0001

LTDRIVER
V04-002

- Local Area Terminal Port Driver I 14

8-JAN-1985 17:22:25 VAX/VMS Macro V04-00
5-SEP-1984 11:40:22 [LAT.BUGSRC]LTDRIVER.MAR;1

Page 3
(3)

:JAY0001
:JAY0001
:JAY0001
:JAY0001
-85

0000 .6 :
0000 .7 :
0000 .8 :
0000 .9 :
0000 176 :
0000 177 :--

Add code to not set any flow control characters if
terminal is set /PASSALL.
Add MicroVAX II to CPU RATING.
Mark terminals OFFLINE when called at DISCONNECT entry.

```
0000 179      .SBTTL Global Definitions
0000 180
0000 181      .ENABLE SUPPRESSION
0000 182
0000 183      :
0000 184      :
0000 185      :
0000 186
0000 187      $LATDEF      ; LAT communication area defs
0000 188      $ACBDEF      ; Ast control block
0000 189      $ADPDEF      ; Define adapter
0000 190      $CRBDEF      ; Define crb
0000 191      $CXBDEF      ; Complex buffer
0000 192      $DDBDEF      ; Define ddb
0000 193      $DEVDEF      ; Define device characteristics
0000 194      $DPTDEF      ; Define dpt
0000 195      $DYNDDEF     ; Dynamic memory block codes
0000 196      $FFIDEF      ; Fast Interface block
0000 197      $FKBDEF      ; Fork block definitions
0000 198      $IPLDEF      ; IPL levels
0000 199      $MTXDEF      ; Mutex offsets
0000 200      $ORBDEF      ; Object's Rights Block offsets
0000 201      $PCBDEF      ; Process control block
0000 202      $PHDDEF      ; Process header block
0000 203      $PRDEF      ; Processor register definitions
0000 204      $RSNDEF      ; Resouce wait codes
0000 205      $TQDEF      ; Timer queue elements
0000 206      $UCBDEF      ; Define ucb
0000 207      $VECDEF      ; Define vector for crb
0000 208
0000 209      $TTYDEF      ; Define terminal driver symbols
0000 210      $TIDEF      ; Define terminal types
0000 211      $TT2DEF      ; Define extended characteristics
0000 212      $TTYMACS     ; Define terminal driver macros
0000 213      $TTYDEFS     ; Define terminal driver symbols
0000 214
```

```

0000 216 .SBTTL Local Definitions
0000 217 :
0000 218 :
0000 219 :
0000 220 .MACRO ALIGN_LONG
0000 221 .iif ne <.&3>, .BLKB <4-<.&3>> ; align to longword boundary, if needed
0000 222 .ENDM ALIGN_LONG
0000 223 :
0000 224 :
0000 225 :
0000 226 :
0000 227 :
00000001 0000 228 LT_HISTORY = 1 ; keep history of received/transitted ni fra
0000 229 ::&& LT_CRED_CHECK = 1 ; validate credit total for UCBs
0000 230 ::&& LT_XMT_CHECK = 1 ; validate transmit frame slots dynamically
0000 231 :
0000 232 :
0000 233 :
0000 234 :
00000007 0000 235 TTY_C_BELL = 7 ; Control_G (^G) ... bell character
00000003 0000 236 LATSC_VMS = 3 ; VAX/VMS product type code
00000004 0000 237 LATSC_VMS_VER = 4 ; Part of VAX/VMS V4.0
00000008 0000 238 LATSC_IPL = 8 ; Fork IPL to synchronize this driver
00000014 0000 239 LATSC_PROGRESS = 20 ; Must make progress every 20 msgs
00000002 0000 240 LATSC_XMT_BUFFERS = 2 ; Number of transmit buffers
00000001 0000 241 LATSC_MAX_RCRED = 1 ; Maximum number of credits extended
00000050 0000 242 LATSC_UCB_BUFSIZ = 80 ; Size of UCB buffering
00000004 0000 243 LATSC_UCB_SLOTSIZ = 4 ; Size of formatted slot buffer in UCB
00000002 0000 244 LATSC_UCB_TIMEOUT = 2 ; Allow 2 seconds before deleting UCB's
00000020 0000 245 LATSC_MAX_CSBS = 32 ; Maximum number of circuits handled
00000040 0000 246 LATSC_MAX_SLOTS = 64 ; Maximum slots in a message, this also
0000 247 ; implies max connections per circuit
00000005 0000 248 LATSC_HI_VER = 5 ; Highest supported LAT protocol version
00000005 0000 249 LATSC_LO_VER = 5 ; Lowest supported LAT protocol version
00000005 0000 250 LATSC_CUR_VER = 5 ; Current LAT protocol version
00000000 0000 251 LATSC_CUR_ECO = 0 ; Current LAT protocol ECO version
000005DC 0000 252 LATSC_MAX_MSGSIZ = 1500 ; Maximum Data Link message size
000000FF 0000 253 LATSC_MAX_SLOTSIZ = 255 ; Maximum Slot size
00000040 0000 254 LATSC_LOC_LEN = 64 ; Maximum length of LOcAtion field
00000020 0000 255 LATSC_GRP_LEN = 32 ; Maximum number of group codes
00000008 0000 256 LATSC_SVC_LEN = 8 ; Maximum number of service classes
00000006 0000 257 LATSC_MAX_SERVICES = 6 ; Maximum number of services
00000009 0000 258 LATSC_MULTICAST1 = ^X0009 ; LAT Multicast address
0000002B 0000 259 LATSC_MULTICAST2 = ^X002B ;
00000F00 0000 260 LATSC_MULTICAST3 = ^X0F00 ;
00000002 0000 261 LATSC_HOST_TIMER = 2 ; Host circuit timeout value (in SEcs)
0000000A 0000 262 LATSC_MULTICAST_TIMER = 10 ; Multicast message timer
00000050 0000 263 LATSC_CIR_TIMER = 80 ; Host preferred circuit timer (in MS)
004C4B40 0000 264 LATSC_SHUTDELAY = 5*1000*1000 ; Delay time = 1/2 second
0000000A 0000 265 LATSC_KEEP_CSBS = 10 ; Keep last 10 CSBs
0000 266 :
0000 267 :
0000 268 :
0000 269 :
0000 270 :
00000134 0000 271 $DEFINI UCB GLOBAL
00000134 0000 272 . = UCBSC_TT_LENGTH

```

```

0134 273 ALIGN_LONG ; align to longword boundary
0134 274 $DEF UCBSL_LT_CSB .BLKL 1 ; CSB address
0138 275 $DEF UCBSL_LT_DEADLINK .BLKL 1 ; Link for dead ucbs
013C 276 $DEF UCBSL_LT_INPBUF .BLKL 1 ; Pointer to LAT type ahead buffer
0140 277 $DEF UCBSB_LT_REMID .BLKB 1 ; Remote slot index number
0141 278 $DEF UCBSB_LT_LOCID .BLKB 1 ; Local slot index number
0142 279 $DEF UCBSB_LT_DATAW .BLKB 1 ; Data waiting flags
0143 280 $VIELD UCB,0,<-
0143 281 <LT_DATA,1,M>- ; * Output data waiting
0143 282 <LT_FLOW,1,M>- ; * Send current flow control values
0143 283 <LT_ABORT,1,M>- ; * Send abort request
0143 284 >
0143 285
0143 286 $DEF UCBSB_LT_LATSTS .BLKB 1 ; Status for local terminal connection
0144 287 $VIELD UCB,0,<-
0144 288 <LT_HANGUP,1,M>- ; * We are hanging up the slot
0144 289 <LT_DEAD,1,M>- ; * We have linked the UCB on deadlist
0144 290 <LT_OVFLOW,1,M>- ; * We had an overflow on input (XOFF)
0144 291 <LT_BOUND,1,M>- ; * Terminal is bound, dcn't delete UCB
0144 292 <LT_INPUT,1,M>- ; * Input stream in progress
0144 293 >
0144 294
0144 295 $DEF UCBSB_LT_STATE .BLKB 1 ; Slot state
0145 296 $EQU UCBSB_LT_STATE_START 1 ; * State is starting
0145 297 $EQU UCBSB_LT_STATE_RUN 2 ; * State is running
0145 298 $EQU UCBSB_LT_STATE_STOP 3 ; * State is stopping (send stop slot)
0145 299 $EQU UCBSB_LT_STATE_KILL 4 ; * Kill UCB immediately
0145 300
0145 301 $DEF UCBSB_LT_REASON .BLKB 1 ; Reason for stopping slot
0146 302 $DEF UCBSB_LT_TCRED .BLKB 1 ; Available credits for this slot
0147 303 $DEF UCBSB_LT_XCRED .BLKB 1 ; Credits extended to concentrator
0148 304 $DEF UCBSB_LT_RCRED .BLKB 1 ; Credits to be returned to concentrator
0149 305 $DEF UCBSB_LT_SLOTSZ .BLKB 1 ; Slot size (negotiated)
014A 306 $DEF UCBSB_LT_MAXC .BLKB 1 ; Size of UCB buffering
014B 307 $DEF UCBSB_LT_CURC .BLKB 1 ; Current character count
014C 308 $DEF UCBSB_LT_CBUF .BLKB LAT$C_UCB_BUFSIZ ; Startio data buffer
019C 309 $DEF UCBSB_LT_EBUF ; Startio data buffer end
019C 310 $DEF UCBSB_LT_LENGTH ; Length of block
019C 311 $DEFEND UCB

```

```

0000 313
0000 314 :+
0000 315 : Circuit state block
0000 316 :
0000 317 : For each concentrator, one virtual circuit is maintained.
0000 318 : Protocol state is maintained in circuit state blocks.
0000 319 : The CSB addresses are stored in a vector starting
0000 320 : at CSB_TABLE.
0000 321 :-
0000 322
0000 323 $DEFINI CSB GLOBAL
0000 324 .BLKL 1
0000 325 .BLKL 1
0008 326 $DEF CSB_W_SIZE .BLKW 1 ; Size of this block in bytes
000A 327 $DEF CSB_B_TYPE .BLKB 1 ; Type of block
000B 328 $DEF CSB_B_STATE .BLKB 1 ; Virtual circuit state:
000C 329 $SECU CSB_C_STATE_HALT 0 ; * Circuit halted state
000C 330 $SECU CSB_C_STATE_START 1 ; * Circuit starting state
000C 331 $SECU CSB_C_STATE_RUN 2 ; * Circuit running state
000C 332 :
000C 333 : Counter area - must follow standard block header:
000C 334 :
000C 335 ASSUME GHBST_CSBCTR EQ
000C 336 $DEF CSB_Z_LCB .BLKB LCB_C_LENGTH ; Circuit block counter area
001C 337 :
001C 338 : Server name area - must immediately follow counter area
001C 339 :
001C 340 $DEF CSB_B_SERVER .BLKB 1 ; Length of server name
001D 341 $DEF CSB_T_SERVER .BLKB GHBSK_NAMELEN ; Server name field
002D 342 $DEF CSB_B_REFC .BLKB 1 ; Number of UCBs attached to this CSB
002E 343 :
002E 344 : Destination Ethernet address - must immediately follow server name area
002E 345 :
002E 346 $DEF CSB_Z_DST .BLKW 3 ; Concentrator physical address
0034 347 :
0034 348 $DEF CSB_Q_XBUFQ .BLKQ 1 ; Current transmit buffer queue
003C 349 $DEF CSB_Q_XWAITQ .BLKQ 1 ; Transmit buffers waiting queue
0044 350 $DEF CSB_L_XCXB .BLKL LATSC_XMT_BUFFERS ; Transmit XCXBs in progress
004C 351 $DEF CSB_L_STOPREASON .BLKL 1 ; Address of the stop reason block
0050 352 $DEF CSB_W_TIMEOUT .BLKW 1 ; Timeout cell for CSB
0052 353 $DEF CSB_W_XMTMO .BLKW 1 ; Timeout count for retransmit
0054 354 $DEF CSB_W_PROGRESS .BLKW 1 ; Progress counter
0056 355 $DEF CSB_W_PROGSEQ .BLKW 1 ; Transmit error control for progress
0058 356 $DEF CSB_B_XMTBSY .BLKB 1 ; Transmit busy and count of waiters
0059 357 $DEF CSB_B_XMTCNT .BLKB 1 ; Count of buffers to transmit
005A 358 $DEF CSB_B_XCXB_INX .BLKB 1 ; XCXB index
005B 359 $DEF CSB_B_SPARE .BLKB 1 ; SPARE
005C 360
005C 361 ; Start of circuit header as it appears in transmitted messages
005C 362 :
005C 363 $DEF CSB_T_CIRCHDR .BLKB 1 ; Lat circuit header
005C 364 $DEF CSB_B_FLAG .BLKB 1 ; Message flags byte
005D 365 $DEF CSB_B_NUM_SLOTS .BLKB 1 ; Number of slots
005E 366 $DEF CSB_W_REMID .BLKB 1 ; Remote virtual circuit id
005E 367 $DEF CSB_B_REMIDN .BLKB 1 ; Remote virtual circuit id number
005F 368 $DEF CSB_B_REMIDS .BLKB 1 ; Remote virtual circuit id sequence
0060 369 $DEF CSB_W_LOCID .BLKB 1 ; Local virtual circuit id

```

```
0060 370 $DEF CSB_B_LOCIDN .BLKB 1 ; Local virtual circuit id number
0061 371 $DEF CSB_B_LOCIDS .BLKB 1 ; Local virtual circuit id sequence
0062 372 $DEF CSB_W_XSEQ ; Transmitter error control
0062 373 $DEF CSB_B_XSEQ .BLKB 1 ; Sequence number being transmitted
0063 374 $DEF CSB_B_XACK .BLKB 1 ; Highest sequence number received
0064 375 $EQU CSB_S_CIRCHDR <.-CSB_T_CIRCHDR> ; Size of header
0064 376
0064 377 ; End of circuit header
0064 378
0064 379 $DEF CSB_W_RSEQ ; Sequence number to receive
0064 380 $DEF CSB_B_RSEQ .BLKB 1 ; Sequence number to receive
0065 381 $DEF CSB_B_RACK .BLKB 1 ; Highest sequence number acknowledged
0066 382 $DEF CSB_W_TIMRESET .BLKW 1 ; Timeout reset value
0068 383 $DEF CSB_B_INX .BLKB 1 ; CSB table index
0069 384 $DEF CSB_B_MAX_SLOTS .BLKB 1 ; Maximum number of slots (negotiated)
006A 385 $DEF CSB_W_MAX_MSGSIZ .BLKW 1 ; Maximum message size (negotiated)
006C 386 $DEF CSB_C_FIXLENGTH ; Length of fixed block
006C 387 :
006C 388 : UCB list area (size = 4*min(lat$c_max_slots,server_slot_count) )
006C 389 :
006C 390 ALIGN_LONG ; align to longword boundary
006C 391 $DEF CSB_L_UCBLST
006C 392
006C 393 $DEFEND CSB
```



```
0000 395
0000 396 :+
0000 397 :
0000 398 : Terminal overflow input buffer definitions
0000 399 : The terminal overflow buffer is only used when a call is made to
0000 400 : the XOFF routine to stop the input flow. The overflow buffer is
0000 401 : allocated and the rest of the input data is copied to the overflow
0000 402 : buffer. When we are resumed at XON, the data from the input overflow
0000 403 : buffer is dumped into the class driver.
0000 404 :
0000 405 :-
0000 406
0000 407 $DEFINI INB GLOBAL
0000 408
0000 409 $DEF INB_W_BOFF .BLKW 1 ; Offset to start of data
0002 410 $DEF INB_W_BCNT .BLKW 1 ; Count of data remaining
0004 411 $DEF INB_L_SPARE .BLKL 1 ; Spare longword
0008 412 $DEF INB_W_SIZE .BLKW 1 ; Size of structure
000A 413 $DEF INB_B_TYPE .BLKB 1 ; Type of structure
000B 414 $DEF INB_B_SPARE .BLKB 1 ; Spare byte
000C 415 $DEF INB_C_LENGTH ; Size of structure header
000C 416
000C 417 $DEFEND INB
0000 418
```



```

0000 420
0000 421 :+
0000 422 :
0000 423 :   Receive buffer offsets and bit definitions
0000 424 :
0000 425 :   The format of receive and transmit buffers is identical for fields
0000 426 :   Starting with and following the destination address field. Ease of
0000 427 :   implemenation dictates that the offsets be defined relative to
0000 428 :   the start of the buffer.
0000 429 :
0000 430 :   These are the offsets into the architecturally defined portion of
0000 431 :   the receive buffer itself.
0000 432 :-
0000 433           $DEFINI RCV      GLOBAL
0000 434
00000000 0000 435   .=0
0000 436 $DEF   RCV_T_DATA      ; Start of data is right here
0000 437 $DEF   RCV_B_FLAG     .BLKB 1 ; Flags byte and message type
0001 438           ;
0001 439           ; Flag bits:
0001 440           ;
0001 441   _VIELD FLAG,0,<-      ; * Flags byte bit definitions recv & xmit
0001 442   <RRF,1,M>,-          ; * Response requested flag
0001 443   <MASTER,1,M>,-      ; * Remote station always sets this flag
0001 444   <MTYPE,6,M>,-      ; * Message type bits
0001 445   >
0001 446           ;
0001 447           ; Message types:
0001 448           ;
0001 449   $EQU   MTYP_C_RUN      0      ; * Run message
0001 450   $EQU   MTYP_C_START  1      ; * Start message
0001 451   $EQU   MTYP_C_HALT   2      ; * Halt message
0001 452   $EQU   MTYP_C_CONF   10     ; * Configuration message
0001 453
0001 454 $DEF   RCV_B_NUM_SLOTS .BLKB 1 ; Number of slots in buffer
0002 455 $DEF   RCV_W_DSTID     ; Destination virtual circuit id
0002 456 $DEF   RCV_B_DSTIDN   .BLKB 1 ; Destination virtual circuit id number
0003 457 $DEF   RCV_B_DSTIDS   .BLKB 1 ; Destination virtual circuit id seq
0004 458 $DEF   RCV_W_SRCID    ; Source virtual circuit id
0004 459 $DEF   RCV_B_SRCIDN   .BLKB 1 ; Source virtual circuit id number
0005 460 $DEF   RCV_B_SRCIDS   .BLKB 1 ; Source virtual circuit id
0006 461           ;
0006 462           ; sequence number
0006 463 $DEF   RCV_B_SEQ        .BLKB 1 ; Buffer sequence number
0007 464 $DEF   RCV_B_ACK       .BLKB 1 ; Sequence number acknowledgement
0008 465 $DEF   RCV_T_MDATA     ; Start of type dependent data
0008 466
0008 467           $DEFEND RCV

```

```

0000 469
0000 470 :
0000 471 : Define START message offsets (used for both xmits and recvs)
0000 472 :
0000 473 $DEFINI STRT GLOBAL
0000 474 $DEF STRT_W_MSGSIZ .BLKW 1 ; Data link frame size
0002 475 $DEF STRT_B_PVER .BLKB 1 ; Circuit protocol version
0003 476 $DEF STRT_B_PECO .BLKB 1 ; Circuit Eng. change order level
0004 477 $DEF STRT_B_MAXSLOTS .BLKB 1 ; Maximum simultaneous slots
0005 478 $DEF STRT_B_DL_BFRS .BLKB 1 ; Number of extra rcv bfrs in server
0006 479 $DEF STRT_B_CIR_TIMR .BLKB 1 ; Local circuit timer in milliseconds
0007 480 $DEF STRT_B_KPA_TIMR .BLKB 1 ; Local keep alive timer in seconds
0008 481 $DEF STRT_W_FAC_NUM .BLKW 1 ; Facility number
000A 482 $DEF STRT_W_PROD_TYP .BLKW 1 ; Product type code
000C 483 $DEF STRT_C_LENGTH ; Length of fixed portion of message
000C 484 :
000C 485 : Start of variable length parameters
000C 486 :
000C 487 $DEF STRT_B_VAR ; Start of variable portion of start message
000C 488 $DEF STRT_B_NODE_LEN .BLKB 1 ; Destination node name length
000D 489 $DEF STRT_T_NODE .BLKB GHBSK_NAMELEN ; Destination node name text
001D 490 $DEF STRT_B_SYS_LEN .BLKB 1 ; Source node name length
001E 491 $DEF STRT_T_SYS .BLKB GHBSK_NAMELEN ; Source node name text
002E 492 $DEF STRT_B_ID_LEN .BLKB 1 ; Source node descriptor length
002F 493 $DEF STRT_T_ID .BLKB GHBSK_IDLEN ; Source node descriptor text
006F 494 $DEF STRT_B_LOC_LEN .BLKB 1 ; Source location length
0070 495 $DEF STRT_T_LOC .BLKB LATSC_LOC_LEN ; Source location text
0080 496 $DEF STRT_B_PARAM .BLKB 1 ; Parameter code
00B1 497 $SEQU STRT_C_VAR_LEN <.-STRT_B_VAR> ; Length of variable portion of start
00B1 498 ; message
00B1 499 $DEFEND STRT
0000 500
0000 501 :
0000 502 : Define STOP message offsets
0000 503 :
0000 504 $DEFINI STOP GLOBAL
0000 505 $DEF STOP_B_RCODE .BLKB ; Circuit disconnect reason
0001 506 $DEF STOP_B_RLEN .BLKB ; Reason length in bytes
0002 507 $DEF STOP_T_REAS ; Reason ASCII text
0002 508 $DEFEND STOP
0000 509

```

```

0000 511
0000 512 :+
0000 513 : Transmit buffer offsets and bit definitions
0000 514 :
0000 515 : The buffer header is defined to be the same as that used by
0000 516 : the Ethernet datalink drivers.
0000 517 :-
0000 518
0000 519 $DEFINI CXB GLOBAL ; Transmit complex buffer
0000 520
0000 521 : Define the transmit/receive offsets in the CXB
0000 522 :
0000001C 0000 523 . = CXBSL_END_ACTION
001C 524 $DEF CXBSL_T_ENDADR .BLKL 1 ; End address of data to transmit
0020 525
0000000C 0020 526 . = CXBSW_LENGTH
000C 527 ASSUME CXBSW_OFFSET EQ CXBSW_LENGTH+2
000C 528 $DEF CXBSL_T_CSB ; CSB address
000C 529
00000014 000C 530 . = CXBSL_IRP
0014 531 $DEF CXBSL_T_SAVE .BLKL 1 ; BOFF & BCNT save area
0018 532
0018 533 $DEFEND CXB
0000 534
0000 535
0000 536 : Start of architecturally defined offsets
0000 537 :
0000 538 $DEFINI XMT GLOBAL
0000 539
0000003A 0000 540 . = CXBSC HEADER - <6+6+2>
003A 541 $DEF XMT_G_DST .BLKW 3 ; Destination address
0040 542
00000048 0040 543 . = CXBSC HEADER
0048 544 $DEF XMT_T_DATA ; Start of transmitted data
0048 545 $DEF XMT_B_FLAG .BLKB 1 ; Flags byte and message type
0049 546
0049 547 :
0049 548 : Run message format:
0049 549 :
0049 550 $DEF XMT_B_NUM_SLOTS .BLKB 1 ; Number of slots in buffer
004A 551 $DEF XMT_W_DSTID ; Destination virtual circuit id
004A 552 $DEF XMT_B_DSTIDN .BLKB 1 ; Destination virtual circuit id number
004B 553 $DEF XMT_B_DSTIDS .BLKB 1 ; Destination virtual circuit id seq
004C 554 $DEF XMT_W_SRCID ; Source virtual circuit id
004C 555 $DEF XMT_B_SRCIDN .BLKB 1 ; Source virtual circuit id number
004D 556 $DEF XMT_B_SRCIDS .BLKB 1 ; Source virtual circuit id
004E 557 ; sequence number
004E 558
004E 559 $DEF XMT_B_SEQ .BLKB 1 ; Buffer sequence number
004F 560 $DEF XMT_B_ACK .BLKB 1 ; Sequence number acknowledgement
0050 561 $DEF XMT_T_MDATA ; Start of RUN message data
0050 562
0050 563 :
0050 564 : Multicast (CONFIGURATION) message format:
0050 565 :
00000049 0050 566 . = XMT_B_FLAG+1
0049 567 $DEF XMT_B_MC_CIR_TIMER .BLKB 1 ; Circuit timer

```

```

004A 568 $DEF XMT_B_MC_HI_VER .BLKB 1 ; Highest supported protocol version
004B 569 $DEF XMT_B_MC_LO_VER .BLKB 1 ; Lowest supported protocol version
004C 570 $DEF XMT_B_MC_CUR_VER .BLKB 1 ; version of this message
004D 571 $DEF XMT_B_MC_CUR_ECO .BLKB 1 ; eco of this message
004E 572 $DEF XMT_B_MC_INCARN .BLKB 1 ; Message incarnation
004F 573 $DEF XMT_B_MC_CHG_FLAG .BLKB 1 ; Changed fields in this msg
0050 574
0050 575 YIELD XMT_CHFLG,0,<-
0050 576 ZGROUP,1,M>,- ; * NODE group codes changed
0050 577 <NDESC,1,M>,- ; * NODE descriptor changed
0050 578 <SVCNAM,1,M>,- ; * Service name/number changed
0050 579 <RATE,1,M>,- ; * Service ratings changed
0050 580 <SDESC,1,M>,- ; * Service classes changed
0050 581 <CLASS,1,M>,- ; * Service classes changed
0050 582 <1>- ; * RESERVED
0050 583 <OTHER,1,M>,- ; * One of the OTHER fields changed
0050 584 >
0050 585 $EQU XMT_CHFLG_M_ALL <^XFF> ; * All codes changed
0050 586
0050 587 $DEF XMT_W_MC_MSG_SIZE .BLKB 1 ; Maximum frame size
0052 588 $EQU XMT_C_MC_LENGTH <.-XMT_T_DATA> ; Size of fixed part of multicast me
0052 589
0052 590 $DEF XMT_B_MC_SET ; Start of LATCP set portion of multicast msg
0052 591 $DEF XMT_B_MC_MULTIMR .BLKB 1 ; Multicast message timer
0053 592 $DEF XMT_B_MC_STATUS .BLKB 1 ; Node status
0054 593 $DEF XMT_B_MC_GRP_LEN .BLKB 1 ; Number of group codes
0055 594 $DEF XMT_T_MC_GROUP .BLKB LATSC_GRP_LEN ; Group code field
0075 595 $DEF XMT_B_MC_NODE_LEN .BLKB 1 ; Node name length
0076 596 $DEF XMT_T_MC_NODE .BLKB GHBSK_NAMELEN ; NODE name
0086 597 $DEF XMT_B_MC_NUM_SVCS .BLKB 1 ; Number of services
0087 598
0087 599 $DEF XMT_B_MC_RATE .BLKB 1 ; Service rating
0088 600 $DEF XMT_B_MC_NAME_LEN .BLKB 1 ; Service name length
0089 601 $DEF XMT_T_MC_NAME .BLKB GHBSK_NAMELEN ; Service name
0099 602 $DEF XMT_B_MC_ID_LEN .BLKB 1 ; Length of Identification field
009A 603 $DEF XMT_T_MC_ID .BLKB GHBSK_IDLEN ; Identification field
00DA 604 $EQU XMT_C_MC_SVC_SIZE <.-XMT_B_MC_RATE> ; Size of each service field
00DA 605
00DA 606 $DEF XMT_T_MC_MORE_SVC .BLKB <<LATSC_MAX_SERVICES-1>*XMT_C_MC_SVC_SIZE>
0279 607
0279 608 $DEF XMT_B_MC SVC_LEN .BLKB 1 ; Number of service classes
027A 609 $DEF XMT_B_MC SVC .BLKB LATSC_SVC_LEN ; Service classes
0282 610 $EQU XMT_C_MC_TOTAL <.-XMT_T_DATA> ; Length of total message
0282 611 ASSUME XMT_C_MC_TOTAL LE LATSC_MAX_MSGSIZ
0282 612
0282 613 $EQU XMT_C_HDRLEN <XMT_ _MDATA - XMT_T_DATA> ; Length of header
0282 614 ;
0282 615 $EQU XMT_C_MAXDATA <LATSC_MAX_MSGSIZ - XMT_C_HDRLEN> ; Data in slots
0282 616 ;
0282 617 $EQU XMT_C_MAXLEN <LATSC_MAX_MSGSIZ + XMT_T_DATA> ; Total size of a buffer
0282 618 ;
0282 619 $DEFEND XMT

```

```

0000 621 :+
0000 622 :
0000 623 : Buffer slot offsets
0000 624 :
0000 625 : Each slot is formatted identically in both
0000 626 :- transmit and receive buffers as defined below.
0000 627 :
0000 628 $DEFINI SLT GLOBAL
0000 629
0000 630 $DEF SLT_T_START ; Start of slot
0000 631 $DEF SLT_B_DSTID .BLKB 1 ; Destination slot ID
0001 632 $DEF SLT_B_SRCID .BLKB 1 ; Source slot ID
0002 633 $DEF SLT_B_COUNT .BLKB 1 ; Character count in slot
0003 634 $DEF SLT_B_REAS ; Stop/reject slot reason (low nibble)
0003 635 $DEF SLT_B_CRED ; Credits extended (low order nibble)
0003 636 $DEF SLT_B_TYPE .BLKB 1 ; Slot type (high order nibble)
0004 637 ; STOP/REJECT reason codes:
0004 638 $EQU SLT_C_USD 1 ; * User requested disconnect
0004 639 $EQU SLT_C_SYS 2 ; * System shutdown
0004 640 $EQU SLT_C_INVSLOT 3 ; * Invalid slot received
0004 641 $EQU SLT_C_INVSVC 4 ; * Invalid service class received
0004 642 $EQU SLT_C_RESOURCE 5 ; * Insufficient resources
0004 643 ; Slot type codes:
0004 644 $EQU SLT_C_DA_SLOT <^X00> ; * DATA A slot
0004 645 $EQU SLT_C_STR_SLOT <^X09> ; * START slot
0004 646 $EQU SLT_C_DB_SLOT <^X0A> ; * DATA B slot
0004 647 $EQU SLT_C_ATT_SLOT <^X0B> ; * ATTENTION slot
0004 648 $EQU SLT_C_REJ_SLOT <^X0C> ; * REJECT slot
0004 649 $EQU SLT_C_STP_SLOT <^X0D> ; * STOP slot
0004 650 $DEF SLT_T_DATA ; Start of slot data
0004 651
0004 652 :
0004 653 : Start slot data definitions
0004 654 :
00000004 0004 655 : = SLT_T_DATA
0004 656 $DEF SLT_B_SVCC .BLKB 1 ; Service class
0005 657 $EQU SLT_C_SVC_INTT 1 ; * Interactive Terminals
0005 658 $DEF SLT_B_AT_SLTSIZ .BLKB 1 ; Maximum length of attention slot
0006 659 $DEF SLT_B_DT_SLTSIZ .BLKB 1 ; Maximum length of Data slot
0007 660 $DEF SLT_B_DST_NAMLEN .BLKB 1 ; Length of destination slot name
0008 661 $DEF SLT_T_DST_NAME .BLKB GHBSK_NAMELEN
0018 662 $DEF SLT_B_SRC_NAMLEN .BLKB 1 ; Length of source slot name
0019 663 $DEF SLT_B_SRC_NAME .BLKB GHBSK_NAMELEN
0029 664 $EQU SLT_S_START_SLOT <.-SLT_T_DATA> ; Length of Start slot
0029 665 $DEF SLT_B_PARAMS ; Start of parameter area
0029 666 $EQU SLT_C_START_FLAG 1 ; * Start slot parameter flag is code 1
0029 667 -VIELD SLT_FLAG,0,<-
0029 668 <REMOTE,1,M>,- ; * Remote line
0029 669 <LOGIN,1,M>,- ; * Disable auto-login
0029 670 <BIND,1,M>,- ; * Terminal UCB is bound to server
0029 671 >
0029 672
0029 673 :
0029 674 : DATA_B slot data definitions
0029 675 :
00000004 0029 676 : = SLT_T_DATA
0004 677 $DEF SLT_B_DB_CONTROL .BLKB 1 ; DATA_B slot control flag

```

```

0005 678      _VIELD  SLT_DB,0,<-
0005 679      <IFENA,1,M>,-
0005 680      <IFDIS,1,M>,-
0005 681      <OFENA,1,M>,-
0005 682      <OFDIS,1,M>,-
0005 683      <BREAK,1,M>,-
0005 684      >
0005 685 $DEF  SLT_B_DB_OFOF  .BLKB  1
0006 686 $DEF  SLT_B_DB_OFON  .BLKB  1
0007 687 $DEF  SLT_B_DB_IFOF  .BLKB  1
0008 688 $DEF  SLT_B_DB_IFON  .BLKB  1
0009 689 $EQU  SLT_S_DB_LEN   <.-SLT_T_DATA>
0009 690 $EQU  SLT_S_DB_SLOT <.-SLT_T_START>
0009 691
0009 692
0009 693 : ATTention slot data definitions
0009 694 :
00000004 0009 695 : = SLT_T_DATA
0004 696 $DEF  SET_B_ATT_CONTROL .BLKB  1
0005 697      _VIELD  SET_ATT,0,<-
0005 698      <.55>,-
0005 699      <ABORT,1,M>,-
0005 700      >
0005 701 $EQU  SLT_S_ATT_LEN   <.-SLT_T_DATA>
0005 702 $EQU  SLT_S_ATT_SLOT <.-SLT_T_START>
0005 703
0005 704 :
0005 705 : STOP slot data definitions
0005 706 :
00000004 0005 707 : = SLT_T_DATA
0004 708 $DEF  SET_B_STP_RLEN  .BLKB  1
0005 709 $EQU  SLT_S_STP_LEN   <.-SLT_T_DATA>
0005 710 $EQU  SLT_S_STP_SLOT <.-SLT_T_START>
0005 711 $DEF  SLT_T_STP_REAS
0005 712
0005 713 :
0005 714 : REJECT slot data definitions
0005 715 :
00000004 0005 716 : = SLT_T_DATA
0004 717 $DEF  SET_B_REJ_RLEN  .BLKB  1
0004 718 $EQU  SLT_S_REJ_LEN   <.-SLT_T_DATA>
0004 719 $EQU  SLT_S_REJ_SLOT <.-SLT_T_START>
0004 720 $DEF  SLT_T_REJ_REAS
0004 721
0004 722 $DEFEND SLT

```

:JAY0002
-1

```

0000 724 :
0000 725 : Local macros
0000 726 :
0000 727 :
0000 728 .MACRO INC_CTR, BASE, TYPE=L, ?L
0000 729 INC^TYPE BASE ; Increment counter
0000 730 BNEQ L ; Br if no overflow
0000 731 DEC^TYPE BASE ; Else, latch at maximum
0000 732 L:
0000 733 .END INC_CTR
0000 734
0000 735 .MACRO ADD_CTR, BASE1, BASE2, TYPE=L, ?L
0000 736 ADD^TYPE BASE1, BASE2 ; Add counters
0000 737 BCC L ; Br if no overflow
0000 738 MNEG^TYPE #1, BASE2 ; Else, latch at maximum
0000 739 L:
0000 740 .END ADD_CTR
0000 741
0000 742 .MACRO $DISPATCH, INDX, VECTOR, TYPE=W, NMODE=S^#, ?MN, ?MX, ?S, ?SS, ?ZZ
0000 743 SS:
0000 744
0000 745 .MACRO $DSP1, $DSP1_1
0000 746 .IRP $DSP1_2, $DSP1_1
0000 747 $DSP2_ $DSP1_2
0000 748 .ENDR
0000 749 .ENDM $DSP1
0000 750
0000 751 .MACRO $DSP2, $DSP2_1, $DSP2_2
0000 752 .=<$DSP2_1-MN>*2 + 5
0000 753 .WORD $DSP2_2-S
0000 754 .ENDM $DSP2
0000 755
0000 756
0000 757 .MACRO $BND1, $BND1_1, $BND1_2, $BND1_3
0000 758 $BND2 $BND1_1, $BND1_2
0000 759 .ENDM $BND1
0000 760
0000 761 .MACRO $BND2, $BND2_1, $BND2_2
0000 762 .IIF $BND2_1, $BND2_2-., .=$BND2_2
0000 763 .ENDM $BND2
0000 764
0000 765 .MACRO $BND $BND_1, $BND_2
0000 766 .IRP $BND_3, <$BND_2>
0000 767 $BNDT $BND_1, $BND_3
0000 768 .ENDR
0000 769 .ENDM $BND
0000 770
0000 771 .=0
0000 772 ZZ:
0000 773 $BND GT, <VECTOR>
0000 774 MX:
0000 775 $BND LT, <VECTOR>
0000 776 MN:
0000 777 .=SS
0000 778
0000 779 CASE^TYPE INDX, #<MN-ZZ>, NMODE^<MX-MN>
0000 780 S:

```

```
0000 781      .REPT  MX-MN+1
0000 782      .WORD  <MX-MN>*2 + 2
0000 783      .ENDR
0000 784
0000 785      .=S
0000 786
0000 787      $DSP1  <<VECTOR>>
0000 788
0000 789      .=<MX-MN>*2 + S + 2
0000 790
0000 791 .ENDM $DISPATCH
0000 792
0000 793
0000 794      .MACRO  SETBIT BIT, BASE, ?L      ; Set specified bit
0000 795      BBSS  BIT, BASE, L
0000 796 L:
0000 797      .ENDM
0000 798
0000 799      .MACRO  CLRBIT BIT, BASE, ?L      ; Clear specified bit
0000 800      BBCC  BIT, BASE, L
0000 801 L:
0000 802      .ENDM
0000 803
0000 804      .MACRO  BREAK
0000 805      JSB   G^INISBRK
0000 806      .ENDM
0000 807
0000 808      .MACRO  DEBUG
0000 809      TSTL  @#-1
0000 810      .ENDM  DEBUG
0000 811
```



```

0000 813      .SBTTL  Driver Prologue Table
0000 814
00000000 815      .PSECT  $$$105_PROLOGUE
0000 816
0000 817      ;
0000 818      ; Driver prologue table:
0000 819      ;
0000 820
0000 821  LT$DPT::
0000 822      DPTAB      -      ; Driver start
0000 823      END=LT$END,-      ; Driver prologue table
0000 824      UCBSIZE=UCB$C_LT_LENGTH,- ; Address of end of module
0000 825      ADAPTER=NULL,-      ; Size of each ucb of port driver
0000 826      NAME=LTDRIVER,-      ; Adapter type
0000 827      VECTOR=LT$VECTOR      ; Name of driver
0000 828      ; Port driver vector table
0038 828      DPT_STORE INIT
0038 829      DPT_STORE UCB,UCB$B_FIPL,B,LAT$C_IPL ; Fork ipl
003C 830      DPT_STORE UCB,UCB$L_DEVCHAR,L,<- ; Characteristics
003C 831      DEVSM_REC!-
003C 832      DEVSM_AVL!-
003C 833      DEVSM_IDV!-
003C 834      DEVSM_ODV!-
003C 835      DEVSM_TRM!-
003C 836      DEVSM_CCL>
0043 837      DPT_STORE UCB,UCB$L_DEVCHAR2,L,<-
0043 838      DEVSM_NNM> ; Prefix name with <node$>
004A 839      DPT_STORE UCB,UCB$B_DEVCLASS,B,DC$ TERM;
004E 840      DPT_STORE UCB,UCB$B_TT_DETYPE,B,TT$ UNKNOWN ; Type
0052 841      DPT_STORE UCB,UCB$W_TT_DESIZE,@W,TTY$GW_DEFBUF ; Buffer size
0059 842      DPT_STORE UCB,UCB$L_TT_DECHAR,@L,TTY$GL_DEFCHAR ; Default characters
0060 843      DPT_STORE UCB,UCB$L_TT_DECHA1,@L,TTY$GL_DEFCHAR2; Default characters
0067 844      DPT_STORE UCB,UCB$W_TT_DESPEE,@B,TTY$GB_DEFSPEED; Default speed
006E 845      DPT_STORE UCB,UCB$W_TT_DESPEE+1,@B,TTY$GB_RSPEED; Default rspeed
0075 846      DPT_STORE UCB,UCB$B_TT_DEPARI,@B,TTY$GB_PARITY ; Default parity
007C 847      ;** DPT_STORE UCB,UCB$B_TT_PARITY,@B,TTY$GB_PARITY ; Default parity
007C 848      DPT_STORE UCB,UCB$B_DEVTYPE,B,TT$ UNKNOWN ; Type
0080 849      DPT_STORE UCB,UCB$W_DEVBUFSIZ,@W,TTY$GW_DEFBUF ; Buffer size
0087 850      DPT_STORE UCB,UCB$L_DEVDEPEND,@L,TTY$GL_DEFCHAR ; Default characters
008E 851      DPT_STORE UCB,UCB$L_TT_DEVDP1,@L,TTY$GL_DEFCHAR2; Default characters
0095 852      DPT_STORE UCB,UCB$W_TT_SPEED,@B,TTY$GB_DEFSPEED ; Default speed
009C 853      DPT_STORE UCB,UCB$W_TT_SPEED+1,@B,TTY$GB_RSPEED ; Default rspeed
00A3 854      DPT_STORE UCB,UCB$B_DIPL,B,LAT$C_IPL ; Device ipl
00A7 855      DPT_STORE UCB,UCB$L_TT_WFLINK,L,0 ; Zero write queue.
00AE 856      DPT_STORE UCB,UCB$L_TT_WBLINK,L,0 ; Zero write queue.
00B5 857      DPT_STORE UCB,UCB$L_TT_RTIMOU,L,0 ; Zero read timed out
00BC 858      ;** DPT_STORE UCB,UCB$W_TT_PRTCTL,W,TTY$M_PC_NOTIME ; Disable all timers
00BC 859      DPT_STORE ORB,ORB$B_FLAGS,B,- ; Default flags
00BC 860      ; ZORB$M_PROT 16> ; SOGW word protection
00C0 861      DPT_STORE ORB,ORB$W_PROT,@W,TTY$GW_PROT ; Default protection
00C7 862      DPT_STORE ORB,ORB$L_OWNER,@L,TTY$GL_OWNUIC ; Default owner uic
00CE 863      DPT_STORE DDB,DOB$L_DDT,D,LT$DDT
00D3 864
00D3 865      DPT_STORE REINIT
00D3 866      DPT_STORE CRB,CRB$L_INTD+VEC$L_INITIAL,D,LT$CTRL_INIT ; Controller init
00D8 867      DPT_STORE CRB,CRB$L_INTD+VEC$L_UNITINIT,D,LT$UNIT_INIT ; Unit init
00DD 868      DPT_STORE END
0000 869

```

LTDRIVER
V04-002

- Local Area Terminal Port Driver L 15
Driver Prologue Table

8-JAN-1985 17:22:25 VAX/VMS Macro V04-00
5-SEP-1984 11:40:22 [LAT.BUGSRC]LTDRIVER.MAR;1

Page 19
(18)

0000 870
0000 871
0000 872
0000 873
0000 874
0000 875
0000 876
0038 877

DDTAB DEVNAM = LT,- ; Dummy port driver dispatch table
START = 0,-
FUNCTB = 0

```

0038 879 .SBTTL Local Storage
0038 880
0038 881
00000038 882 .PSECT $$$1,5_DRIVER, LONG
0038 883
0038 884 :
0038 885 : Local storage
0038 886 :
0038 887 :
0038 888 :+
0038 889 : Circuit State Block table
0038 890 :
0038 891 : This table is indexed by RCV_B_DSTIDN in received buffers. A
0038 892 : concentrator always uses the same value when referring to a
0038 893 : host for a second time. The RCV_B_DSTIDS byte is a check byte
0038 894 : which helps prevent random or outdated references to CSBs.
0038 895 :
0038 896 :-
0038 897
0038 898 .ALIGN LONG
0038 899
0000008R 0038 900 GHB_L_CSBTABLE:
0038 901 .BLKL LATSC_MAX_CSBS ; CSB address table
0088 902 GHB_B_LOC_CHK: ; Table of local
000000D9 0088 903 .BLKB LATSC_MAX_CSBS+1 ; circuit checks
00D9 904
00D9 905
00D9 906 :
00D9 907 : Timer queue element
00D9 908 :
00D9 909 : Used to timeout CSB's that have had no activity.
00D9 910 :
00D9 911
00D9 912 .ALIGN LONG
0000010C 00DC 913 GHB_G_TQE:
00DC 914 .BLKB TQESC_LENGTH
010C 915
010C .1 :
: JAY0002 010C .2 : The following is used during controller init to determine what the
: JAY0002 010C .3 : maximum number of slots supported by this VAX processor.
: JAY0002 010C .4 :
: JAY0002 010C .5 :
: JAY0002 010C .6 GHB_B_MAX_SLOTS:
: JAY0002 00 010C .7 .BYTE 0 ; Cell to contain max slot value
: JAY00C2 010D .8
: JAY0002 010D .9 GHB_T_MAX_SLOT_TABLE:
: JAY0002 40 010D .10 .BYTE LATSC_MAX_SLOTS ; 0 (non-existent)
: JAY0002 40 010E .11 .BYTE 64 ; PR$_SID_TYP780 1
: JAY0002 40 010F .12 .BYTE 64 ; PR$_SID_TYP750 2
: JAY0002 20 0110 .13 .BYTE 32 ; PR$_SID_TYP730 3
: JAY0002 FF 0111 .14 .BYTE 255 ; PR$_SID_TYP790 4
: JAY0002 40 0112 .15 .BYTE 64 ; PR$_SID_TYP855 5
: JAY0002 80 0113 .16 .BYTE 128 ; PR$_SID_TYP8NN 6
: JAY0002 10 0114 .17 .BYTE 16 ; PR$_SID_TYPUV1 7
: JAY0002 30 0115 .18 .BYTE 48 ; PR$_SID_TYPUV2 8
: JAY0002 40 0116 .19 .BYTE LATSC_MAX_SLOTS ; 9
: JAY0002 40 0117 .20 .BYTE LATSC_MAX_SLOTS ; (patch area) 10

```

LTDRIVER
V04-002

- Local Area Terminal Port Driver N 15
Local Storage

8-JAN-1985 17:22:25 VAX/VMS Macro V04-00
5-SEP-1984 11:40:22 [LAT.BUGSRC]LTDRIVER.MAR;1

Page 21
(19)

:JAY0002	40	0118	.21	.BYTE	LATSC_MAX_SLOTS	:	11
:JAY0002	40	0119	.22	.BYTE	LATSC_MAX_SLOTS	:	12
:JAY0002	40	011A	.23	.BYTE	LATSC_MAX_SLOTS	:	13
:JAY0002	40	011B	.24	.BYTE	LATSC_MAX_SLOTS	:	14
:JAY0002	40	011C	.25	.BYTE	LATSC_MAX_SLOTS	:	15
:JAY0002	40	011D	.26	.BYTE	LATSC_MAX_SLOTS	:	16
:JAY0002		011E	.27				
:JAY0002	00000010	011E	.28				

MAX_PR_SID = 16

```

011E 917
011E 918 :
011E 919 : Host counter block
011E 920 :
011E 921 :
011E 922 .ALIGN LONG
00000180 0120 923 GHB_AREA: .BLKB GHB$$_COMM_AREA ; Communications area
0120 924 .BLKB GHB$$_COMM_AREA
0180 925
0180 926 .ALIGN LONG
0180 927
00000184 0180 928 GHB_L_UCBO: .BLKL 1 ; LT UCB zero address
00000188 0184 929 GHB_L_DEADLINK: .BLKL 1 ; List of dead UCBs
0000018C 0188 930 GHB_L_FFI: .BLKL 1 ; FFI block address
00000190 018C 931 GHB_L_NULLCPU: .BLKL 1 ; Pointer to null CPU time
00000194 0190 932 GHB_L_LASTCPU: .BLKL 1 ; Value of last null CPU sample
00000198 0194 933 GHB_L_NULLSEC: .BLKL 1 ; Value of latt null CPU sample on secondary
00000199 0198 934 GHB_B_STATUS: .BLKB 1 ; Host status byte
0199 935 -VIELD GHB_STS,0,<-
0199 936 <MULTI,1,M>,- ; Multicast buffer in use
0199 937 <ACTIVE,1,M>,- ; LTDRIVER is active
0199 938 <SHUT,1,M>,- ; Shutdown is in progress
0199 939 <TQE,1,M>,- ; Waiting for TQE to fire
0199 940 >
0000019A 0199 941 GHB_B_INCARN: .BLKB 1 ; Incarnation number for multicast message
0000019B 019A 942 GHB_B_OLD_CSBCNT: .BLKB 1 ; Count of old CSBs on queue
0000019C 019B 943 GHB_B_RATE: .BLKB 1 ; Initial rating computed from memory size
019C 944 :
019C 945 ; Multicast data set by LATCP
019C 946 :
019C 947 .ALIGN LONG
0000019E 019C 948 GHB_W_MC_SIZE: .BLKW 1 ; Size of name and ID strings
000003E8 019E 949 GHB_T_MC_DATA: .BLKB XMT_C_MC_TOTAL+16 ; Name and ID storage + fudge factor
03E8 950 :
03E8 951 : Multicast (Configuration) Message Destination Ethernet address
03E8 952 :
03E8 953 :
03E8 954 .ALIGN LONG
0009 03E8 955 GHB_Q_MULTIAADD: .WORD LAT$C_MULTIAADD1 ; LAT Multicast destination address
002B 03EA 956 .WORD LAT$C_MULTIAADD2 ;
0F00 03EC 957 .WORD LAT$C_MULTIAADD3 ;
03EE 958 ; Picks up next word also (garbage)
03EE 959 :
03EE 960 ; Data to control multicast buffer
03EE 961 :
000003F0 03EE 962 GHB_W_MULTIMR: .BLKW 1 ; Timer for multicast transmits
000003F4 03F0 963 GHB_L_MULTIBFR: .BLKL 1 ; Address of the multicast buffer
03F4 964 :
03F4 965 :
03F4 966 ; Host Multicast (Configuration) Message fixed portion template
03F4 967 ; (Fields are order-dependent and adjacent)
03F4 968 :
03F4 969 ASSUME XMT_B_FLAG EQ XMT_T_DATA
03F4 970 ASSUME XMT_B_MC_CUR_TIMER EQ XMT_B_FLAG+1
03F4 971 ASSUME XMT_B_MC_HI_VER EQ XMT_B_MC_CUR_TIMER+1
03F4 972 ASSUME XMT_B_MC_LO_VER EQ XMT_B_MC_HI_VER+1
03F4 973 ASSUME XMT_B_MC_CUR_VER EQ XMT_B_MC_LO_VER+1

```

```

03F4 974 ASSUME XMT_B_MC_CUR_ECO EQ XMT_B_MC_CUR_VER+1
03F4 975 ASSUME XMT_B_MC_INCARN EQ XMT_B_MC_CUR_ECO+1
03F4 976 ASSUME XMT_B_MC_CHG_FLAG EQ XMT_B_MC_INCARN+1
03F4 977 ASSUME XMT_W_MC_MSG_SIZ EQ XMT_B_MC_CHG_FLAG+1
03F4 978 ASSUME XMT_B_MC_SET EQ XMT_W_MC_MSG_SIZ+2
03F4 979 GHB_T_MC_MSG: ; Start of multicast msg fixed data
28 03F4 980 .BYTE MTYP_C_CONF@FLAG y MTYPE ; Message type
08 03F5 981 .BYTE <LATSC_CIR_TIMER+95/10 ; Host preferred Circuit timer
05 03F6 982 .BYTE LATSC_HI_VER ; Highest LAT protocol version supported
05 03F7 983 .BYTE LATSC_LO_VER ; Lowest LAT protocol version supported
05 03F8 984 .BYTE LATSC_CUR_VER ; Current LAT protocol version
00 03F9 985 .BYTE LATSC_CUR_ECO ; Current LAT protocol ECO version
00 03FA 986 .BYTE 0 ; Message incarnation (filled in later)
00 03FB 987 .BYTE 0 ; Change flags (filled in later)
05DC 03FC 988 .WORD LATSC_MAX_MSGSIZ ; Maximum message size
03FE 989
03FE 990
03FE 991 : Standard Start message (fixed portion) template
03FE 992 :
03FE 993 ASSUME STRT_W_MSGSIZ EQ 0
03FE 994 ASSUME STRT_B_PVER EQ STRT_W_MSGSIZ+2
03FE 995 ASSUME STRT_B_PECO EQ STRT_B_PVER+1
03FE 996 ASSUME STRT_B_MAXSLOTS EQ STRT_B_PECO+1
03FE 997 ASSUME STRT_B_DL_BFRS EQ STRT_B_MAXSLOTS+1
03FE 998 ASSUME STRT_B_CIR_TIMR EQ STRT_B_DL_BFRS+1
03FE 999 ASSUME STRT_B_KPA_TIMR EQ STRT_B_CIR_TIMR+1
03FE 1000 ASSUME STRT_W_FAC_NUM EQ STRT_B_KPA_TIMR+1
03FE 1001 ASSUME STRT_W_PROD_TYP EQ STRT_W_FAC_NUM+2
00000400 03FE 1002 GHB_W_STRT_LEN: .BLKW 1 ; Length of start message
0400 1003 GHB_T_STRT_MSG: ; Start of start message fixed data
05DC 0400 1004 .WORD LATSC_MAX_MSGSIZ ; Maximum Ethernet frame size
05 0402 1005 .BYTE LATSC_CUR_VER ; Current LAT protocol version
00 0403 1006 .BYTE LATSC_CUR_ECO ; Current LAT ECO version
00 0404 .1 .BYTE 0 ; Maximum number of Slots per circuit
00 0405 1008 .BYTE 0 ; Number of additional buffers (remote)
00000408 0406 1009 .BLKB 2 ; Remote parameters only
0000 0408 1010 .WORD 0 ; Facility number (not yet implemented)
0003 040A 1011 .WORD LATSC_VMS ; Product type code
040C 1012
040C 1013 ASSUME STRT_B_VAR EQ STRT_W_PROD_TYP+2
000004B1 040C 1014 GHB_T_STRT_VAR: .BLKB STRT_C_VAR_LEN ; Variable portion of start message
04B1 1015 :
04B1 1016 : Standard Start slot template
04B1 1017 :
04B1 1018 GHB_T_STRT_SLOT: ; Start of the start slot data
01 04B1 1019 .BYTE SLT_C_SVC_INIT ; Service class
FF 04B2 1020 .BYTE LATSC_MAX_SLOTSIZ ; Attention slot size
FF 04B3 1021 .BYTE LATSC_MAX_SLOTSIZ ; Data slot size
000004B6 04B4 1022 .BLKB 2 ; Null source and destination slot names
00000005 04B6 1023 GHB_C_STRT_SLOTL = <.-GHB_T_STRT_SLOT>

```

:JAY0002
-1

```

04B6 1025      .SBTTL Port Vector
04B6 1026
04B6 1027 :
04B6 1028 : The associated class driver uses this table to command the port driver.
04B6 1029 : The address of this table is contained in the terminal ucb extension area.
04B6 1030 : The offset definitions are defined by ttydefs.
04B6 1031 :
04B6 1032 :
04B6 1033 :
04B6 1034 :     LAT specific dispatch table
04B6 1035 :
04B6 1036 :     ORDER DEPENDENCY
04B6 1037 :     The following symbol must immediately precede the vector
04B6 1038 :
04B6 1039 :
04B6 1040 :     .ALIGN LONG
04B8 1041 GHB_L_AREA:
000004BC 04B8 1042 .BLKL 1 ; Address of counter vector
04BC 1043 LT$VECTOR:
04BC 1044 $VECINI LT,LT$NULL
04F4 1045 $VEC STARTIO,LT$STARTIO ; Start new output
04C0 1046 $VEC DISCONNECT,LT$DISCONNECT ; Hangup port
04C4 1047 $VEC ABORT,LT$ABORT ; Abort output
04E0 1048 $VEC SET_LINE,LT$FLOW_CHANGE ; Change flow control
04C8 1049 $VEC XON,LT$XON ; Send xon sequence
04D0 1050 $VEC XOFF,LT$XOFF ; Send xoff sequence
04D4 1051 : $VEC DS_SET,LT$DS_SET ; Dataset transitions
04D4 1052 $VECEND
04F8 1053
04F8 1054 LT$NULL: ; Null port routine
05 04F8 1055 RSB

```

```
04F9 1057
04F9 1058 :
04F9 1059 : Stop reasons
04F9 1060 :
04F9 1061
04F9 1062 .MACRO $REASON RSNCD, RSNBIT, RSNSTR, ?L
04F9 1063 .ALIGN WORD
04F9 1064 .SIGNED WORD L-.
04F9 1065 .BYTE RSNCD, RSNBIT
04F9 1066 L: .ASCIC /RSNSTR/
04F9 1067 .ENDM
04F9 1068
00000001 04F9 1069 LAT_C_CRSN_DONE = 1 ; No slots in VC
00000002 04F9 1070 LAT_C_CRSN_IVMSG = 2 ; Invalid message
00000003 04F9 1071 LAT_C_CRSN_IVSCI = 3 ; Invalid slave conn id
00000004 04F9 1072 LAT_C_CRSN_IVMCI = 4 ; invalid master conn id
00000005 04F9 1073 LAT_C_CRSN_DISC = 5 ; Disconnect requested
00000006 04F9 1074 LAT_C_CRSN_PROGRESS = 6 ; No progress being made
00000007 04F9 1075 LAT_C_CRSN_TMO = 7 ; Timeout of circuit
00000008 04F9 1076 LAT_C_CRSN_RETRANS = 8 ; Retransmit count exceeded
00000009 04F9 1077 LAT_C_CRSN_RESOURCE = 9 ; No resources
0000000A 04F9 1078 LAT_C_CRSN_TIMRANG = 10 ; Timer value out of range
04F9 1079
00000000 04F9 1080
04F9 1081 LAT_C_RSN_XXX = 0 ; junk
04F9 1082
04F9 1083 LT_RSN_START:
04F9 1084 $REASON -
04F9 1085 GHBSV_START,-
04F9 1086 LAT_C_RSN_XXX,-
04F9 1087 <other than start msg with a zero src index>
0529 1088 LT_RSN_CSBZERO:
0529 1089 $REASON -
0529 1090 GHBSV_CSBZERO,-
0529 1091 LAT_C_RSN_XXX,-
0529 1092 <circuit index zero>
0541 1093 LT_RSN_CSBRANGE:
0541 1094 $REASON -
0541 1095 GHBSV_CSBRANGE,-
0541 1096 LAT_C_RSN_XXX,-
0541 1097 <circuit index out of range>
0561 1098 LT_RSN_CSBINVALID:
0561 1099 $REASON -
0561 1100 GHBSV_CSBINVALID,-
0561 1101 LAT_C_RSN_XXX,-
0561 1102 <circuit index invalid>
057C 1103 LT_RSN_CSBSTALE:
057C 1104 $REASON -
057C 1105 GHBSV_CSBSTALE,-
057C 1106 LAT_C_RSN_XXX,-
057C 1107 <circuit stale reference>
0598 1108 LT_RSN_HALT:
0598 1109 $REASON -
0598 1110 GHBSV_HALT,-
0598 1111 LAT_C_RSN_XXX,-
0598 1112 <circuit forced to halt>
05B3 1113 LT_RSN_INVALIDLOCID:
```



```
05B3 1114 $REASON -
05B3 1115 GHBSV_INVALIDLOCID,-
05B3 1116 LAT_C_RSN_XXX,-
05B3 1117 <invalid local slot id>
05CE 1118 LT_RSN_INVALIDREMID:
05CE 1119 $REASON -
05CE 1120 GHBSV_INVALIDREMID,-
05CE 1121 LAT_C_RSN_XXX,-
05CE 1122 <invalid remote slot id>
05E9 1123 LT_RSN_BADCREDITS:
05E9 1124 $REASON -
05E9 1125 GHBSV_BADCREDITS,-
05E9 1126 LAT_C_RSN_XXX,-
05E9 1127 <bad number of credits in slot>
060C 1128 LT_RSN_REPCREATE:
060C 1129 $REASON -
060C 1130 GHBSV_REPCREATE,-
060C 1131 LAT_C_RSN_XXX,-
060C 1132 <repeat create of slot by server>
0630 1133 LT_RSN_REPDISC:
0630 1134 $REASON -
0630 1135 GHBSV_REPDISC,-
0630 1136 LAT_C_RSN_XXX,-
0630 1137 <repeat disconnect of slot by server>
0658 1138
0658 1139 .ALIGN LONG
0658 1140
```



```

      05 15 0695 1198      BLEQ 40$      ; Br if yes, powerfail
      80 51 C0 0697 1199      ADDL R1,(R0)+  ; Else, add bias
      F7 11 069A 1200      BRB 30$      ; Loop til done
      069C 1201 40$:
      075D'CF 9E 069C 1202      MOVAB LT$STRETRY,-  ; Set address of start entry point
      FA8D'CF      06A0 1203      GHB$L STRETRY+GHB_AREA ;
      0835'CF 9E 06A3 1204      MOVAB LT$SHUTENTRY,-  ; Set address shutdown entry point
      FA82'CF      06A7 1205      GHB$L SHUTENTRY+GHB_AREA;
      0965'CF 9E 06AA 1206      MOVAB LT$SETENTRY,-  ; Set address of multicast set entry
      FA73'CF      06AE 1207      GHB$L SETENTRY+GHB_AREA ;
      FA6B'CF 9E 06B1 1208      MOVAB GHB_AREA,-  ; Save address of the common area
      FE00'CF      06B5 1209      GHB_L AREA      ; so we can find them from LATCP
      00000000'GF 90 06B8 1210      MOVB G^EXE$GQ SYSTIME,- ; Get a random number
      FAD8'CF      06BE 1211      GHB_B_INCARN    ; and save for the message seed value
      06C1 1212
      FAD9'CF 9E 06C1 1213      MOVAB GHB_T_MC DATA,- ; Address of multicast data area
      FA74'CF      06C5 1214      GHB$L_NODE+GHB_AREA
      06C8 1215
      06C8 1216      ASSUME GHB$B_LATVERSION EQ GHB$W_VMSVERSION+2
      06C8 1217      ASSUME GHB$B_LATECO EQ GHB$B_LATVERSION+1
      00050004 8F D0 06C8 1218      MOVL #<<LAT$C_CUR_ECO>>@8! - ; Store current ECO
      FA4F'CF      06CE
      06D1 1219      LAT$C_CUR_VER>@16! - ; and VERSION
      06D1 1220      LAT$C_VMS_VER,-  ; and VMS VERSION
      06D1 1221      GHB$W_VMSVERSION+GHB_AREA
      FAC4'CF 0A 90 06D1 1222      MOVB #LAT$C_KEEP_CSB,-  ; Set number of CSBs to keep around
      F95E'CF 9E 06D3 1223      GHB_B_OLD_CSBCNT
      FA63'CF      06DA 1224      MOVAB GHB_L_CSBTABLE,-  ; Store address of CSB table
      50 FA63'CF 9E 06DD 1225      GHB$L_CSBLST+GHB_AREA
      60 07 D5 06DD 1226      MOVAB GHB$Q_OLD_CSBS+GHB_AREA,R0 ; Get address of old CSB entries
      07 12 06E2 1227      TSTL (R0)      ; Already initialized?
      60 50 D0 06E4 1228      BNEQ 50$      ; Br if yes, skip next section
      04 A0 50 D0 06E6 1229      MOVL R0,(R0)   ; Initialize listhead
      06E9 1230      MOVL R0,4(R0)
      06ED 1231      ; ...
      06ED 1232 50$:
      06ED .1      ; Set up the maximum circuit session limit value for this processor
      06ED .2
      00000000'GF 9A 06ED .3      MOVZBL G^EXE$GB_CPUTYPE,R0 ; Get CPU type code
      50 10 91 06F3
      08 1E 06F4 .4      CMPB #MAX_PR_SID,R0 ; Within range?
      40 8F 90 06F7 .5      BGEQU 55$      ; Branch if so
      FA0D'CF      06F9 .6      MOVB #LAT$C_MAX_SLOTS,-  ; Else load default maximum
      06FF .7
      08 11 06FF .8      BRB W^GHB_B_MAX_SLOTS ; And branch
      FA07'CF 40 90 0701 .9 55$:      MOVB GHB_T_MAX_SLOT_TABLE[R0],- ; Load maximum session count
      FA03'CF      0706
      0709 .10      W^GHB_B_MAX_SLOTS ;
      0709 .11      ;
      0709 .12      ; Setup the initial rating value, based on memory size
      0709 .13
      00000000'GF D0 0709 .14 60$:      MOVL G^MMG$GL_MAXPFN,R0 ; Get the max PFN count
      50 070F
      -3 00CJ2000 8F C6 0710 1236      DIVL #8192,R0 ; Weight by 4MB chunks
      50 0716
      50 08 50 C3 0717 1237      SUBL3 R0,#8,R0 ; Compute for up to 32 MB systems

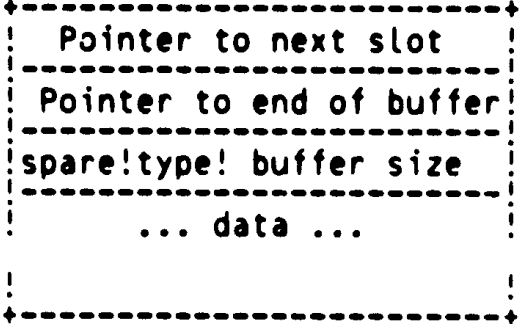
```

```

02 18 071B 1238      BGEQ 70$           ; Br if result okay
50 D4 071D 1239      CLRL R0            ; Else, more than 32 MB - impressive!
FF 8F 50 83 071F 1240 70$: SUBB3 R0,#255,GHB_B_RATE ; Store initial rating
FA75 CF           0723
05 0726 1241      RSB
0727 1242
0727 1243      .IF DEFINED LT_HISTORY
```

```

0727 1245 .SBTTL LT$HISTORY - Save history in the buffer
0727 1246 :++
0727 1247 : LT$HISTORY - Save history in the buffer
0727 1248 :
0727 1249 : Save history on transmits and receives.
0727 1250 : The history buffer is defined as follows:
0727 1251 :
0727 1252 :
0727 1253 :
0727 1254 :
0727 1255 :
0727 1256 :
0727 1257 :
0727 1258 :
0727 1259 :
0727 1260 :
0727 1261 :
0727 1262 :
0727 1263 :
0727 1264 : Inputs:
0727 1265 : R0 -> data to save
0727 1266 :
0727 1267 : Outputs:
0727 1268 : R1 is clobbered
0727 1269 :--
0727 1270 :
00000020 0727 1271 : SLOT_SIZE = 32 ; Size of each history slot
0727 1272 :
0727 1273 LT$HISTORY:
51 F9FD CF D0 0727 1274 MOVL GHBSL_HISTORY+GHB_AREA,R1 ; Point to history buffer
2E 13 0727 1275 BEQL 90$ ; Br if none
0727 1276 :
50 DD 0727 1277 PUSHL R0 ; Save R0
20 C1 0730 1278 ADDL3 #SLOT_SIZE,- ; Compute end address for this slot
04 A1 50 D1 0732 1279 HBF_L_NEXT(R1),R0
04 A1 50 D1 0734 1280 CMPL R0,HBF_L_BUFEND(R1) ; Are we past the end?
0C A1 9E 0738 1281 BLEQU 30$ ; Br if no, continue
51 61 D0 073A 1282 MOVAB HBF_L_DATA(R1),- ; Else, back to beginning
00000000 GF 7D 073D 1283 HBF_L_NEXT(R1)
81 81 7D 073E 1284 30$: MOVL HBF_L_NEXT(R1),R1 ; Get address of next available slot
81 81 7D 0741 1285 MOVQ G^EXE$GQ_SYSTIME,(R1)+ ; Store time in first quadword
50 6E D0 0747 :
81 80 7D 0748 1286 MOVL (SP),R0 ; Get back buffer address
81 80 7D 074B 1287 MOVQ (R0)+,(R1)+ ; Save the data
81 80 7D 074E 1288 MOVQ (R0)+,(R1)+ ; Save the data
81 80 7D 0751 1289 MOVQ (R0)+,(R1)+ ; Save the data
F9CF DF 51 D0 0754 1290 ASSUME HBF_L_NEXT EQ 0
50 BED0 0754 1291 MOVL R1,GHBSL_HISTORY+GHB_AREA ; Save address of next slot
0759 1292 POPL R0 ; Restore R0
075C 1293 :
05 075C 1294 90$: RSB
075D 1295 .ENDC ;; DEFINED LT_HISTORY
    
```



```

075D 1297      .SBTTL LT$STRETRY - Entry to start driver
075D 1298      :++
075D 1299      : LT$STRETRY - Start LTDRIVER entry point
075D 1300      :
075D 1301      : Functional description:
075D 1302      :
075D 1303      : This entry is called by the startup program to initialize the
075D 1304      : driver. Here we save the datalink UCB address, allocate the CSB's
075D 1305      : and start the first receive.
075D 1306      :
075D 1307      : Inputs:
075D 1308      : R5 = UCB address of datalink
075D 1309      :
075D 1310      : Implicit inputs:
075D 1311      :
075D 1312      : Called in user context.
075D 1313      :
075D 1314      : Outputs:
075D 1315      : R0 = success or failure
075D 1316      :--
075D 1317      :
075D 1318      :
075D 1319      : LT$STRETRY:
075D 1320      : DSBINT #LAT$C_IPL ; Raise IPL to prevent interruption
F9CC CF 55 D0 0763 1321 : MOVL R5,GHB$L_UCB+GHB_AREA ; Save the datalink UCB address
0768 1322 :
59 F8CC CF 9E 0768 1323 : MOVAB GHB_L_CSBTABLE,R9 ; Get CSB table base address
SA 20 9A 076D 1324 : MOVZBL #LAT$C_MAX_CSBS,R10 ; Get table length in longwords
FB 89 D4 0770 1325 20$: CLRL (R9)+ ; Zero the whole table
FB 5A F5 0772 1326 : SOBGTR R10,20$ ; Do whole table
0775 1327 :
0775 1328 : Save the last CPU idle value
0775 1329 :
FA13 DF D0 0775 1330 : MOVL @GHB_L_NULLCPU,- ; Save last idle value
FA14 CF 0779 1331 : GHB_L_LASTCPU
00000000'GF D0 077C 1332 : MOVL G^EXE$GL_MP,R0 ; Get ptr to MP code
50 0782 :
07 13 0783 1333 : BEQL 40$ ; Br if not there
0000'CO D0 0785 1334 : MOVL MP$SGL_NULLCPU(R0),- ; Else, get Secondary null time
FA08 CF 0789 1335 : GHB_L_NULLSEC
078C 1336 40$: :
078C 1337 : See if the datalink supports the fast interface and if so
078C 1338 : then allocate an FFI block and initialize that interface.
078C 1339 :
50 D4 078C 1340 : CLRL R0 ; Assume error
51 24 A5 D0 078E 1341 : MOVL UCBSL_CRB(R5), R1 ; Get address of datalink's CRB
51 40 A1 D0 0792 1342 : MOVL CRBSL_INTD+VECSL_START(R1), R1 ; Get address of FFI INIT routine
36 13 0796 1343 : BEQL 70$ ; Br if not present
0045 30 0798 1344 : BSBW LT$GETFFI ; Else, create and initialize FFI
30 50 E9 079B 1345 : BLBC R0,70$ ; Br if error
F9E5 CF 54 D0 079E 1346 : MOVL R4,GHB_L_FFI ; Else, save FFI block address
07A3 1347 :
07A3 1348 : CLRBIT #GHB_STS_V_MULT1,- ; Clear the multicast bury flag
07A3 1349 : GHB_B_STATUS
FC43 CF D4 07A9 1350 : CLRL GHB_C_MULTIBFR ; No multicast buffer is possible
01B5 30 07AD 1351 : BSBW LT$SETENTRY ; Setup multicast message
07B0 1352 :

```

50	F9CC CF	D0	07B0	1353		MOVL	GHB_L_UCB0, R0	:	Set unit seed in the ucb
	17	13	07B5	1354		BEQL	70\$:	No ucb0, bad news
	60	B4	07B7	1355		CLRW	UCB\$W_MB_SEED(R0)	:	Start with unit 1
	105C	30	07B9	1356		BSBW	LT\$SET_TIMER	:	Start timer, so we always have a tick
			07BC	1357					
55	F974 CF	D0	07BC	1358		MOVL	GHB\$L_UCB+GHB_AREA, R5	:	Get datalink UCB address
	5C A5	B6	07C1	1359		INCW	UCB\$W_REFC(R5)	:	One more ref to keep it around for us
	8000 8F	A8	07C4	1360		BISW	#^x8000,UCB\$W_DEVSTS(R5)	:	Allow unit to be re-started
	68 A5		07C8						
			07CA	1361					automatically by datalink
50	00'8F	9A	07CA	1362		MOVZBL	#SS\$ NORMAL, R0	:	Return success
	0B 50	E8	07CE	1363	70\$:	BLBS	R0, 80\$:	Br if okay
	F95F CF	D4	07D1	1364		CLRL	GHB\$L_UCB+GHB_AREA	:	Else, forget about datalink UCB
	SE	10	07D5	1365		BSBB	LT\$SHOTENTRY	:	Clean it all up
50	0000'8F	3C	07D7	1366		MOVZWL	#SS\$_INSFMEM, R0	:	Return not enough memory
			07DC	1367	80\$:	ENBINT		:	Restore IPL
		05	07DF	1368		RSB			
			07E0	1369					

```

07E0 1371      .SBTTL  LT$GETFFI - Get the FFI block and initialize FFI interface
07E0 1372      :++
07E0 1373      : LT$GETFFI - Get FFI block
07E0 1374      :
07E0 1375      : Functional description:
07E0 1376      :
07E0 1377      :     Initialize the FFI interface.
07E0 1378      :
07E0 1379      : Inputs:
07E0 1380      :
07E0 1381      :     R1 = Address of FFI INIT routine in datalink driver
07E0 1382      :     R5 = UCB address of datalink
07E0 1383      :
07E0 1384      : Outputs:
07E0 1385      :
07E0 1386      :     R0 = Status
07E0 1387      :     R1-R4 are destroyed.
07E0 1388      :
07E0 1389      :--
07E0 1390
07E0 1391  LT$GETFFI:
07E0 1392      PUSHL  R1                ; Save init routine address
07E2 1393      MOVZBL #FFISC_LENGTH,R1    ; Set length of buffer to allocate
07E5 1394      JSB    G*EXESALONONPAGED ; Try to allocate an FFI block
07EB 1395      BLBC  R0,30$          ; Br if error
07EE 1396      MOVL  R2,R4                ; Copy FFI block address
07F1 1397      ASSUME FFISL_FL EQ 0
07F1 1398      ASSUME FFISL_BL EQ FFISL_FL+4
07F1 1399      ASSUME FFISW_SIZE EQ FFISL_BL+4
07F1 1400      ASSUME FFISB_TYPE EQ FFISW_SIZE+2
07F1 1401      ASSUME FFISB_SPARE EQ FFISB_TYPE+1
07F1 1402      CLRQ  (R2)+                ; Clear link pointers
07F3 1403      MOVL  #<DYN$C_BUFIO@16>!FFISC_LENGTH,(R2)+ ; Store block size & type
07F9
07FA 1404      ASSUME FFISL_CTX_DL F7 FFISB_SPARE+1
07FA 1405      ASSUME FFISL_XMIT EQ FFISL_CTX_DL+4
07FA 1406      ASSUME FFISL_XMIT_DONE EQ FFISC_XMIT+4
07FA 1407      ASSUME FFISL_RECV_DONE EQ FFISL_XMIT_DONE+4
07FA 1408      ASSUME FFISL_ERROR EQ FFISL_RECV_DONE+4
07FA 1409      ASSUME FFISL_SHUT_DONE EQ FFISL_ERROR+4
07FA 1410      CLRQ  (R2)+                ; Zero CTX & XMIT routine address
07FC 1411      MOVAB W*LT$XMT_FFIDONE,(R2)+ ; Store XMIT Complete routine address
0801 1412      MOVAB W*LT$FFI_RCV_MSG,(R2)+ ; Store RECV Complete routine address
0806 1413      MOVAB W*LT$ASYNCERR,(R2)+   ; Store address of ERROR handler
080B 1414      MOVAB W*LT$SHUT_DONE,(R2)+   ; Store address of SHUTDOWN handler
0810 1415      ASSUME FFISL_SPARE0 EQ FFISL_SHUT_DONE+4
0810 1416      ASSUME FFISL_SPARE1 EQ FFISL_SPARE0+4
0810 1417      ASSUME FFISL_SPARE2 EQ FFISL_SPARE1+4
0810 1418      ASSUME FFISL_SPARE3 EQ FFISL_SPARE2+4
0810 1419      CLRQ  (R2)+                ; Not used
0812 1420      CLRQ  (R2)+
0814 1421      ASSUME FFISL_DL_UCB EQ FFISL_SPARE3+4
0814 1422      ASSUME FFISL_PID EQ FFISL_DL_UCB+4
0814 1423      ASSUME FFISW_CHAN EQ FFISC_PID+4
0814 1424      MOVL  R5,(R2)+                ; Store datalink UCB address
0817 1425      CLRL  (R2)+                ; Zero PID
0819 1426      CLRW  (R2)+                ; Zero CHAN
    
```


50	00'8F	9A	081B	1427	MOVZBL	#SS\$ _NORMAL,R0	:	Return success
	51	8ED0	081F	1428	30\$:	POPL	R1	: Restore R1
	OF	50	E9	0822	1429	BLBC	R0,90\$: Br if allocation error
		53	D4	0825	1430	CLRL	R3	: No parameters
		61	16	0827	1431	JSB	(R1)	: Else, call datalink init routine
	08	50	E8	0829	1432	BLBS	R0,90\$: Br if success
	50	54	D0	082C	1433	MOVL	R4,R0	: Else, copy FFI block address
		13A7	30	082F	1434	BSBW	LT\$DEALPOOL	: Deallocate the FFI block
		50	D4	0832	1435	CLRL	R0	: Return failure
			05	0834	1436	90\$:	RSB	

```

0835 1438 .SBTTL LTSSHUTENTRY - Shutdown entry point
0835 1439 :++
0835 1440 : LTSSHUTENTRY - Shutdown LTDRIVER entry point
0835 1441 :
0835 1442 : Functional description:
0835 1443 :
0835 1444 :     Shutdown things when we are done.
0835 1445 :
0835 1446 : Inputs:
0835 1447 :     none
0835 1448 :
0835 1449 : Implicit inputs:
0835 1450 :
0835 1451 :     Called in user context.
0835 1452 :
0835 1453 : Outputs:
0835 1454 :     none
0835 1455 :--
0835 1456
0835 1457 LTSSHUTENTRY:
0835 1458     DSBINT #LAT$C IPL ; Sync with rest of things
0835 1459     SETBIT #GHB_STS V SHUT,- ; Indicate that shutdown has been
0835 1460             GHB_B_STATUS ; requested
0841 1461 :
0841 1462 :
0841 1463 :     Deallocate and shut down all the circuits
0841 1464 :
59  F7F3 CF DE 0841 1465     MOVAL GHB_L_CSBTABLE, R9 ; Table of CSB addresses
    5A 20 9A 0846 1466     MOVZBL #LAT$C_MAX_CSBS, R10 ; and its length
    58 89 D0 0849 1467 40$: MOVL (R9)+, R8 ; Get a CSB address
    03 13 084C 1468     BEQL 60$ ; No CSB address here
    OED9 30 084E 1469     BSBW LT$CIRCDEAD ; Kill all processes on this one
    F5 5A F5 0851 1470 60$: SOBGTR R10, 40$ ; Til end of table
    0854 1471 :
    0854 1472 :
    0854 1473 :     We can reduce our interest in the datalink UCB here early because
    0854 1474 :     the LAT control program still has a channel to the UCB and is still
    0854 1475 :     active. The UCB will disappear when the LATCP deassigns its last
    0854 1476 :     channel to it.
    0854 1477 :
    0854 1478 :
59  F8DC CF D0 0854 1479     MOVL GHB$L_UCB+GHB_AREA, R9 ; Get datalink UCB and reduce our interest
    19 13 0859 1480     BEQL 80$ ; No ucb here
    8000 8F AA 085B 1481     BICW #^x8000,UCBSW_DEVSTS(R9); Don't want datalink to re-start
    68 A9 085F 1482 :
    5C A9 B7 0861 1483 :     any more!
    F8CC CF D4 0861 1483     DECW UCBSW_REFC(R9) ; Let datalink UCB go away.
    0864 1484     CLRL GHB$L_UCB+GHB_AREA ; Forget we ever had it
    0868 1485 :     This prevents more circuits from
    0868 1486 :     being accepted.
    0868 1487 :
    0868 1488 :     Wait for any datalink re-start timer to expire. Spin on
    0868 1489 :     the INTERLOCK bit.
    0868 1490 :
    0868 1491     SETIPL #IPL$ASTDEL ; Prevent us from being deleted,
    0868 1492 :     but allow scheduling activity
04 68 A9 OE E1 086B 1493 70$: BBC #14,UCBSW_DEVSTS(R9),80$ ; Proceed if UCB can evaporate

```

```

70 10 0870 1494 BSBB BUILD_TQE ; Wait around for a while
F7 11 0872 1495 BRB 70$ ; Try again
    0874 1496
    0874 1497 80$:
    0874 1498
    0874 1499
    0874 1500
    0874 1501
    0874 1502
    0874 1503
    0874 1504
    0874 1505
    0877 1506
    02 11 0877 1507 BRB 110$ ; Prevent us from being deleted,
    0879 1508 ; but allow scheduling activity
    0879 1509 ; See if CSB's are all gone
    0879 1510 ; Now delay while we wait for CSB's to run down
    67 10 0879 1511 100$: BSBB BUILD_TQE ; Build TQE to wait on and wait
    087B 1512
59 F7B9 CF DE 087B 1513 110$: MOVAL GHB_L_CSBTABLE,R9 ; Table of CSB addresses
    SA 20 9A 0880 1514 MOVZBL #LAT$C_MAX_CSBS, R10 ; and its length
    58 89 D0 0883 1515 120$: MOVL (R9)+, R8 ; Get a CSB address
    F1 12 0886 1516 BNEQ 100$ ; Just spin here looking
    F8 5A F5 0888 1517 SOBGR R10, 120$ ; Til end of table
    088B 1518
    088B 1519 ; Now wait for all the UCBs to dissappear except unit 0
    088B 1520
    02 11 088B 1521 BRB 160$ ; Check if we're all done
    088D 1522
    53 10 088D 1523 140$: BSBB BUILD_TQE ; Build TQE to wait on and wait
    088F 1524
    088F 1525 ; Now delay while we wait for terminal UCB's to run down
    088F 1526
50 FBED CF D0 088F 1527 160$: MOVL GHB_L_UCB0,R0 ; Get the LT UCB address
    OA 13 0894 1528 BEQL 180$ ; ???
    51 24 A0 D0 0896 1529 MOVL UCBSL_CRB(R0),R1 ; Get the CRB address
    01 0C A1 B1 089A 1530 CMPW CRB$W_REFC(R1),#1 ; All done ?
    ED 12 089E 1531 BNEQ 140$ ; Br if no
    08A0 1532 180$:
    08A0 1533 ; Reset completion routines for datalink's FFI interface
    08A0 1534
    08A0 1535
55 F8E1 CF D0 08A3 1536 SETIPL #LAT$C_IPL ; Sync access to rest of driver
    OC 13 08A8 1537 MOVL GHB_L_FFI, R5 ; Get FFI block address
    1BD8 CF 9E 08AA 1538 BEQL 200$ ; Br if none
    18 A5 08AE 1539 MOVAB W^LT$DROPCXB,- ; Drop all RECEIVE buffers
    1BCF CF 9E 08B0 1540 MOVAB W^LT$FFIPOSTDONE,- ; Deallocate all TRANSMIT CXBs
    14 A5 08B4 1541 FFISL_XMIT_DONE(R5)
    08B6 1542 200$:
    08B6 1543 ; Deallocate the multicast buffer
    08B6 1544
50 FB36 CF D0 08B6 1545 MOVL GHB_L_MULTIBFR,R0 ; Get the multicast buffer
    OB 13 08BB 1546 BEQL 220$ ; Br if none
    08BD 1547 ASSUME GHB_STS_V_MULTI EQ 0
    03 F8D7 CF EB 08BD 1548 BLBS GHB_B_STATUS,220$ ; Br if multicast buffer is active
    1314 30 08C2 1549 BSBW LT$DEALPOOL ; Else, deallocate the multicast buffer
    FB27 CF D4 08C5 1550 220$: CLRL GHB_L_MULTIBFR ; Forget we had a multicast buffer

```

```
08C9 1551      ;  
08C9 1552      ; Flush all CSBs on the old CSB display queue  
08C9 1553      ;  
50  F877 DF 0F 08C9 1554 240$: REMQUE @GHB$Q_OLD_CSBS+GHB_AREA,R0 ; Get CSB address  
      05 1D 08CE 1555      BVS 300$ ; Br if none  
      1306 30 08D0 1556      BSBw LT$DEALPOOL ; Else, deallocate the CSB  
      F4 11 08D3 1557      BRB 240$ ; Loop til done  
      OA 90 08D5 1558 300$: MOVB #LAT$C_KEEP_CSBS,- ; Reset number of CSBs to keep around  
      F8C0 CF 08D7 1559      GHB_B_OLD_CSBCNT  
      08DA 1560      ;  
      08DA 1561      ; Mark LTDRIVER as inactive  
      08DA 1562      ;  
      F8BA CF 94 08DA 1563      CLRB GHB_B_STATUS ; Clear ACTIVE flag to stop timer  
      08DE 1564      ; clear SHUTdown flag  
      08DE 1565      ; clear MULTicast buffer flag  
      08DE 1566      ENBINT  
      05 08E1 1567      RSB  
      08E2 1568
```

```

08E2 1570      .SBTTL TQE TIMER BUILDING AND WAITING ROUTINES
08E2 1571      :++
08E2 1572      : BUILD_TQE - Build a TQE
08E2 1573      :
08E2 1574      : Inputs:
08E2 1575      :     IPL = ASTDEL
08E2 1576      :
08E2 1577      : Implicit inputs:
08E2 1578      :     Called in user context
08E2 1579      :
08E2 1580      :
08E2 1581      : Outputs:
08E2 1582      :     R0-R5 are destroyed
08E2 1583      :--
08E2 1584      :
08E2 1585      BUILD_TQE:
00000000'GF 16 08E2 1586      JSB      G^EXES$ALLOCTQE      ; Allocate a TQE (returns IPL = ASTDEL)
                   4A 50 E9 08E8 1587      BLBC      R0,90$      ; Br if allocation failure
08 A2 01 90 08EB 1588      MOVB      #TQESC_SSSNGL,TQESB_RQTYPE(R2) ; Indicate single shot request
OC A2 36'AF 9E 08EF 1589      MOVAB     B^SHUT_TIMER,TQESL_FPC(R2) ; Set wake up routine address
                   7E DC 08F4 1590      MOVPSL   -(SP)      ; Push current PSL to resume process
08F6 1591      ASSUME     IPL$ SYNCH EQ IPL$_TIMER
08F6 1592      SETIPL   #IPL$ SYNCH      ; Sync access to system data base
00000000'GF 7D 08F9 1593      MOVQ      G^EXES$GQ_SYSTIME,R0      ; Get current time
                   50
004C4B40 8F C0 0900 1594      ADDL      #LAT$C_SHUTDELAY,R0      ; Calculate expiration time
                   50
                   51 00 DB 0907 1595      ADWC      #0,R1      ; Add carry to top half
                   55 52 DO 090A 1596      MOVL      R2,R5      ; Copy TQE address
00000000'GF 16 090D 1597      JSB      G^EXES$INSTIME      ; Insert TQE on timer queue
                   08 11 0913 1598      CLRBIT   #GHB_STS_V TQE,-      ; Indicate TQE not complete - yet
                   03 E0 0913 1599      GHB_B_STATUS
                   03 E0 0919 1600      BRB      60$      ; Wait for TQE
14 F878 CF E0 091B 1601 40$: BBS      #GHB_STS_V TQE -      ; Br if TQE is complete
                   7E DC 091D 1602      GHB_B_STATUS,90$
00000000'GF 7E DC 0921 1603      MOVPSL   -(SP)      ; Push current PSL to resume process
                   54 DO 0923 1604 60$: MOVL      G^SCH$GL_CURPCB,R4      ; Get current PCB address
                   50 01 9A 092A 1605      MOVZBL   #RSNS_A$T$WAIT,R0      ; Set waiting resource
00000000'GF 16 092D 1606      JSB      G^SCH$R$WAIT      ; Make process wait for resource
                   E6 11 0933 1607      BRB      40$      ; Check if TQE expired
                   05 0935 1608 90$: RSB      ; Return to caller
                   0936 1609
                   0936 1610      :++
                   0936 1611      : SHUT_TIMER
                   0936 1612      :
                   0936 1613      : This routine wakes up the waiting process
                   0936 1614      :
                   0936 1615      : Inputs:
                   0936 1616      :     R4 = PCB address
                   0936 1617      :     R5 = TQE address
                   0936 1618      :
                   0936 1619      :     IPL = IPL$_TIMER
                   0936 1620      :
                   0936 1621      : Outputs:
                   0936 1622      :     R0 is destroyed
                   0936 1623      :     R5 = TQE address of system non-repeating TQE

```

		0936	1624	;	--				
		0936	1625						
		0936	1626	SHUT_TIMER:					
		0936	1627	SETBIT	#GHB_STS_V TQE,-				; Indicate TQE has completed
		0936	1628		GHB-B STATUS				
50	01	9A	093C	1629	MOVZBL	#RSNS_ASTWAIT,R0			; Set waiting resource
00000000	'GF	16	093F	1630	JSB	G^SCH\$RAVAIL			; Mark resource available
50	55	D0	0945	1631	MOVL	R5,R0			; Copy TQE address
	128E	30	0948	1632	BSBW	LT\$DEALPOOL			; Deallocate the TQE
00000000	'GF	DE	094B	1633	MOVAL	G^EXESAL_TQENOREPT,R5			; Setup SYSTEM no repeat TQE address
	55		0951						
		05	0952	1634	RSB				; Return to caller

```

0953 1636 .SBTTL LTSSHUT_DONE - Datalink shutdown complete entry
0953 1637 :++
0953 1638 : LTSSHUT_DONE - Datalink shutdown complete entry point
0953 1639 :
0953 1640 : Functional description:
0953 1641 :
0953 1642 : Deallocate the FFI block when datalink is all done.
0953 1643 :
0953 1644 : inputs:
0953 1645 : R4 = FFI block address
0953 1646 :
0953 1647 : outputs:
0953 1648 : R0,R1,R4 are destroyed
0953 1649 : All other registers must be preserved
0953 1650 :--
0953 1651
0953 1652 LTSSHUT_DONE::
50 2C BB 0953 1653 PUSHR #*M<R2,R3,R5> ; Save registers
    FB2F CF D0 0955 1654 MOVL GHB_L_FFI,R0 ; Get FFI block address
    FB2A CF D4 095A 1655 CLRL GHB_L_FFI ; Zero pointer to FFI block
    1278 30 095E 1656 BSBW LT$DEALPOOL ; Deallocate the FFI block
    2C BA 0961 1657 POPR #*M<R2,R3,R5> ; Restore registers
    05 0963 1658 RSB

```

```
0964 1660 .SBTTL LTSASYNCERR - Process async errors from datalink
0964 1661 :++
0964 1662 : LTSASYNCERR - Process async errors from datalink
0964 1663 :
0964 1664 : Functional description:
0964 1665 :
0964 1666 : Do Nothing.
0964 1667 :
0964 1668 :--
05 0964 1669 LTSASYNCERR:
0964 1670 RSB
```



```

0965 1672 .SBTTL LT$SETENTRY - Set a new multicast message
0965 1673 ;++
0965 1674 : LT$SETENTRY - Set new multicast message
0965 1675 :
0965 1676 : Functional description:
0965 1677 :
0965 1678 : Take the data from the multicast data area and store it in a new message
0965 1679 : or bump the incarnation so we know to do it next time.
0965 1680 :
0965 1681 : Inputs:
0965 1682 :     None.
0965 1683 :
0965 1684 : Outputs:
0965 1685 :     R0 = success or failure
0965 1686 :     All other registers are preserved.
0965 1687 :
0965 1688 : Implicit outputs:
0965 1689 :     Multicast timer set to 1 second interval.
0965 1690 :
0965 1691 :--
0965 1692
0965 1693 LT$SETENTRY:
0965 1694 DSBINT #LATSC_IPL
007E 8F BB 0968 1695 PUSHR #^M<R1,R2,R3,R4,R5,R6> : Save registers
           56 D4 096F 1696 CLRL R6 : Assume no old CXB change flags
52 FA7B CF D0 0971 1697 MOVL GHB_L_MULTIBFR, R2 : Get the Multicast CXB
           1A 13 0976 1698 BEQL 10$ : Br if none, go make one
           F81D CF 96 0978 1699 INCB GHB_B_INCARN : Bump the incarnation count
03 F818 CF E9 097C 1700 ASSUME GHB_STS_V_MULT| EQ 0
           0099 31 097C 1701 BLBC GHB_B_STATUS, 5$ : Br if multicast buffer is free
           0981 1702 BRW 70$ : Else, do this operation later
           0984 1703
           0984 1704 : Multicast buffer is available, deallocate it!
           0984 1705
56 4F A2 90 0984 1706 5$: MOVB XMT_B_MC_CHG_FLAG(R2),R6: Save previous change flags
           50 52 D0 0988 1707 MOVL R2,-R0 : Deallocate the old CXB buffer
           FA61 CF D4 0988 1708 CLRL GHB_L_MULTIBFR : NO more multicast CXB buffer
           1247 30 098F 1709 BSBW LT$DEALPOOL : Dump the buffer
           0992 1710
51 F806 CF 3C 0992 1711 10$: MOVZWL GHB_W_MC_SIZE, R1 : Get size of data set by LATCP
           00000052 8F C0 0997 1712 ADDL #XMT_B_MC_SET, R1 : plus message overhead
           099D
           00000000 GF 16 099E 1713 JSB G^EXESALONONPAGED : Now get the buffer
           7B 50 E9 09A4 1714 BLBC R0, 80$ : No pool for buffer
           08 A2 51 B0 09A7 1715 MOVW R1, CXBSW_SIZE(R2) : Size of buffer
           09AB 1716 ASSUME CXBSB_CODE EQ CXBSB_TYPE+1
           0119 8F B0 09AB 1717 MOVW #<1085!DYN$C CXB,- : Set type and indicate multicast
           0A A2 B0 09AF 1718 CXBSB_TYPE(R2) : buffer
           0048 8F B0 09B1 1719 MOVW #XMT_T_DATA,(CXBSW_BOFF(R2)) : Set offset to start of data
           18 A2
FA34 CF 52 D0 09B7 1720 MOVL R2, GHB_L_MULTIBFR : Save address of the buffer
           098C 1721
           098C 1722 : Copy fixed portion of multicast message
           098C 1723
51 FA34 CF 9E 098C 1724 MOVAB GHB_T_MC_MSG,R1 : Point to msg template
           61 0A 28 09C1 1725 MOVCS #XMT_C_MC_LENGTH,(R1),- : Copy the fixed portion of message
           48 A2 09C4 1726 XMT_T_DATA(R2)
    
```

```

09C6 1727
09C6 1728
09C6 1729
51 F7D4 CF 9E 09C6 1730
61 F7CD CF 28 09CB 1731
    63
09D0
09D1 1732
09D1 1733
09D1 1734
52 FA1B CF D0 09D1 1735
56 FF 8F 8D 09D6 1736
    4F A2
09DA
09DC 1737
    F7B9 CF 90 09DC 1738
    4E A2 09E0 1739
50 48 A2 9E 09E2 1740
    53 50 C2 09E6 1741
1A A2 53 B0 09E9 1742
    09ED 1743
    09ED 1744
    09ED 1745
51 F7AF CF 9E 09ED 1746
53 FA16 CF 9E 09F2 1747
    56 53 D0 09F7 1748
    50 81 9A 09FA 1749
    51 50 C0 09FD 1750
    50 61 9B 0A00 1751
    50 50 B6 0A03 1752
63 61 50 28 0A05 1753
    50 61 9B 0A09 1754
    50 50 B6 0A0C 1755
63 61 50 28 0A0E 1756
    83 B4 0A12 1757
    53 56 C2 0A14 1758
    53 0C A1 0A17 1759
    F9E1 CF 0A1A
    0A1D 1760
50 0000'8F 3C 0A1D 1761 70$:
F9C7 CF 01 B0 0A22 1762 80$:
    007E 8F BA 0A27 1763 90$:
    0A2B 1764
    05 0A2E 1765
    0A2F 1766

: Copy the variable data set by LATCP
MOVAB GHB_T_MC_DATA,R1 ; Get address of variable data
MOVCS GHB_W_MC_SIZE,(R1),(R3) ; Copy the variable data

: Set change flags, and incarnation number
MOVL GHB_L_MULTIBFR,R2 ; Get back the multicast buffer
XOR#B3 #XMT_CFG_M_ALL,R6,- ; Set change flags

XMT_B_MC_CHG_FLAG(R2)
MOVB GHB_B_INCARN, ; Set incarnation number
XMT_B_MC_INCARN(R2)
MOVAB XMT_T_DATA(R2), R0 ; Address of start of data
SUBL R0,-R3 ; Calculate the length
MOVW R3, CXBSW_BCNT(R2) ; Set length of message

: Copy the node name and descriptor for the START message
MOVAB GHB_T_MC_DATA+2,R1 ; Get address of multicast variable data
MOVAB GHB_T_START_VAR,R3 ; Get address of start msg variable data
MOVL R3,R6 ; Save address of start of variable data
MOVZBL (R1)+, R0 ; Get size of group codes
ADDL R0, R1 ; Calculate address of node name
MOVZBW (R1), R0 ; Get size of node name
INCW R0 ; Include the byte count
MOVCS R0,(R1),(R3) ; Copy the node name
MOVZBW (R1), R0 ; Get size of node descriptor
INCW R0 ; Include the byte count
MOVCS R0,(R1),(R3) ; Copy the node descriptor
CLRW (R3)+ ; No location text, no parameters
SUBL R6, R3 ; Calculate length of start message
ADDW3 #START_C_LENGTH, R3, - ; Store length in holding cell

GHB_W_START_LEN
MOVZWL #SSS-NORMAL, R0 ; Return success
MOVW #1, GHB_W_MULTIMR ; Send multicast msg on next tick
POPR #*M<R1,R2,R3,R4,R5,R6> ; Restore registers
ENBINT
RSB
    
```



```

0A84 1822          SETBIT #TTYSV_PC_NOTIME,-      : no timer service please
0A84 1823          UCBSW_TT_PRTCTL(R5)         : ...
0A8A 1824 40$.    :
0A8A 1825          : Finish initializing UCB.
0A8A 1826          :
OF 10 0A8A 1827    BSBB LTSUCB_INIT             : Initialize the terminal UCB
OC 11 0A8C 1828    BRB 90$                      : And leave
50 0134 C5 D0 0A8E 1829          :
66 A0 B0 0A8E 1830 70$:  MOVL UCBSL_LT_CSB(R5), R0 : On power fail, reset the timeout
50 A0 05 13 0A93 1831          : BEQL 90$ : No CSB, just forget it
66 A0 B0 0A95 1832          : MOVW CSB_W_TIMRESET(R0),- : And just wait on some thing
50 A0 05 0A98 1833          : CSB_W_TIMEOUT(R0) : To happen.
05 0A9A 1834 90$:  RSB

```

```

0A9B 1836      .SBTTL LTSUCB_INIT - Initialize a terminal UCB
0A9B 1837      :++
0A9B 1838      : LTSUCB_INIT - Initialize a terminal UCB
0A9B 1839      :
0A9B 1840      : Functional description:
0A9B 1841      :
0A9B 1842      :      Init the terminal UCB before starting a process to use it.
0A9B 1843      :
0A9B 1844      : Inputs:
0A9B 1845      :      R5 = UCB address
0A9B 1846      :
0A9B 1847      : Outputs:
0A9B 1848      :      R5 = UCB address
0A9B 1849      :      R0, R1 clobbered
0A9B 1850      :--
0A9B 1851
0A9B 1852 LTSUCB_INIT:
0080 54 A5 B5 0A9B 1853 TSTW UCBSW_UNIT(R5)      : Is this unit #0?
      3D 13 0A9E 1854 BEQL 90$                  : Br if yes, skip it
      5C A5 B5 0AA0 1855 TSTW UCBSW_REFC(R5)      : Any channels attached?
01   54 A5 21 12 0AA3 1856 BNEQ 40$          : Br if yes, don't reinit UCB
      51 78 0AA5 1857 ASHL UCBSW_UNIT(R5),#1,R1 : Build unit's bit mask
0080 C5 51 B0 0AAA 1858 MOVW R1,UCBSB_ERTCNT(R5)  : Save it
      0141 C5 94 0AAF 1859 CLRB UCBSB_LT_LOCID(R5)  : Flag as available (no local slot indx)
      FA05 CF DE 0AB3 1860 MOVAL LTSVECTOR,-      : Set the port address in ucb
      0118 C5 0AB7 1861 UCBSL_TT_PORT(R5)      : in case port driver reloaded
      00CC C5 D4 0ABA 1862 CLRL UCBSL_TT_WFLINK(R5)  : Zero to force reinit of queue head
      51 0114 C5 D0 0ABE 1863 MOVL UCBSL_TT_CLASS(R5),R1 : Address class vector table
      08 B1 16 0AC3 1864 JSB @CLASS_SETUP_UCB(R1)  : Init UCB fields
      0AC6 1865 40$:
      0142 C5 90 0AC6 1866 MOVW #UCBSM_LT_DATA,-      : Look for data first time
      50 BF 90 0ACB 1867 UCBSB_LT_DATAW(R5)
      014A C5 90 0ACB 1868 MOVW #LATSC_UCB_BUFSIZ,-      : Init buffer
      014B C5 94 0ACE 1869 UCBSB_LT_MAXC(R5)
      0146 C5 94 0AD1 1870 CLRB UCBSB_LT_CURC(R5)
      0120 C5 B4 0AD5 1871 CLRB UCBSB_LT_TCRED(R5)
      05 0AD9 1872 CLRW UCBSW_TT_OUTLEN(R5)
      05 0ADD 1873 90$: RSB
    
```

```

OADE 1875 .SBTTL LT$STARTIO - Start I/O routine
OADE 1876 :++
OADE 1877 : LT$STARTIO - Start I/O Operation
OADE 1878 :
OADE 1879 : Functional description:
OADE 1880 :
OADE 1881 : This routine is entered from the device independent terminal startio
OADE 1882 : Routine to enable output interrupts on an idle unit.
OADE 1883 :
OADE 1884 : Inputs:
OADE 1885 :
OADE 1886 : R3 = character and cc = plus
OADE 1887 : R3 = address and cc = minus and R2 = count
OADE 1888 : R5 = ucb address
OADE 1889 :
OADE 1890 : Outputs:
OADE 1891 :
OADE 1892 : R5 = UCB address
OADE 1893 : R0,R1,R2,R3 and R4 are modified
OADE 1894 :--
OADE 1895
OADE 1896 LT$STARTIO: ; Start I/O on unit
58 0760 8F BB OADE 1897 PUSHR #^M<R5,R6,R8,R9,R10>
0134 C5 D0 OAE2 1898 MOVL UCBS$LT_CS(B5),R8 ; Get Circuit State Block
59 13 OAE7 1899 BEQL 85$ ; Br if UCB is hanging up
OAE9 1900 ;
OAE9 1901 ; Dispatch on our terminal UCB state
OAE9 1902 ;
OAE9 1903 $DISPATCH UCBS$LT_STATE(R5),TYPE=B,-
OAE9 1904 <- ; state ; action
OAE9 1905 'CB$C_LT_STATE_KILL 168$>,- ; KILL state, flush data
OAE9 1906 >
OAF1 1907 ;
OAF1 1908 ; All other states -- okay to send data
OAF1 1909 ;
0142 01 88 OAF1 1910 BISB #UCBSM_LT_DATA,- ; Assume output data available
C5 OAF3 1911 UCBS$LT_DATAW(R5)
OAF6 1912 ;
OAF6 1913 ; Buffer data locally
OAF6 1914 ;
OAF6 1915 5$: $DISPATCH UCBS$TT_OUTTYPE(R5),TYPE=B,-
OAF6 1916 <- ; type ; action
OAF6 1917 < 0 170$>,- ; No data, allow more calls
OAF6 1918 < 1 10$>,- ; Single character
OAF6 1919 >
OB00 1920 ;
OB00 1921 ; Default to burst data
OB00 1922 ;
53 011C C5 D0 OB00 1923 MOVL UCBS$TT_OUTADR(R5),R3 ; Get output buffer address
52 0120 C5 3C OB05 1924 MOVZWL UCBS$TT_OUTLEN(R5),R2 ; and character count
21 11 OB0A 1925 BRB 30$ ; Continue in common path
OB0C 1926 ;
50 014A C5 9E OB0C 1927 10$: MOVAB UCBS$LT_MAXC(R5),R0 ; Get address of maxc
60 80 91 OB11 1928 CMPB (R0)+,(R0) ; Any room left?
02 14 OB14 1929 BGTR 20$ ; Br if yes
60 97 OB16 1930 DECB (R0) ; Make room for it
60 96 OB18 1931 20$: INCB (R0) ; Update count

```

```

51      9A  OB1A  1932      MOVZBL (R0),R1      : Get count
50      51  CO  OB1D  1933      ADDL   R1,R0       : Form destination address
60      53  90  OB20  1934      MOVB   R3,(R0)     : Load character
          OB23  1935      :
          OB23  1936      : We are setting up the address and count of buffered characters in
          OB23  1937      : case we are balanced and we should send these characters in a msg
          OB23  1938      : to come out of balanced mode
          OB23  1939      :
52      014B C5 9A  OB23  1940      MOVZBL UCBSB_LT_CURC(R5), R2 : Get count of buffered characters
53      014C C5 9E  OB28  1941      MOVAB  UCBSB_LT_CBUF(R5), R3 : Get address of buffer
          00  E1  OB2D  1942 30$:  BBC    #FLAG_V_RRF,-      : Br if we can we send an
          03 5C AB  OB2F  1943      CSB_B_FLAG(R8),70$      : unsolicited message
          00E0 31  OB32  1944 60$:  BRW    190$
          OB35  1945      :
          OB35  1946      : Transmit data to concentrator
          OB35  1947      :
          0146 C5 97  OB35  1948 70$:  DECB   UCBSB_LT_TCRED(R5) : Use a credit
          0A 18  OB39  1949      BGEQ   90$          : Br if we have one to use
          0146 C5 96  OB3B  1950 80$:  INCB   UCBSB_LT_TCRED(R5) : Else, no credits
          OB3F  1951      :
          OB3F  1952      : Woops. We find we have data and we can't send it anywhere since
          OB3F  1953      : we have no credit. Never mind we can just leave here. We will
          OB3F  1954      : receive a credit message eventually. We are set non-balanced here
          OB3F  1955      : and if we leave so that all the int bits are turned off in the
          OB3F  1956      : other ucbs, and this one, then we won't get anymore embarrassing
          OB3F  1957      : startio calls. When we get the credit message we will start things
          OB3F  1958      : up again.
          00CE 31  OB3F  1959      :
          OB3F  1960      BRW    180$          : Leave quietly
          OB42  1961      :
          OB42  1962      : We have nowhere to put the data and the circuit is down, so just
          OB42  1963      : clear interrupt expected, discard the data and return
          OB42  1964      :
          00C4 31  OB42  1965      :
          OB42  1966 85$:  BRW    168$          : Leave, flush data & clear INT expected
          OB45  1967      :
          56 34 AB 7E  OB45  1968 90$:  MOVAQ  CSB_Q_XBUFQ(R8),R6 : Get transmit buffer queue address
          56 66 D1  OB49  1969      CML    (R6),-R6       : Is there a transmit buffer?
          ED 13  OB4C  1970      BEQL   80$          : Br if no transmit buffer, ignore it
          5D AB 01 90  OB4E  1971      MOVB   #1, CSB_B_NUM_SLOTS(R8) : We have data for the server now
          OB52  1972      SETBIT  #FLAG_V_RRF,-      : No more unsolicited messages
          OB52  1973      CSB_B_FLAG(R8)
          52 AB 02  B0  OB57  1974      MOVW   #LATSC_HOST_TIMER,- : Set retransmit timer
          OB5B  1975      CSB_W_XMTTMO(R8)
          50 56 66 D0  OB5B  1976      MOVL   (R6),R6       : Get the buffer address
          014C C5 9E  OB5E  1977      MOVAB  UCBSB_LT_CBUF(R5), R0 : Address the UCB extension
          50 53 D1  OB63  1978      CML    R3,R0        : Is that our buffer?
          04 12  OB66  1979      BNEQ   100$        : Br if no
          OB68  1980      : ***
          OB68  1981      : Reading the code I noticed the following line(s) would not work if
          OB68  1982      : the startio entry were to discover that it were balanced after
          OB68  1983      : previous previous calls stored data in the UCB extensions because
          OB68  1984      : we were not balanced. I guess that never happens, because UCBSB_LT_CURC
          OB68  1985      : would not be equal to one, as assumed below, but some higher value. BEM
          OB68  1986      : ***
          014B C5 97  OB68  1987      DECB   UCBSB_LT_CURC(R5) : Yes. Don't save the character there
          OB6C  1988 100$:

```

```

59 50 A6 9E OB6C 1989 MOVAB XMT_T_MDATA(R6), R9 ; Address of the slot to use (first)
5A 0149 C5 9A OB70 1990 MOVZBL UCBSB_LT_SLOTSZ(R5), R10; Get the maximum slot size
      OB75 1991
      OB75 1992 ASSUME SLT_B_DSTID EQ 0
      OB75 1993 ASSUME SLT_B_SRCID EQ <SLT_B_DSTID+1>
      OB75 1994 ASSUME SLT_B_COUNT EQ <SLT_B_SRCID+1>
      OB75 1995 ASSUME SLT_B_CRED EQ <SLT_B_COUNT+1>
      OB75 1996
      OB75 1997 ASSUME UCBSB_LT_LOCID EQ UCBSB_LT_REMID+1
      OB75 1998
89 0140 C5 B0 OB75 1999 MOVW UCBSB_LT_REMID(R5), (R9)+ ; Load remote and local ids
89 89 5A 90 OB7A 2000 MOVB R10, (R9)+ ; Save the character count
      OB7D 2001
      OB7D 2002 ; Extend credits to concentrator if needed
      OB7D 2003
      OB7D 2004 ADDB3 UCBSB_LT_XCRED(R5),- ; See if we need to send
50 01 81 OB81 2005 #LAT$C_MAX_RCRED,R0 ; a credit to the remote
0148 C5 50 80 OB83 2006 ADDB R0,UCBSB_LT_RCRED(R5) ; Add them to total credits to send
0147 C5 50 82 OB88 2007 SUBB R0,UCBSB_LT_XCRED(R5) ; Credits have been extended
89 0148 C5 90 OB8D 2008 MOVW UCBSB_LT_RCRED(R5), (R9)+ ; Load credits
0148 C5 94 OB92 2009 CLRB UCBSB_LT_RCRED(R5) ; Don't send them again!
      OB96 2010
5A 52 D1 OB96 2011 110$: CMPL R2, R10 ; Do we have more than we can handle?
      OB99 2012 BLEQU 120$ ; Br if no, its just fine
52 5A D0 OB9B 2013 MOVL R10, R2 ; Use just what we can handle
5A 52 C2 OB9E 2014 120$: SUBL R2, R10 ; Adjust what we have left
      OBA1 2015 ; ***
      OBA1 2016 ; Reading the code I noticed the following line(s) would not work if
      OBA1 2017 ; the startio entry were to discover that it were balanced after
      OBA1 2018 ; previous calls stored data in the UCB extensions because
      OBA1 2019 ; we were not balanced. I guess that never happens, because R2 would
      OBA1 2020 ; not reflect the UCB extension data area. BEM
      OBA1 2021 ; ***
011C C5 52 C0 OBA1 2022 ADDL2 R2,UCBSL_TT_OUTADR(R5) ; Update UCB state
0120 C5 52 A2 OBA6 2023 SUBW2 R2,UCBSW_TT_OUTLEN(R5) ; Update UCB state
      DD 2024 PUSHL R5 ; Save the UCB address
69 63 52 28 OBAD 2025 MOVCL3 R2,(R3),(R9) ; Copy the string
      55 B EDO OBB1 2026 POPL R5 ; Restore the UCB address
59 53 D0 OBB4 2027 MOVL R3,R9 ; Set address of next available byte
      5A D5 OBB7 2028 TSTL R10 ; Do we have any left?
      28 13 OBB9 2029 BEQL 155$ ; Br if no, all done.
010C D5 16 OBBB 2030 JSB @UCBSL_TT_GETNXT(R5) ; Else, call back class driver for more
      19 13 OBBF 2031 BEQL 150$ ; Br if no more to be had
      OC 14 OBC1 2032 BGTR 140$ ; Br is single character
      OBC3 2033
      OBC3 2034 ; Found a burst of data
      OBC3 2035
53 011C C5 D0 OBC3 2036 MOVL UCBSL_TT_OUTADR(R5),R3 ; Get output buffer address
52 0120 C5 3C OBC8 2037 MOVZWL UCBSW_TT_OUTLEN(R5),R2 ; and charcter count
      C7 11 OBCD 2038 BRB 110$ ; Else buffer address
      OBCF 2039
      OBCF 2040 140$:
      OBCF 2041 ; Found a single character
      OBCF 2042
      OBCF 2043
52 01 D0 OBCF 2043 MOVL #1, R2 ; Else, just one character returned
69 53 90 OBD2 2044 MOVW R3, (R9) ; and fake many chars returned
53 59 D0 OBD5 2045 MOVL R9, R3 ; Copy buffer address

```



```

BC 11 OBD8 2046 BRB 110$ ; and continue
      OBDA 2047
64 A5 02 AA OBDA 2048 150$: BICW #UCBSM_INT,UCBSL_STS(R5); Allow more start io calls (checked)
      01 8A OBDE 2049 BICB #UCBSM_LT_DATA,- ; no output data available
      0142 C5 OBE0 2050 UCBSM_LT_DATAW(R5)
52 59 56 C3 OBE3 2051 155$: SUBL3 R6, R9, R2 ; Make the length of the data to
00000048 BF C2 OBE7 2052 SUBL #XMT_T_DATA, R2 ; transmit
      52 OBE8 2053
      59 D6 OBEE 2053 INCL R9 ; Round end address of data
      01 CA OBF0 2054 BICL #1,R9
1C A6 59 D0 OBF3 2055 MOVL R9,CXBSL_T_ENDADR(R6) ; Store it in the CXB
      52 0 C2 OBF7 2056 SUBL #<<XMT_T_MDATA- - ; Make total characters in slot
      OBF8 2057 XMT_T_DATA> + - ; minus those already accounted for
      OBF9 2058 SLT_T_DATA>, R2 ; beyond the circuit and slot headers
52 A6 52 90 OBF9 2059 MOVB R2,<SLT_B_COUNT+- ; Store that count away.
      OBF8 2060 XMT_T_MDATA>(R6) ; In the first slot maxc
      OBF8 2061
      OBF8 2062 .IF DEFINED LT_XMT_CHECK
      OBF8 2063 ;
      OBF8 2064 ; debug code
      OBF8 2065 ;
      OBF8 2066 MOVQ CSB_T_CIRCHDR(R8),- ; Move the circ header
      OBF8 2067 XMT_T_DATA(R6) ; to look at real msg
      OBF8 2068 MOVZBL XMT_B_NUM_SLOTS(R6), R1 ; number of valid slots
      OBF8 2069 BNEQ 160$ ; We have some, skip check
      OBF8 2070 ; No slots - we must only have a circuit header!
      OBF8 2071 MOVAB XMT_T_MDATA(R6), R0 ; Past circuit header
      OBF8 2072 CMPL R0,R9 ; Only have a circuit header?
      OBF8 2073 BEQL 160$ ; Br if yes, okay
      OBF8 2074 DEBUG ; Not correct
      OBF8 2075
      OBF8 2076 .ENDC ;; DEFINED LT_XMT_CHECK
      OBF8 2077
54 58 D0 OBF8 2078 160$: MOVL R8, R4 ; CSB address for call
      OE9C 30 OC01 2079 BSBW LT$XMITONE ; Transmit the buffer
0760 BF BA OC04 2080 POPR #*M<R5,R6,R8,R9,R10>
      05 OC08 2081 RSB
      OC09 2082
      OC09 2083 168$: ;
      OC09 2084 ; Discard output data, no session active !
      OC09 2085 ;
004D 30 OC09 2086 BSBW LT$FLUSH_DATA ; Flush the output data
      OC0C 2087
      OC0C 2088 170$: ;
      OC0C 2089 ; Allow more calls to startio
      OC0C 2090 ;
64 A5 02 AA OC0C 2091 BICW #UCBSM_INT, - ; Allow more calls here
      0760 BF BA OC10 2092 UCBSL_STS(R5)
      05 OC14 2093 180$: POPR #*M<R5,R6,R8,R9,R10>
      OC15 2094 RSB
      OC15 2095
      OC15 2096 ;
      OC15 2097 ;
      OC15 2098 ;
      OC15 2099 ;
      OC15 2100 ;
      OC15 2101 ;
      OC15 2101

```

Not balanced, so buffer data if we can.
R2 = count
R3 -> data

```

54 014C C5 9E 0C15 2102 190$: MOVAB UCBSB_LT_CBUF(R5), R4 : Address of character buffer
    54 53 D1 0C1A 2103      CMPL  R3, R4      : Is this the character buffer?
    28 13 0C1D 2104      BEQL  210$       : Br if yes, check if its full
50 014B C5 9A 0C1F 2105      MOVZBL UCBSB_LT_CURC(R5), R0 : Calculate amount of room
    51 D4 0C24 2106      CLRL  R1         : left in the output buffer
014A C5 50 83 0C26 2107      SUBB3 R0, -      :
    51      0C2B      :
    52 51 D1 0C2C 2108      UCBSB_LT_MAXC(R5), R1 :
    DF 15 0C2F 2109      CMPL  R1, R2     : Do we have room?
0120 C5 B4 0C31 2110      BLEQ  180$      : Br if no, so shut output off
014B C5 52 80 0C35 2111      CLRW  UCBSW TT_OUTLEN(R5) : Else, forget about the data here.
    51 6440 9E 0C3A 2113      ADDB2 R2, UCBSB_LT_CURC(R5) : Calculate the number of chars
    55 DD 0C3E 2114 200$: MOVAB (R4)(R0), -R1 : and get address to put them
61 63 52 28 0C40 2115      PUSHL R5        : Save the UCB address
    55 8ED0 0C44 2116      MOVC3 R2, (R3), (R1) : Copy the string
    0C47 2117      POPL  R5        : Restore the UCB address
    014B C5 91 0C47 2119 210$: CMPB  UCBSB_LT_CURC(R5), - : Is buffer full?
    014A C5      0C4B      :
    C0 18 0C4E 2120      UCBSB_LT_MAXC(R5)
010C D5 16 0C50 2121      BGEQ  180$      : Br if yes, shut off output
    B6 13 0C54 2122      JSB  @UCBSL TT_GETNXT(R5) : Else, get more data
    FE9D 31 0C56 2123      BEQL  170$      : No more data, so just return
    0C59 2124      BRW  58        : Else, process data
    0C59 2125

```

```

0C59 2127 .SBTTL LTSFLUSH_DATA - Flush buffered output data and drain TT buffer
0C59 2128 :++
0C59 2129 : LTSFLUSH_DATA - Flush buffered output data and drain class driver buffer
0C59 2130 :
0C59 2131 : Functional description:
0C59 2132 :
0C59 2133 : Flush buffered output data and drain class driver output buffer.
0C59 2134 :
0C59 2135 : Inputs:
0C59 2136 :     R5 = UCB address
0C59 2137 :
0C59 2138 : Outputs:
0C59 2139 :     R3 is destroyed.
0C59 2140 :--
0C59 2141 :
0C59 2142 LTSFLUSH_DATA:
010C DS 16 0C59 2143 20$: JSB @U B$LT_GETNXT(R5) ; Call back for more to class driver
02 13 0C5D 2144 BEQL 90$ ; Br if no more to be had
FB 11 0C5F 2145 BRB 20$ ; Else, drain all output data
0120 CS B4 0C61 2146 90$: CLRW UCBSW_TT_OUTLEN(R5) ; Clear CLASS driver output data
014B CS 94 0C65 2147 CLRB UCBSB_LT_CURC(R5) ; Dump UCB output buffering
05 0C69 2148 RSB

```

LT
V0
:
J
-1

```

OC6A 2150      .SBTTL LTSABORT - Abort (flush) buffered output data
OC6A 2151      :++
OC6A 2152      : LTSABORT - Abort buffered output data
OC6A 2153      :
OC6A 2154      : Functional description:
OC6A 2155      :
OC6A 2156      : Abort (flush) buffered output data
OC6A 2157      :
OC6A 2158      : Inputs:
OC6A 2159      :   R5 = UCB address
OC6A 2160      :
OC6A 2161      : Outputs:
OC6A 2162      :
OC6A 2163      : --
OC6A 2164      :
OC6A 2165      : LTSABORT:
014B C5 94 OC6A 2166      CLRB UCBSB_LT_CURC(R5)      : Dump UCB buffering
0120 C5 B4 OC6E 2167      CLRW UCBSW_TT_OUTLEN(R5)    : Dump pending class driver data,
OC72 2168      : (might have dissappeared)
0142 C5 04 88 OC72 2169      BISB #UCBSM_LT_ABORT,-      : Abort requested
OC74 2170      RSB UCBSB_LT_DATAW(R5)
OC77 2171      :
OC78 2172      :
OC78 2173      :
OC78 2174      :
OC78 2175      :
OC78 2176      : .SBTTL LTSFLOW_CHANGE - Change flow control
OC78 2177      :
OC78 2178      : ++
OC78 2179      : LTSFLOW_CHANGE
OC78 2180      :
OC78 2181      : Function: Inform terminal server of change in host flow control.
OC78 2182      : The terminal server will mimic host policy.
OC78 2183      :
OC78 2184      : Inputs:
OC78 2185      :   R5 = UCB address
OC78 2186      :
OC78 2187      : Outputs:
OC78 2188      :
OC78 2189      : --
OC78 2190      :
OC78 2191      : LTSFLOW_CHANGE:
0142 C5 02 88 OC78 2192      BISB #UCBSM_LT_FLOW,-      : Flow change
OC7A 2193      RSB UCBSB_LT_DATAW(R5)
OC7D 2194      :
OC7E 2195      :

```

```

OC7E 2197 .SBTTL LT$XON - Resume input stream into class driver
OC7E 2198 :++
OC7E 2199 : LT$XON - Resume input stream into class driver
OC7E 2200 :
OC7E 2201 : Inputs:
OC7E 2202 : R3 = XON character
OC7E 2203 : R5 = UCB address
OC7E 2204 :
OC7E 2205 : Outputs:
OC7E 2206 :
OC7E 2207 : R3 is destroyed
OC7E 2208 :--
OC7E 2209 LT$XON:
1803 8F BB OC7E 2210 PUSHR #*M<R0,R1,R11,AP> : Save registers
OC82 2211 CLRBIT #UCBSV_LT_OVERFLOW,- : Clear overflow indicator
OC82 2212 UCBSB_LT_LATSTS(R5)
50 013C C5 D0 OC88 2213 MOVL UCBSL_LT_INPBUF(R5),R0 : Get address of input buffer
OC8D 2214 BEQL 70$ : Br if none, nothing to do
0147 C5 96 OC8F 2215 INCB UCBSB_LT_XCRED(R5) : Okay to return credit now
OC93 2216
OC93 2217 .IF DEFINED LT_CRED_CHECK
OC93 2218 BLEQ 10$ : Br if okay
OC93 2219 DEBUG : Else, we went too far
OC93 2220 10$:
OC93 2221 .ENDC ;; DEFINED LT_CRED_CHECK
OC93 2222
OC93 2223 ASSUME UCBSB_LT_LATSTS EQ JCBSB_LT_DATAW+1
OC93 2224
1001 8F AB OC93 2225 BISW #<UCBSM_LT_INPUTAB>!UCBSM_LT_DATA,- : Data potentially available
0142 C5 OC97 2226 UCBSB_LT_DATAW(R5) : and INPUT stream in enabled
64 A5 02 88 OC9A 2227 BISB #UCBSM_INT,UCBSL_STS(R5); Force no data to be returned
5C 02 A0 3C OC9E 2228 MOVZWL INB_W_BCNT(R0),AP : Get count of bytes remaining
5B 60 3C OCA2 2229 MOVZWL INB_W_BOFF(R0),R11 : Get offset to start of input data
5B 50 C0 OCA5 2230 ADDL R0,R11 : Calculate address of input data
OCAB 2231
OCAB 2232 : Feed remaining data into class driver
OCAB 2233
53 8B 9A OCA8 2234 30$: MOVZBL (R11)+,R3 : Get next character
0110 D5 16 OCA8 2235 JSB @UCBSL_TT_PUTNXT(R5) : Pass character to class driver
0143 C5 E0 OCAF 2236 BBS #UCBSV_LT_OVERFLOW,- : Br if overflow, wait for next XON
OF F0 5C F5 OCBS 2237 UCBSB_LT_LATSTS(R5),50$
50 013C C5 D0 OCBB 2238 SOBGTR AP,30$ : Loop if more
OF 19 30 OCBD 2239 MOVL UCBSL_LT_INPBUF(R5),R0 : Get input buffer address
013C C5 D4 CC00 2240 BSBW LT$DEALPOOL : Deallocate the input buffer
0143 C5 8A OCC4 2241 50$: CLRL UCBSL_LT_INPBUF(R5) : Zero input buffer pointer
1803 8F BA OCC6 2242 50$: BICB #UCBSM_LT_INPUT,- : Input stream is now disabled
OCCE 2243 UCBSB_LT_LATSTS(R5)
OCCE 2244 70$: POPR #*M<R0,R1,R11,AP> : Restore registers
OCCE 2245 RSB
OCCE 2246
OCCE 2247 .SBTTL LT$XOFF - Stop input stream into class driver
OCCE 2248
OCCE 2249 :++
OCCE 2250 : LT$XOFF - Stop input stream into class driver
OCCE 2251 :
OCCE 2252 : Inputs:
OCCE 2253 : R3 = XOFF character (^S or ^G)

```

```

OCCE 2254 :      R5 = UCB address
OCCE 2255 :      AP = count of bytes remaining
OCCE 2256 :      R11 = address of input data
OCCE 2257 :
OCCE 2258 :      Implicit inputs:
OCCE 2259 :
OCCE 2260 :      UCBSV_LT_INPUT bit must be on UCBSB_LT_LATSTS status byte.
OCCE 2261 :
OCCE 2262 :      Outputs:
OCCE 2263 :      R3 is destroyed
OCCE 2264 :
OCCE 2265 :      --
OCCE 2266 :      LT$XOFF:
07 37 BB OCCE 2267      PUSHR   #*M<R0,R1,R2,R4,R5>      : Save registers
   53 91 OCDO 2268      CMPB    R3,#ITY_C_BELL      : Is this a control G?
   44 13 OCDS 2269      BEQL   90$                : Br if yes, do nothing
3E 0143 C5 E1 OCDS 2270      BBC    #UCBSV_LT_INPUT,-      : Br if input stream not enabled
01 01 5C D1 OCDB 2271      UCBSB_LT_LATSTS(R5),90$
   39 15 OCDE 2272      Cmpl   AP,#1                : Any data to buffer? (counting current)
   5C D7 OCEO 2273      BLEQ  90$                : Br if no, skip it
OCCE 2274      DECL   AP                  : Else, account for character just passed
OCCE 2275      SETBIT #UCBSV_LT_OVERFLOW,-      : Set overflow indicator
OCCE 2276      UCBSB_LT_LATSTS(R5)
52 013C C5 D0 OCEB 2277      MOVL  UCBSL_LT_INPBUF(R5),R2 : Get input buffer
   16 12 OCED 2278      BNEQ  30$                : Br if one there
51 5C 0C C1 OCEF 2279      ADDL3 #INB_C_LENGTH,AP,R1     : Else, calculate length of buffer
00000000 GF 16 OCF3 2280      JSB   G*EXESALONONPAGED     : Allocate a buffer
   1D 50 E9 OCF9 2281      BLBC  R0,90$              : Br if no buffer
013L C5 52 D0 OCFC 2282      MOVL  R2,UCBSL_LT_INPBUF(R5) : Save address of input buffer
08 A2 51 B0 OD01 2283      MOVW  R1,INB_W_SIZE(R2)      : Set size of input buffer
OCCE 2284      30$: ASSUME  INB_B_SPARE EQ INB_B_TYPE+1
OCCE 2285      MOVZBW #DYRSC_BUFID,INB_B_TYPE(R2) : Set structure type
0A A2 13 98 OD05 2286      MOVW  AP,INB_W_BCNT(R2)      : Set size of remaining string
02 A2 5C B0 OD09 2287      MOVW  #INB_C_LENGTH,INB_W_BOFF(R2) : Set offset to start of data
62 0C B0 OD10 2288      :
OCCE 2289      : Don't return credit... wait for XON
OCCE 2290      :
0147 C5 97 JD10 2291      DECB  UCBSB_LT_XCRED(R5)      : Don't return credit
OCCE 2292      :
OCCE 2293      .IF DEFINED LT_CRED_CHECK
OCCE 2294      CMPB  UCBSB_LT_XCRED(R5),- : Did we goof?
OCCE 2295      #-LATSC_MAX_RCRED
OCCE 2296      BGEQU 10$                : Br if okay
OCCE 2297      DEBUG : Else, we went too far
OCCE 2298      10$:
OCCE 2299      .ENDC  ;; DEFINED LT_CRED_CHECK
68 5C 28 OD14 2300      :
OC  A2 37 BA OD14 2301      MOV(C3 AP,(R11),INB_C_LENGTH(R2) : Copy the data
OCCE 2302      90$: POPR   #*M<R0,R1,R2,R4,R5>      : Restore registers
OCCE 2303      RSB   : Return to caller
OCCE 2304

```

```

OD1C 2306      .SBTTL LTSFFI_RCV_MSG - FAST Interface Receive Complete routine
OD1C 2307      :++
OD1C 2308      : LTSFFI_RCV_MSG - FAST Interface Receive Complete routine
OD1C 2309      :
OD1C 2310      : Functional description:
OD1C 2311      :
OD1C 2312      : Each time a frame is received, the data is passed to the class driver
OD1C 2313      : for each terminal represented within the frame. Echoed data is copied
OD1C 2314      : into a frame to be transmitted back to the concentrator, and finally
OD1C 2315      : the transmit frame is queued for transmission after appropriate
OD1C 2316      : virtual circuit maintenance functions are performed.
OD1C 2317      :
OD1C 2318      : Inputs:
OD1C 2319      :
OD1C 2320      :     R3 = CXB address
OD1C 2321      :     R4 = FFI address
OD1C 2322      :
OD1C 2323      :     IPL = SYNCH
OD1C 2324      :
OD1C 2325      : Outputs:
OD1C 2326      :
OD1C 2327      :     R1-R5 are preserved.
OD1C 2328      :
OD1C 2329      :     The interrupt is dismissed when the received frame
OD1C 2330      :     has been completely processed. The CXB is always
OD1C 2331      :     returned to the datalink driver.
OD1C 2332      :
OD1C 2333      :--
OD1C 2334      :
OD1C 2335      : ASSUME LATSC_IPL EQ IPLS_SYNCH
OD1C 2336      : LTSFFI_RCV_MSG:
OD1C 2337      : PUSHR #*M<R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,AP,FP> ; Save registers
OD20 2338      :
OD20 2339      : MOVZWL CXBSW_BOFF(R3),R6 ; Get offset to start of data
56 18 A3 3C OD20 2339      :
56 53 C0 OD24 2340      : ADDL R3,R6 ; Get pointer to start of data
OD27 2341      :
OD27 2342      : .IF DEFINED LT_HISTORY
OD27 2343      :
OD27 2344      : Save history of receives
OD27 2345      :
50 66 9E OD27 2346      : MOVAB RCV_B_FLAG(R6), R0 ; Address is circuit header
50 F9FA 30 OD2A 2347      : BSBW LTSHISTORY ; Store the data
OD2D 2348      : .ENDC ;; DEFINED LT_HISTORY
OD2D 2349      :
OD2D 2350      : INC_CTR GHBSL_RCOUNT+GMB_AREA . Count frame
OD37 2351      :
OD37 2352      : Validate received frame
OD37 2353      :
50 02 A6 3C OD37 2354      : MOVZWL RCV_W_DSTID(R6),R0 ; Get remote vci
76 12 OD38 2355      : BNEQ 120$ ; Got index, okay
91 OD3E 2356      : CMPB #<MTYP_C_START@FLAG_V_MTYPE>!- ; Is this a start message?
OD3E 2357      : FLAG_M_MASTER -
66 06 OD3E 2358      : RCV_B_FLAG(R6)
29 12 OD40 2359      : BNEQ 91$ ; Br if no, signal protocol error
OD42 2360      :
OD42 2361      : START message received on an inactive circuit, try to start up a
OD42 2362      : new circuit.

```

```

F3EE CF D5 OD42 2363 :
      4F 13 OD42 2364 : TSTL GHBSL_UCB+GMB_AREA : Data link UCB still here?
      04 A6 B5 OD46 2365 : BEQL 99$ : No, then don't accept circuit
      32 13 OD48 2366 : TSTM RCV_W_SRCID(R6) : Is the source ID zero?
      06 A6 95 OD48 2367 : BEQL 96$ : Br if yes, protocol error
      32 12 OD4D 2368 : TSTB RCV_B_SEQ(R6) : Is this the correct sequenced msg?
      057B 30 OD50 2369 : BNEQ 97$ : Br if not, protocol error
      10 50 EB OD52 2370 : BSBW LTSNEW_CIRCUIT : Initialize and/or allocate CSB
      50 D5 OD58 2371 : BLBS R0, 19$ : Go ahead, CSB address in R8
      40 12 OD58 2372 : : startup any transmits waiting
      50 D5 OD58 2373 : TSTL R0 : Are we ignoring a duplicate start msg?
      40 12 OD5A 2374 : BNEQ 100$ : Br if yes, just retransmit start msg
      2F 11 OD5C 2375 : INC_CTR GHBSL_RESOURCE+GMB_AREA : Else, no CSB to use
      2F 11 OD66 2376 : BRB 99$ : So just ignore it
      00B5 31 OD68 2377 :
      00B5 31 OD68 2378 19$: BRW 190$ : Long branch
      5D 00 9A OD68 2379 :
      5D 17 11 OD6E 2380 91$: MOVZBL #GHBSV_START,FP : Invalid START message received
      5D 02 9A OD70 2381 : BRB 98$ : continue
      5D 12 11 OD73 2382 93$: MOVZBL #GHBSV_CSBRANGE,FP : Bad range on CSB index
      5D 03 9A OD75 2383 : BRB 98$ : continue
      5D 0D 11 OD78 2384 94$: MOVZBL #GHBSV_CSBINVALID,FP : Invalid CSB index
      5D 04 9A OD7A 2385 : BRB 98$ : continue
      5D 08 11 OD7D 2386 95$: MOVZBL #GHBSV_CSBSTALE,FP : Stale CSB index
      5D 06 9A OD7F 2387 : BRB 98$ : continue
      5D 03 11 OD82 2388 96$: MOVZBL #GHBSV_INVALIDREMID,FP : Invalid remote circuit id
      5D 0A 9A OD84 2389 : BRB 98$ : continue
      5D 0A 9A OD84 2390 97$: MOVZBL #GHBSV_INVALIDSEQ,FP : Invalid sequence number
      5D 0A 9A OD84 2391 : BRB 98$ : continue
      5D 0A 9A OD84 2392 98$: SETBIT FP,GHBSL_PROTOMASK+GMB_AREA : Set bit mask
      5D 0A 9A OD84 2393 : INC_CTR GHBSL_PROTOCOL+GMB_AREA : Increment counter
      3FFE 8F BA OD97 2394 99$: POPR #R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,AP,FP> : Restore registers
      3FFE 8F 05 OD98 2395 : RSB : Return to caller
      3FFE 8F 05 OD98 2396 :
      3FFE 8F 05 OD98 2397 :
      3FFE 8F 05 OD98 2398 :
      3FFE 8F 05 OD98 2399 :
      3FFE 8F 05 OD98 2400 100$: INC_CTR GHBSL_DUPLMSG+GMB_AREA : We received a duplicate message
      3FFE 8F 05 OD98 2401 : INC_CTR GHBSL_RETRANS+GMB_AREA : We are retransmitting a message
      3FFE 8F 05 OD98 2402 : BRW LTSREPLY_REXMIT : This data is stale, resend buffers
      3FFE 8F 05 OD98 2403 :
      3FFE 8F 05 OD98 2404 :
      3FFE 8F 05 OD98 2405 :
      3FFE 8F 05 OD98 2406 :
      3FFE 8F 05 OD98 2407 :
      3FFE 8F 05 OD98 2408 :
      3FFE 8F 05 OD98 2409 :
      3FFE 8F 05 OD98 2410 :
      3FFE 8F 05 OD98 2411 :
      3FFE 8F 05 OD98 2412 :
      3FFE 8F 05 OD98 2413 :
      3FFE 8F 05 OD98 2414 :
      3FFE 8F 05 OD98 2415 :
      3FFE 8F 05 OD98 2416 :
      3FFE 8F 05 OD98 2417 :
      3FFE 8F 05 OD98 2418 :
      3FFE 8F 05 OD98 2419 :

      Non-zero circuit ID !
      - circuit must be active already

      Inputs: R0 = CSB index
              R3 = CSB address
              RA = Pointer to start of data

      51 50 9A OD83 2413 120$: MOVZBL R0,R1 : Get CSB index
      20 51 D1 OD86 2414 : CMPL R1,#LATSC_MAX_CSBS : Range check
      58 F274 CF41 B5 14 OD89 2415 : BGTR 93$ : Br if bad range, protocol error
      60 A8 50 B1 13 OD8B 2416 : MOVL GMB_L_CSBTABLE-4[R1],R8 : Get CSB address
      60 A8 50 B1 13 OD8B 2417 : BEQL 94$ : Br if invalid CSB, protocol error
      60 A8 50 B1 12 OD83 2418 : CMPW R0,CSB_W_LOCID(R8) : Valid reference?
      60 A8 50 B1 12 OD83 2419 : BNEQ 95$ : Br if no, Stale reference to CSB

```



```

      ODC9 2420      INC_CTR CSB_Z_LCB+LCB_L_MSG_RCV(R6) ; Count one more rcvd msg
      ODD1 2421      :
      ODD1 2422      : Check ACK number and complete any transmits acknowledged
      ODD1 2423      :
53 07 AC 9A ODD1 2424 130$: MOVZBL RCV_B_ACK(R6), R3      ; Get the ack from the message
65 AB 53 90 ODD5 2425      MOVB R3, CSB_B_RACK(R6)      ;
53 62 AB 82 ODD9 2426      SUBB CSB_B_XSEQ(R6), R3      ; Set up the normalized ack
52 3C AB 7E ODDD 2427      MOVAQ CSB_Q_XWAITQ(R6), R2      ; Look through the wait queue for it
      51 52 00 ODE1 2428      MOVL R2, RT      ;
      ODE4 2429      :
      51 61 00 ODE4 2430 140$: MOVL (R1), R1      ; Next item in queue
      52 51 01 ODE7 2431 150$: CMPL R1, R2      ; End of queue
4E A1 62 AB 13 ODEA 2432      BEQL 190$      ; Done with that search
      50 83 ODEC 2433      SUBB3 CSB_B_XSEQ(R6), -      ; Normalized the seq of message
      ODF1 2434      XMT_B_SEQ(R1), -      ;
      ODF2 2435      RO      ;
      53 50 91 ODF2 2436      CMPB RO, R3      ; If the seq is greater than ack,
      ED 14 ODF5 2437      BGTR 140$      ; wait on another ack
      54 61 00 ODF7 2438      MOVL (R1), R4      ; Save link to next item
      51 61 0F ODFA 2439      REMQUE (R1), R1      ; Remove acked message from wait queue
38 BB 61 0E ODFD 2440      INSQUE (R1), -      ; And insert on tail of transmit queue
      OE01 2441      @CSB_Q_XBUFQ+4(R6)      ;
      59 AB 97 OE01 2442      DECB CSB_B_XMTCNT(R6)      ; One less to transmit
      51 54 00 OE04 2443      MOVL R4, RT      ; Restore link
      DE 11 OE07 2444      BRB 150$      ;
      OE09 2445 170$:      :
      OE09 2446      : Invalid sequence number received
      OE09 2447      :
      OE09 2448      :
      89 11 OE11 2449 180$: INC_CTR CSB_Z_LCB+LCB_W_SEQ_ERR(R6), W ; Bad sequence number
      OE12 2450      BRB 100$      ; Also count as duplicate
      OE13 2451      :
      OE13 2452      : No transmit buffer available
      OE13 2453      :
      FF77 31 OE1D 2454 190$: INC_CTR GHBSL_NOXBFR+GMB_AREA ; Not transmit buffer available
      OE20 2455      BRW 99$      ; And exit
      OE20 2456      :
      OE20 2457      : Make sure that we have a transmit buffer to respond with
      52 34 AB 7E OE20 2458 190$: MOVAG CSB_Q_XBUFQ(R6), R2      ; Look to see if we have at least one
      62 52 01 OE24 2459      CMPL R2, (R2)      ; transmit buffer
      EA 13 OE27 2460      BEQL 180$      ; Br if None, just retransmit and wait
      OE29 2461      :
      66 AB 80 OE29 2462      MOVW CSB_W_TIMRESET(R6), -      ; Reset timer for timeout of circuit
      50 AB OE2C 2463      CSB_W_TIMEOUT(R6)      ; Since we got a valid message.
64 AB 06 A6 91 OE2E 2464      CMPB RCV_B_SEQ(R6), CSB_B_RSEQ(R6) ; Right sequence number?
      D4 12 OE33 2465      BNEQ 170$      ; Br if no, retransmit all messages
      OE35 2466      :
      OE35 2467      :
      OE35 2468      :
      OE35 2469      : Dispatch on message type
      OE35 2470      :
      OE35 2471      : Inputs:
      OE35 2472      :
      OE35 2473      : RB = CSB address
      OE35 2474      : R6 = (XB) address
      OE35 2475      :

```



```

OE4C 2491 .SBTTL PROCESS RECEIVED MESSAGE
OE4C 2492 :++
OE4C 2493 :
OE4C 2494 : Other guy is halted or all screwed up
OE4C 2495 :
OE4C 2496 :--
OE4C 2497 :
OE4C 2498 LT$FORCE_HALT: ; receive halt msg
OE4C 2499 ; invalid circuit state in msg
OE4C 2500 ; invalid circuit state in CSB
OE4C 2501 ; invalid local vci
OE4C 2502 SETBIT #GHBSV HALT,-
OE4C 2503 GHBSL_PROTOMASK+GHB AREA ; Circuit forced to halted
OE52 2504 INC_CTR GHBSL_PROTOCOL+GHB_AREA ; Increment protocol error counter
OE5C 2505
08CB 30 OE5C 2506 LT$STATE_HALT:
0403 31 OE5C 2507 BSBW LT$CIRCDEAD ; Circuit is dead now
OE5F 2508 BRW LT$REPLY_REXMIT ; Transmit halt message
OE62 2509
OE62 2510 :++
OE62 2511 :
OE62 2512 : Other system is halted
OE62 2513 :
OE62 2514 :--
OE62 2515 :
08C5 30 OE62 2516 LT$RCV_HALT_MSG:
03F8 31 OE62 2517 BSBW LT$CIRCDEAD ; Circuit is dead now
OE65 2518 BRW LT$REPLY_DONE ; All done
OE68 2519
OE68 2520 :++
OE68 2521 :
OE68 2522 : Other guy is starting up
OE68 2523 :
OE68 2524 :--
OE68 2525 :
OE68 2526 LT$RCV_START_MSG:
OE68 2527
OE68 2528 $DISPATCH CSB_B_STATE(R8),TYPE=B,- ; Dispatch on our CSB state
OE68 2529 <- ; state ; action
OE68 2530 <CSB_C_STATE_HALT LT$STATE_START>,- ; Enter the START state
OE68 2531 <CSB_C_STATE_START LT$REPLY_MSG>,- ; Re-send START message
OE68 2532 <CSB_C_STATE_RUN LT$STATE_HALT>,- ; Enter the HALTED state
OE68 2533 >
07 11 OE73 2534
OE73 2535 BRB LT$FORCE_HALT . What kind of kinky state is this ?
OE75 2536
OE75 2537 :++
OE75 2538 :
OE75 2539 : Other guy is running
OE75 2540 :
OE75 2541 :--
OE75 2542 :
OE75 2543 LT$RCV_RUN_MSG:
OE75 2544
OE75 2545 $DISPATCH CSB_B_STATE(R8),TYPE=B,- ; Dispatch on our CSB state
OE75 2546 <- ; state ; action
OE75 2547 <CSB_C_STATE_HALT LT$REPLY_MSG>,- ; Send STOP message

```

LTDRIVER
V04-002

- Local Area Terminal Port Driver C 3
PROCESS RECEIVED MESSAGE

8-JAN-1985 17:22:25 VAX/VMS Macro V04-00 Page 61
5-SEP-1984 11:40:22 [LAT.BUGSRC]LTDRIVER.MAR;1 (41)

```

      OE75 2548      <CSB_C_STATE_START      LTSSTATE_ENTER_RUN>,- ; Enter RUN state
      OE75 2549      <CSB_C_STATE_RUN      LTSSTATE_RUN>,- ; Stay in RUN state
      OE75 2550      >
      OE80 2551      BRB
CA 11 OE80 2552      LTSFORCE_HALT      ; What kind of kinky state is this ?
```

LT
VO

```

.OE82 2554 .SBTTL CIRCUIT STATE TRANSITIONS
.OE82 2555 :++
.OE82 2556 :
.OE82 2557 : We can go only from halted to starting
.OE82 2558 :
.OE82 2559 :--
.OE82 2560 :
.OE82 2561 LT$STATE START:
01 90 .OE82 2562 MOVB #CSB_C_STATE_START,- ;
0B A8 .OE84 2563 CSB_B_STATE(R8) ; Now we are starting
04 90 .OE86 2564 MOVB #MTYP_C_START@FLAG_V_MTYPE,- ; Set the start message type
5C A8 .OE88 2565 CSB_B_FLAG(R8) ;
04 A6 B0 .OE8A 2566 MOVW RCV_W_SRCID(R6),- ;
5E A8 .OE8D 2567 CSB_W_REMID(R8) ; Save his circuit id
59 34 A8 D0 .OE8F 2568 MOVL CSB_Q_XBUFQ(R8),R9 ; Format a start message
50 F569 CF 9E .OE93 2569 MOVAB GHBT_STRT_MSG,R0 ; Get start address of template data
60 F562 CF 28 .OE98 2570 MOVCS GHBT_W_STRT_LEN,(R0),- ; Copy the start message
50 A9 .OE9D 2571 XMT_T_MDATA(R9) ;
1C A9 53 D0 .OE9F 2572 MOVL R3,CXBSL_T_ENDADR(R9) ; Store end address of data
03B4 31 .OEA3 2573 BRW LT$REPLY_ALT ; Go send the reply message
.OEA6 2574 :
.OEA6 2575 :++
.OEA6 2576 :
.OEA6 2577 : We transit to the running state only from the starting state
.OEA6 2578 :
.OEA6 2579 :--
.OEA6 2580 :
.OEA6 2581 LT$STATE ENTER_RUN: ; Enter the RUNNING state
02 90 .OEA6 2582 MOVB #CSB_C_STATE_RUN,- ; Now we are running
0B A8 .OEA8 2583 CSB_B_STATE(R8) ;
00 90 .OEA A 2584 MOVB #MTYP_C_RUN@FLAG_V_MTYPE,- ; Set the run message type
5C A8 .OEA C 2585 CSB_B_FLAG(R8) ;
.OEAE 2586 :
.OEAE 2587 :++
.OEAE 2588 :
.OEAE 2589 : Update state from validated frame
.OEAE 2590 :
.OEAE 2591 :--
.OEAE 2592 :
.OEAE 2593 LT$STATE RUN:
04 A6 B1 .OEAE 2594 CMPW RCV_W_SRCID(R6),- ; Referencing a valid circuit?
5E A8 .OEB1 2595 CSB_W_REMID(R8) ;
59 97 12 .OEB3 2596 BNEQ LT$FORCE_HALT ; Br if not, force a halt
34 A8 D0 .OEB5 2597 MOVL CSB_Q_XBUFQ(R8),R9 ; Get transmit buffer address
.OEB9 2598 :
.OEB9 2599 : Set up for the receive loop
.OEB9 2600 :
.OEB9 2601 CLRBIT #FLAG_V_RRF,- ; No response requested until
.OEB9 2602 CSB_B_FLAG(R8) ; a credit is used
5D A8 94 .OEBE 2603 CLRW CSB_B_NUM_SLOTS(R8) ; Assume no output data (balanced)
52 A8 B4 .OEC1 2604 CLRW CSB_W_XMTTMO(R8) ; Circuit state is no longer timed
57 08 A6 9E .OEC4 2605 MOVAB RCV_T_MDATA(R6),R7 ; Receive buffer first slot address
59 50 A9 9E .OEC8 2606 MOVAB XMT_T_MDATA(R9),R9 ; Transmit buffer first slot address
5A 6A A8 3C .OEC C 2607 MOVZWL CSB_W_MAX_MSGSIZ(R8),R10 ; Total chars available to slots
5D 01 A6 9A .OED0 2608 MOVZBL RCV_B_NUM_SLOTS(R6),FP ; Get slot loop count
03 12 .OED4 2609 BNEQ SLOT_COOP_IN ; Got some input data
016B 31 .OED6 2610 BRW SLOT_LOOP_OUT ; Go process output buffer

```

LTDRIVER
V04-002

- Local Area Terminal Port Driver^{E 3}
CIRCUIT STATE TRANSITIONS

0ED9 2611

8-JAN-1985 17:22:25 VAX/VMS Macro V04-00
5-SEP-1984 11:40:22 [LAT.BUGSRC]LTDRIVER.MAR;1

Page 63
(42)

LT
VO

```

OED9 2613 .SBTTL PROCESS RECEIVED SLOT DATA
OED9 2614 :++
OED9 2615 : PROCESS RECEIVED SLOT DATA
OED9 2616 :
OED9 2617 : Functional description:
OED9 2618 :
OED9 2619 : Now we are set up to pass data to and from the class driver:
OED9 2620 :
OED9 2621 : The following loop copies data from the receive buffer into the class
OED9 2622 : driver.
OED9 2623 :
OED9 2624 : Inputs:
OED9 2625 : R0-R4 available
OED9 2626 : R5 = UCB address
OED9 2627 : R6 = CXB address
OED9 2628 : R7 = current receive buffer slot address
OED9 2629 : R8 = CSB address
OED9 2630 : R9 = current transmit buffer slot address
OED9 2631 : R10 = transmit buffer bytes left
OED9 2632 : R11 = scratch (used as current input slot character copy base register)
OED9 2633 : AP = scratch (used as loop count register)
OED9 2634 : FP = number of slots to process in received message
OED9 2635 :
OED9 2636 : Outputs:
OED9 2637 : none.
OED9 2638 :
OED9 2639 : We continue by processing the output slot loop.
OED9 2640 :
OED9 2641 :--
OED9 2642 :
OED9 2643 :.ENABLE LOCAL_BLOCK
OED9 2644 :SLOT_LOOP IN: ; Input data slot loop
OED9 2645 :MOVZBL SLT_B_DSTID(R7),R1 ; Is this slot alive ?
OED9 2646 :BNEQ 20$ ; Br if possible
OED9 2647 :TSTB SLT_B_SRCID(R7) ; Is the remote index valid?
OED9 2648 :BEQL 10$ ; Br if not, ignore slot
OED9 2649 :
OED9 2650 : ; Zero local slot index with non-zero remote (someone's attempting
OED9 2651 : ; to login). Must be a start slot!
OED9 2652 :
OED9 2653 :EXTZV #4,#4,SLT_B_CRED(R7),R0 ; Get slot type
OED9 2654 :
OED9 2655 :CMPB #SLT_C_STR_SLOT,R0 ; Is this a start slot ?
OED9 2656 :BNEQ LOOP_ABORT ; Br if not, ignore slot
OED9 2657 :BSBW LT$ACC_UCB ; Try to allocate an LT UCB
OED9 2658 :BLBS R0,10$ ; Br if success
OED9 2659 :INC_CTR GMB$RESOURCE+GMB_AREA ; Else, no UCB for us to use
OED9 2660 :
OED9 2661 : ; Send back a reject slot in the case of resource errors
OED9 2662 :
OED9 2663 :CPL R10,- ; Enough room for REJECT slot header?
OED9 2664 :#SLT_S_REJ_SLOT
OED9 2665 :BLSS 10$ ; Nope, just ignore the data
OED9 2666 :INCB CSB_B_NUM_SLOTS(R8) ; Increment slot count
OED9 2667 :MOVB SLT_B_SRCID(R7),(R9)+ ; Copy local id from receive message
OED9 2668 :CLRB (R9)+ ; Tell him we are not here !
OED9 2669 :CLRB (R9)+ ; Load status byte count
OED9 2670 :MOVB #SLT_C_REJ_SLOT@4,(R9) ; Load REJECT slot format

```

```

51 67 9A
   3D 12
01 A7 95
   35 13
50 04 04 EF
   03 A7
   50 09 91
   57 12
   0656 30
   24 50 E8
04 5A D1
89 5D A8 96
01 A7 90
   89 94
   89 94
69 C0 8F 90

```

```

: JAY0002
: JAY0002
: JAY0002
: JAY0002
: JAY0002
: JAY0002
: JAY0002
: JAY0002

```

```

89 05 88 OF12 9 BISB #SLT_C_RESOURCE,(R9)+ ; Load reason code (resource error)
5A 04 C2 OF15 10 SUBL2 #SLT_S_REJ_SLOT,R10 ; Adjust bytes left for other slots
00EC 31 OF18 2674 10$: BRW NEXT_SLOT ; Just process the next slot.
OF1B 2675 20$:
OF1B 2676
OF1B 2677
: Validate UCB reference - non-zero local index
69 A8 51 91 OF1B 2678 CMPB R1,CSB_B_MAX_SLOTS(R8) ; Do a range check on the local index
58 1A OF1F 2679 BGTRU 90$ ; Br if not in range
55 68 A841 D0 OF21 2680 MOVL CSB_L_UCBLST-4(R8)[R1],R5 ; Get UCB address
2D 18 OF26 2681 BGEQ 70$ ; That slot doesn't live here anymore
01 A7 91 OF28 2682 CMPB SLT_B_SRCID(R7),UCBSB_LT_REMID(R5) ; Valid remote reference?
0140 C5 OF2B
63 13 OF2E 2683 BEQL LOAD_CREDITS ; Br if yes, process slot
01 A7 95 OF30 2684 TSTB SLT_B_SRCID(R7) ; Else, are we hanging up from remote?
10 12 OF33 2685 BNEQ LOOP_ABORT ; Br if no, invalid slot
OF35 2686
OF35 2687 : Received a zero in the slot source-id - is this a stop slot?
OF35 2688
OF35 2689 40$: EXTZV #4,#4,SLT_B_CRED(R7),R0 ; Get slot type
50 04 04 EF OF38
03 A7 OF3B
50 0D 91 OF3B 2690 CMPB #SLT_C_STP_SLOT,R0 ; Is this a STOP slot?
05 12 OF3E 2691 BNEQ LOOP_ABORT ; Br if not, do nothing
0740 30 OF40 2692 BSBW LT$HANGUP_UCB_NOW ; Hangup the UCB to logout process
4B 11 OF43 2693 BRB NEXT_SLOT0 ; Process next slot
OF45 2694
OF45 2695 : Flag protocol error and ignore slot
OF45 2696
OF45 2697 LOOP_ABORT:
OF45 2698 SETBIT #GHBSV_INVALIDLOCID,-
OF45 2699 GHBSL_PROTOMASK+GHB_AREA ; Invalid remote ID or slot format
OF4B 2700 INC_CTR GHBSL_PROTOCOL+GHB_AREA ;
OF55 2701
OF55 2702 : Conditionally send stop slot
OF55 2703
OF55 2704
01 A7 95 OF55 2705 70$: TSTB SLT_B_SRCID(R7) ; Check remote slot index
36 13 OF58 2706 BEQL NEXT_SLOT0 ; Br if empty slot
OF5A 2707
OF5A 2708 ASSUME SLT_S_STP_SLOT&1 EQ 1 ; Assume we need rounding
06 5A D1 OF5A 2709 CMPL R10,- ; Enough room for STOP slot header?
OF5D 2710 #SLT_S_STP_SLOT+1
OF5D 2711 BLSS NEXT_SLOT0 ; nope, just ignore the data
5D A8 96 OF5F 2712 INCB CSB_B_NUM_SLOTS(R8) ; Increment slot count
89 01 A7 90 OF62 2713 MOVB SLT_B_SRCID(R7),(R9)+ ; Copy local id from receive message
89 89 94 OF66 2714 CLRB (R9)+ ; Tell him we are not here !
89 01 90 OF68 2715 MOVB #SLT_S_STP_LEN,(R9)+ ; Load slot byte count
89 D0 8F 90 OF6B 2716 MOVB #SLT_C_STP_SLOT&4,(R9)+ ; Load stop slot format
89 03 90 OF6F 2717 MOVB #SLT_C_INV_SLOT,(R9)+ ; load reason code
OF72 2718 ASSUME SLT_S_STP_SLOT&1 EQ 1 ; Assume we need rounding
89 89 94 OF72 2719 CLRB (R9)+ ; round up to even boundary
5A 06 C2 OF74 2720 SUBL2 #SLT_S_STP_SLOT+1,R10 ; Adjust bytes left for other slots
17 11 OF77 2721 BRB NEXT_SLOT0 ; Go do next input slot
OF79 2722
OF79 2723 : Range check error, is this really a disconnect slot ?
OF79 2724
01 A7 95 OF79 2725 90$: TSTB SLT_B_SRCID(R7) ; Empty remote slot index? (stop slot?)
12 13 OF7C 2726 BEQL NEXT_SLOT0 ; Br if yes, ignore the slot.

```


		OF7E	2727	SETBIT	#GHBSV	INVALIDREMID,-	
		OF7E	2728		GHBSL	PROTOMASK+GHB	AREA; Invalid LOCID with REMID non_zero
		OF84	2729	INC_CTR	GHBSL	PROTOCOL+GHB	AREA ; Count as protocol error
C5	11	OF8E	2730	BRB	70\$; Send a stop slot
		OF90	2731				
		OF90	2732	NEXT_SLOTO:			
0074	31	OF90	2733	BRW	NEXT_SLOT		; Branch aid
		OF93	2734				
		OF93	2735	.DISABLE	LOCAL_BLOCK		

```

OF93 2737 .SBTTL PROCESS SLOT CREDITS RECEIVED
OF93 2738 :++
OF93 2739 :
OF93 2740 : Update UCB credit account from received message slots
OF93 2741 :
OF93 2742 :--
OF93 2743 LOAD_CREDITS:
50 03 A7 90 OF93 2744 MOVB SLT_B_CRED(R7),R0 ; Get credits and slot type
OF97 2745 ASSUME SLT_C_DA_SLOT EQ 0 ; assumption
04 00 EF OF97 2746 BGEQ 50$ ; Got a DATA_A slot
50 03 A7 EF OF99 2747 EXTZV #0, #4, SLT_B_CRED(R7), R0 ; Get the credit counts
04 04 EF OF9C 2748 EXTZV #4, #4, SLT_B_CRED(R7), R1 ; Get the slot type
51 03 A7 OFA2 2749 $DISPATCH R1,TYPE=B,- ; Dispatch on slot type
OFA5 2750 <- ; slot type ; action
OFA5 2751 <SLT_C_DB_SLOT 20$>,- ; DATA_B slot???
OFA5 2752 <SLT_C_ATT_SLOT 40$>,- ; Attention slot
OFA5 2753 >
FF95 31 OFAD 2754 BRW LOOP_ABORT ; All others, including Reject slot
OFAD 2755 : DATA_B slot
OFB0 2756 :
OFB0 2757 :
OFB0 2758 :
01 88 OFB0 2759 20$: BISB #UCBSM_LT_DATA,- ; Remember to at least send back the
0142 C5 OFB2 2760 UCBSB_LT_DATAW(R5) ; credits!
0147 C5 96 OFB5 2761 INCB UCBSB_LT_XCRED(R5) ; One less credit on remote end (negative)
03 15 OFB9 2762 BLEQ 40$ ; Br if credits not exceeded, proceed
06D2 30 OFBB 2763 BSBW LT$HANGUP_UCB ; Else, Hangup the UCB to logout process
OFBE 2764 : ATTention slot, IGNORE DATA
OFBE 2765 :
0146 C5 50 80 OFBE 2766 40$: ADDB R0,UCBSB_LT_TCRED(R5) ; Update credits
42 11 OFC3 2767 BRB NEXT_SLOT ; Skip to next slot
OFC5 2768 : RUN slot
OFC5 2769 :
OFC5 2770 :
0146 C5 50 80 OFC5 2771 50$: ADDB R0,UCBSB_LT_TCRED(R5) ; Update credits
OFC5 2772 :
OFC5 2773 :
OFC5 2774 :
OFC5 2775 .IF DEFINED LT_CRED_CHECK
OFC5 2776 CMPB UCBSB_LT_TCRED(R5),#2
OFC5 2777 BLEQ 60$ ; Br if ok
OFC5 2778 SETBIT #GHBSV_BADCREDITS,-
OFC5 2779 GHBSL_PROTOMASK+GHBSL_AREA, Too many credits in host
OFC5 2780 INC CTR GHBSL_PROTOCOL+GHBSL_AREA ; Too many credits in host
OFC5 2781 BSBW LT$HANGUP_UCB ; Hangup the UCB to logout process
OFC5 2782 60$:
OFC5 2783 .ENDC ;; DEFINED LT_CRED_CHECK
OFC5 2784 :
OFC5 2785 :
OFC5 2786 : Move terminal data through class driver
OFC5 2787 :
OFC5 2788 R3 - contains the character
OFC5 2789 R5 - UCB address
OFC5 2790 :
50 02 A7 9A OFCA 2791 MOVZBL SLT_B_COUNT(R7),AP ; Character count for this slot

```

```

58 04 37 13 OFCE 2792      BEQL  NEXT_SLOT          : No input here
0143 C5 00 9E OFD0 2793      MOVAB SLT_T_DATA(R7),R11 : Source of characters
      2D OFD4 2794      BBS  #UCBSV_LT_HANGUP, -  : If device is hanging up, do not con-
      OFD9 2795
0147 C5 96 OFDA 2795      UCBSB_LT_LATSTS(R5),NEXT_SLOT : fuse class driver with more data
      05 15 OFDA 2796      INCB UCBSB_LT_XCRED(R5)    : One less credit on remote end (negative)
      06AD 30 OFDE 2797      BLEQ 70$                : Br if credits not exceeded, proceed
      22 11 OFE0 2798      BSRW LT$HANGUP_UCB      : Else, Hangup the UCB to logout process
      OFE3 2799      BRB  NEXT_SLOT         : Continue
      OFE5 2800 70$:
      OFE5 2801
      OFE5 2802
      OFE5 2803
      OFE5 2804
      OFE5 2805      ASSUME UCBSB_LT_LATSTS EQ UCBSB_LT_DATAW+1
      OFE5 2806      BISW #<UCBSM_LT_INPUT@8>!UCBSM_LT_DATA,- : Data potentially available
0142 C5 88 OFE9 2807      UCBSB_LT_DATAW(R5)      : and INPUT stream in enabled
64 A5 02 88 OFEC 2808      BISB #UCBSM_INT,UCBSL_STS(R5); Force no data to be returned
      OFFO 2809
      OFFO 2810
      OFFO 2811
      OFFO 2812 80$:      MOVZBL (R11)+,R3          : Get next character
      OFF3 2813      BBS  #UCBSV_LT_OVFLOW,- : Br if overflow, wait for XOM
      OFF5 2814      UCBSB_LT_LATSTS(R5),90$
0143 C5 16 OFF9 2815      JSB @UCBSL_TT_PUTNXT(R5) : Pass character to class driver
0110 D5 12 OFFD 2816      BNEQ ECHO_ERROR         : Br if character being echoed
      EE 5C F5 OFFF 2817      SOBGTR AP,80$          : Do all character's
      1002 2818
      1002 2819 90$:      BICB #UCBSM_LT_INPUT,- : Input stream is now disabled
0143 C5 1004 2820      UCBSB_LT_LATSTS(R5)
      1007 2821
      1007 2822
      1007 2823
      1007 2824
      1007 2825      NEXT_SLOT:
02 A7 9A 1007 2825      MOVZBL SLT_B_COUNT(R7), R0 : Max bytes in the slot
05 A740 9E 1008 2826      MOVAB <SLT_T_DATA+1>(R7)- : Compute address of next slot +1
      57 100F 2827      [R0], R7                : to help round to word
      57 01 CA 1010 2828      BICL #1, R7             : make it a word boundary.
      5D 07 D7 1013 2829      DECL TP                 : Loop on all slots
      2D 13 1015 2830      BEQL SLOT_LOOP_OUT    : Done all input slots, now process
      1017 2831
      FEFB 31 1017 2832      BRW  SLOT_LOOP_IN     : any output data
      101A 2833
      101A 2834      ECHO_ERROR:         : Character being echoed???
      101A 2835      DEBUG                :.88 Should not get to here

```

```

1020 2837 .SBTTL BUILD A STOP SLOT
1020 2838 :++
1020 2839 : BUILD A STOP SLOT
1020 2840 :
1020 2841 : Functional description:
1020 2842 :
1020 2843 : Build a stop slot in the out message buffer, then delete the UCB.
1020 2844 :
1020 2845 : Inputs:
1020 2846 : R5 = UCB address
1020 2847 : R8 = CSB address
1020 2848 : R9 = current transmit buffer slot address
1020 2849 : R10 = transmit buffer bytes left
1020 2850 :
1020 2851 : Outputs:
1020 2852 : R5 = UCB address
1020 2853 : R8 = CSB address
1020 2854 : R9 = current transmit buffer slot address
1020 2855 : R10 = transmit buffer bytes left
1020 2856 :
1020 2857 :--
1020 2858 :
1020 2859 SEND_STOP_SLOT:
014B C5 95 1020 2860 TSTB UCBSB_LT_CURC(R5) ; Is UCB data waiting?
03 13 1024 2861 BEQL 30$ ; Br if no, send the STOP slot
00B9 31 1026 2862 BRW CREDIT_CHECK ; Send data if our credit is good
1029 2863
04 5A D1 1029 2864 30$: CMPL R10, #SLT_T_DATA ; Enough room for STOP slot?
1E 15 102C 2865 BLEQ ONE_LESS ; Br if no, send it next time
SD AB 96 102E 2866 INCB CSB_B_NUM_SLOTS(R8) ; Update slot count
1031 2867
89 0140 C5 90 1031 2868 ASSUME SLT_B_DSTID EQ 0
1031 2869 MOVB UCBSB_LT_REMID(R5), (R9)+ ; Send the remote slot index
1036 2870
1036 2871 ASSUME SLT_B_SRCID EQ SLT_B_DSTID+1
89 B4 1036 2872 ASSUME SLT_B_COUNT EQ SLT_B_SRCID+1
1036 2873 CLRW (R9)+ ; Zero SRCID for a stop slot
1038 2874 ; and no data
1038 2875 ASSUME SLT_B_CRED EQ SLT_B_COUNT+1
89 D0 8F 90 1038 2876 MOVB #SLT_C_STP_SLOT@4,(R9)+ ; Load stop slot format & no credits
5A 04 C2 103C 2877 SUBL #SLT_T_DATA,R10 ; Adjust bytes left for other slots
103F 2878 ;;;** Skip call if HOST BIND
0694 30 103F 2879 BSBW LTKILLUCB ; Wipe out the UCB now.
08 11 1042 2880 BRB ONE_LESS ; Get next slot

```

LTI
V04

:J
-1

:J
:J
:J
-3

```

1044 2882 .SBT~L PROCESS OUTPUT SLOT DATA
1044 2883 :++
1044 2884 : PROCESS OUTPUT SLOT DATA
1044 2885 :
1044 2886 : Functional description:
1044 2887 :
1044 2888 : Now output any waiting data from each UCB on this circuit and any
1044 2889 : data returned from the class driver is transferred to the appropriate
1044 2890 : output slot in the transmit buffer.
1044 2891 :
1044 2892 : Inputs:
1044 2893 : R0-R4 available
1044 2894 : R5 = UCB address
1044 2895 : R6,R7 = scratch
1044 2896 : R8 = CSB address
1044 2897 : R9 = current transmit buffer slot address
1044 2898 : R10 = transmit buffer bytes left
1044 2899 : R11-FP = scratch
1044 2900 :
1044 2901 : Outputs:
1044 2902 : none.
1044 2903 :
1044 2904 : We continue processing in the send reply message section.
1044 2905 :
1044 2906 :--
1044 2907 SLOT_LOOP OUT:
5D 2D AB 9A 1044 2908 MOVZBL CSB_B_REF(C(R8)),FP ; Set output UCB loop count
57 6C AB DE 1048 2909 MOVAL CSB_L_UCBLST(R8),R7 ; Set base address of UCB vector
104C 2910 ONE_LESS: ; One less unit left
SD D7 104C 2911 DECL FP ; Any UCBs left ?
03 18 104E 2912 BGEQ 10$ ; Br if yes
01EB 31 1050 2913 BRW LOOP_EXIT ; Else, exit loop
1053 2914 10$:
55 87 D0 1053 2915 MOVL (R7)+,R5 ; Get next UCB address
FB 13 1056 2916 BEQL 10$ ; ignore holes in non-dense vector
1058 2917
1058 2918 ; Dispatch on our terminal UCB state
1058 2919
1058 2920 $DISPATCH UCBSB_LT_STATE(R5),TYPE=B,-
1058 2921 <- : state : action
1058 2922 <UCBSC_LT_STATE_STOP SEND_STOP_SLOT>,- ; STOP state, send stop slot
1058 2923 <UCBSC_LT_STATE_KILL ONE_LESS>,- ; KILL state, ignore this
1058 2924 >
1062 2925 ; RUN and START, just fall through
0142 C5 95 1062 2926
E4 13 1066 2927 TSTB UCBSB_LT_DATAW(R5) ; Is there any data waiting?
1068 2928 BEQL ONE_LESS ; Br if no skip this UCB
1068 2929
1068 2930 : Check for ABORT message first
1068 2931 :
1A 0142 C5 02 E5 1068 2932 BBCC #UCBSV_LT_ABORT,- ; Br if no abort requested
106A 2933 UCBSB_LT_DATAW(R5),30$ ; and clear flag
106E 2934
106E 2935 ; Send an ATTENTION slot
106E 2936
06 5A D1 106E 2937 ASSUME SLT_S_ATT_SLOT&1 EQ 1 ; Assume we need rounding
106E 2938 CMPL R10, #SLT_S_ATT_SLOT+1 ; Enough room for ATTENTION slot ?

```

```

      3D 15 1071 2939      BLEQ 50$      : Br if no, skip it (use br helper)
      SD AB 96 1073 2940      INCB CSB_B_NUM_SLOTS(RB) : Update slot count
      1076 2941
      1076 2942      ASSUME UCBSB_LT_LOCID EQ UCBSB_LT_REMID+1
      1076 2943      ASSUME SLT_B_DSTID EQ 0
      1076 2944      ASSUME SLT_B_SRCID EQ SLT_B_DSTID+1
      1076 2945      ASSUME SLT_B_COUNT EQ SLT_B_SRCID+1
      1076 2946      ASSUME SLT_B_TYPE EQ SLT_B_COUNT+1
      1076 2947      ASSUME SLT_B_ATT_CONTROL EQ SLT_B_TYPE+1
      1076 2948
      89 0140 C5 80 1076 2949      MOVW UCBSB_LT_REMID(R5), (R9)+ : Send the remote slot id
      89 01 90 107B 2950      MOVB #SLT_S_ATT_LEN, (R9)+ : Set ATTENTION slot data size
      89 80 8F 90 107E 2951      MOVB #SLT_C_ATT_SLOT@4, (R9)+ : Load ATTn type & no credits
      1082 2952
      1082 2953
      89 20 80 1082 2954      ASSUME SLT_S_ATT_SLOT@1 EQ 1 : Assume we need rounding
      SA 06 C2 1085 2955      MOVW #SLT_ATT_ABORT, (R9)+ : Send abort flag & zero byte
      1088 2956      SUBL #SLT_S_ATT_SLOT+1, R10 : Account from room taken
      1088 2957      :
      1088 2958      : Check for flow control message
      1088 2959      :
      54 0142 C5 E5 1088 2960      30$: BBCC #UCBSV_LT_FLOW,- : Br if no flow requested
      108A 2961      UCBSB_LT_DATAW(R5), 90$ : and clear flag
      108E 2962      :
      108E 2963      : Send DATA_B slot
      108E 2964      :
      0146 C5 97 108E 2965      DECB UCBSB_LT_TCRED(R5) : Use up a credit
      0C 18 1092 2966      BGEQ 40$ : Br if okay, continue
      0146 C5 96 1094 2967      INCB UCBSB_LT_TCRED(R5) : Else, restore credits
      1098 2968      SETBIT #UCBSV_LT_FLOW,- : Remember to send flow control
      1098 2969      UCBSB_LT_DATAW(R5) : when we have good credit
      42 11 109E 2970      BRB 90$ : And continue
      10A0 2971
      10A0 2972      40$:
      10A0 2973      ASSUME SLT_S_DB_SLOT@1 EQ 1 : Assume we need rounding
      OA 5A D1 10A0 2974      CMPL R10,- : Enough room for DATA_B slot ?
      10A3 2975      #<SLT_S_DB_SLOT+1> : rounded up
      3D 15 10A3 2976      50$: BLEQ 90$ : Br if no, skip it (branch helper)
      SD AB 96 10A5 2977      INCB CSB_B_NUM_SLOTS(RB) : Update slot count
      10A8 2978
      10A8 2979      ASSUME UCBSB_LT_LOCID EQ UCBSB_LT_REMID+1
      10A8 2980      ASSUME SLT_B_DSTID EQ 0
      10A8 2981      ASSUME SLT_B_SRCID EQ SLT_B_DSTID+1
      10A8 2982      ASSUME SLT_B_COUNT EQ SLT_B_SRCID+1
      10A8 2983      ASSUME SLT_B_TYPE EQ SLT_B_COUNT+1
      10A8 2984      ASSUME SLT_B_DB_CONTROL EQ SLT_B_TYPE+1
      10A8 2985      ASSUME SLT_B_DB_OFOF EQ SLT_B_DB_CONTROL+1
      10A8 2986      ASSUME SLT_B_DB_OFOW EQ SLT_B_DB_OFOF+1
      10A8 2987      ASSUME SLT_B_DB_IFOF EQ SLT_B_DB_OFOW+1
      10A8 2988      ASSUME SLT_B_DB_IFOW EQ SLT_B_DB_IFOF+1
      10A8 2989
      89 0140 C5 80 10A8 2990      MOVW UCBSB_LT_REMID(R5), (R9)+ : Send the remote slot id
      89 05 90 10AD 2991      MOVB #SLT_S_DB_LEN, (R9)+ : Set DATA_B data size
      89 A0 8F 90 10B0 2992      MOVB #SLT_C_DB_SLOT@4, (R9)+ : Load DATA_B type & no credits
      10B4 .1
      10B4 .2
      10B4 .3
      : If the terminal is set to PASSALL, then we will disable all
      : flow control. Else, we will use XON/XOFF flow control if we

```

:JAV0001
:JAV0001
:JAV0001

```

:JAY0001      1084      .4      : are set /HOSTSYNC or we will use NULL/BELL flow control if
:JAY0001      1084      .5      : we are set /NOHOSTSYNC.
:JAY0001      1084      .6      :
:JAY0001      1084      .7      :
:JAY0001      11131113 89 0A 90 1084 .7      MOVB #SLT_DB_M_IFDIS!SLT_DB_M_OFDIS,(R9)+ ; Assume no flow control
:JAY0001      11131113 8F D0 1087 .8      MOVL #<17245!219216>!<17285!21920>,(R9)+ ; Assume default values
:JAY0001      1A 44 A5 0A 90 108D .9      BBS #TTSV_PASSALL,UCBSL_DEVDEPEND(R5),70$ ; Br if PASSALL
:JAY0001      FB A9 05 90 10C3 .10     MOVB #SLT_DB_M_IFENA!SLT_DB_M_OFENA,-5(R9) ; Else, use flow control
:JAY0001      04 44 A5 04 E0 10C7 .11     BBS #TTSV_HOSTSYNC,UCBSL_DEVDEPEND(R5),60$ ; Assumed ^S and ^Q
:JAY0001      FE A9 07 B0 10CC .12     MOVW #<7205,-2(R9) ; Else, we were wrong
:JAY0001      10D0      .13
:JAY0001      10D0      .14 60$: ; Check /TTSYNC
:JAY0001      10D0      .15
:JAY0001      08 44 A5 05 E0 10D0 .16     BBS #TTSV_TTSYNC,UCBSL_DEVDEPEND(R5),70$ ; Br if ^S and ^Q
:JAY0001      10D5      .17 ; assumption was correct
:JAY0001      FB A9 04 8A 10D5 3003 BICB #SLT_DB_M_OFENA,-5(R9) ; Else, clear output flow enable
:JAY0001      FB A9 08 88 10D9 3004 BISB #SLT_DB_M_OFDIS,-5(R9) ; and set output flow disable
:JAY0001      10DD      3005
:JAY0001      10DD      3006 70$:
:JAY0001      10DD      3007 ASSUME SLT_S_DB_SLOT&1 EQ 1 ; Assume we need rounding
:JAY0001      5A 89 94 10DD 3008 CLRB (R9)+ ; Round up
:JAY0001      0A 0A C2 10DF 3009 SUBL #SLT_S_DB_SLOT+1,R10 ; Adjust bytes left (rounded up)
:JAY0001      10E2      3010
:JAY0001      10E2      3011 ;
:JAY0001      10E2      3012 ; Check if there is any output data waiting
:JAY0001      10E2      3013 ;
:JAY0001      10E2      3014 90$:
:JAY0001      10E2      3015

```

```

10E2 3017 :++
10E2 3018 :
10E2 3019 : Any credits and enough room in output buffer for this slot ?
10E2 3020 :
10E2 3021 :--
10E2 3022 CREDIT_CHECK: ; UNUSED LABEL (breaks up LSB)
10E2 3023 :
10E2 3024 : Flush all characters to transmit buffer
10E2 3025 :
10E2 3026 : R8 = CSB address
10E2 3027 : R5 = UCB address
10E2 3028 :
56 0149 C5 9A 10E2 3029 MOVZBL UCBSB_LT_SLOTSZ(R5),R6 ; Get maximum slot size
56 5A D1 10E7 3030 CMPL R10,R6 ; Is space left greater?
56 03 14 10EA 3031 BGTR 10$ ; Br if yes, then we can use it
56 5A D0 10EC 3032 MOVL R10,R6 ; Else, use lesser size
56 04 C2 10EF 3033 10$: SUBL #SLT_T_DATA, R6 ; Subtract slot header size
08 14 10F2 3034 BGTR 30$ ; Br if enough room
10F4 3035 20$: SETBIT #FLAG_V_RRF,- ; Else, indicate response requested
10F4 3036 CSB_B_FLAG(R8) ; since not all data fit in buffer
FF50 31 10F9 3037 BRW ONE_CESS ; No room left in output slot
5C 56 D0 10FC 3038 30$: MOVL R6,AP ; Get room left in output slot
5B 04 A9 9E 10FF 3039 MOVAB SLT_T_DATA(R9),R11 ; Sink of characters
1103 3040 :
1103 3041 : Check if we have credits, or if we need to return credits
1103 3042 : before building the message.
1103 3043 :
0146 C5 95 1103 3044 TSTB UCBSB_LT_TCRED(R5) ; Any credits we can use to send msg?
09 14 1107 3045 BGTR 40$ ; Br if yes, okay to build data message
1109 3046 ASSUME LAT$C_MAX_RCRED EQ 1
0147 C5 95 1109 3047 TSTB UCBSB_LT_XCRED(R5) ; Remote have any credits?
E5 12 110D 3048 BNEQ 20$ ; Br if yes, just leave now
00C4 31 110F 3049 BRW USE_CREDIT ; Else, send back credits
1112 3050 40$:
1112 3051 : Any data from startio ?
1112 3052 :
52 014B C5 9A 1112 3053 MOVZBL UCBSB_LT_CURC(R5),R2 ; Get characters in startio ?
4A 13 1117 3054 BEQL 70$ ; Br if none
53 014C C5 9E 1119 3055 MOVAB UCBSB_LT_CBUF(R5),R3 ; Set address of buffer
52 5C D1 111E 3056 CMPL AP,R2 ; Got enough room in xmit buffer?
29 18 1121 3057 BGEQ 60$ ; Br if yes
1123 3058 :
1123 3059 : R2 = COUNT
1123 3060 : R3 -> DATA
1123 3061 :
014B C5 5C 82 1123 3062 SUBB2 AP, UCBSB_LT_CURC(R5) ; Save residual count
55 DD 1128 3063 PUSHL R5 ; Save the UCB address
68 63 5C 28 112A 3064 MOVCL AP,(R3),(R11) ; Copy the string from UCB to output bfr
55 8ED0 112E 3065 POPL R5 ; Restore the UCB address
5B 53 D0 1131 3066 MOVL R3,R11 ; Set address of next available byte
52 014C C5 9E 1134 3067 MOVAB UCBSB_LT_CBUF(R5), R2 ; Copy residual chars to beginning
53 014B C5 9A 1139 3068 MOVZBL UCBSB_LT_CURC(R5), R3 ; of UCB buffer area
55 DD 113E 3069 PUSHL R5 ; Save the UCB address
62 61 53 28 1140 3070 MOVCL R3,(R1),(R2) ; Move the string up in the UCB
55 8ED0 1144 3071 POPL R5 ; Restore the UCB address
5C D4 1147 3072 CLRL AP ; No more room left
00BA 31 1149 3073 BRW USE_CREDIT ; Use a credit and send them

```



```

114C 3074
014B C5 94 114C 3075 60$: CLR B UCBSB_LT_CURC(R5) ; No data now
5C 52 C2 1150 3076 : SUBL2 R2,AP ; Adjust room left in slot
55 DD 1153 3077 : PUSHL R5 ; Save the UCB address
6B 63 52 28 1155 3078 : MOVC3 R2,(R3),(R11) ; Copy the string
55 8ED0 1159 3079 : POPL R5 ; Restore the UCB address
5B 53 D0 115C 3080 : MOVL R3,R11 ; Set address of next available byte
5C D5 115F 3081 : TSTL AP ; Any room left in slot?
73 15 1161 3082 : BLEQ USE_CREDIT ; No, just transmit data
1163 3083 :
1163 3084 : Check if there is any data from last time
1163 3085 :
70$: CVTWL UCBSW TT_OUTLEN(R5),R2 ; Any data left from last time?
52 0120 C5 32 1163 3086 : BLEQ GET_MORE ; Br if no, try for new data
53 011C C5 D0 116A 3088 : MOVL UCBSL TT_OUTADR(R5),R3 ; Get address of data
52 5C D1 116F 3089 : CMPL AP,R2 ; Enough room in buffer?
03 18 1172 3090 : BGEQ 80$ ; Br if yes
52 5C D0 1174 3091 : MOVL AP,R2 ; Use all the space we have left
0120 C5 52 A2 1177 3092 80$: SUBW2 R2,UCBSW TT_OUTLEN(R5) ; Adjust descriptor
011C C5 52 C0 117C 3093 : ADDL2 R2,UCBSL TT_OUTADR(R5) ;
5C 52 C2 1181 3094 : SUBL R2,AP ; Adjust count of chars left
55 DD 1184 3095 : PUSHL R5 ; Save the UCB address
6B 63 52 28 1186 3096 : MOVC3 R2,(R3),(R11) ; Copy the string
55 8ED0 118A 3097 : POPL R5 ; Restore the UCB address
5B 53 D0 118D 3098 : MOVL R3,R11 ; Set address of next available byte
5C D5 1190 3099 : TSTL AP ; Any room left now?
02 14 1192 3100 : BGTR GET_MORE ; Br if yes, get some more data
40 11 1194 3101 : BRB USE_CREDIT ; Else, try next slot
1196 3102 :
1196 3103 :
1196 3104 : Loop draining class driver until no data is left or buffer is full
1196 3105 :
3106 GET_MORE:
010C D5 16 1196 3107 : JSB @UCBSL TT_GETNXT(R5) ; Get characters from class driver
119A 3108 :
119A 3109 : R1-R4 modified
119A 3110 : R3 = ?? and cc = 0 ; No more data
119A 3111 : R3 = char and cc = + ; One character
119A 3112 : R3 = ?? and cc = - ; Burst data (TT_OUTADR, TT_OUTLEN)
119A 3113 : R5 = ucb address
119A 3114 :
40 13 119A 3115 : BEQL USE_CREDIT1 ; No characters
09 19 119C 3116 : BLSS GOT_A_LOT ; Got more than one character
119E 3117 :
8B 53 90 119E 3118 : MOVB R3,(R11)+ ; Copy character to output
5C D7 11A1 3119 : DECL AP ; Adjust room in slot
F1 14 11A3 3120 : BGTR GET_MORE ; Go get more
2F 11 11A5 3121 : BRB USE_CREDIT ; No room left in slot
11A7 3122 :
3123 GOT_A_LOT:
53 011C C5 D0 11A7 3123 : MOVL UCBSL TT_OUTADR(R5),R3 ; Get output buffer address
52 0120 C5 3C 11AC 3124 : MOVZWL UCBSW TT_OUTLEN(R5),R2 ; and charcter count
52 5C D1 11B1 3125 : CMPL AP,R2 ; Got enough room in output buffer?
03 18 11B4 3126 : BGEQ 10$ ; Br if yes
52 5C D0 11B6 3127 : MOVL AP,R2 ; Output maximum possible
5C 52 C2 11B9 3128 10$: SUBL2 R2,AP ; Adjust room left in slot
011C C5 52 C0 11BC 3129 : ADDL R2,UCBSL TT_OUTADR(R5) ; Update output address
0120 C5 52 A2 11C1 3130 : SUBW2 R2,UCBSW TT_OUTLEN(R5) ; and charcter count

```

6B	63	55	DD	11C6	3131	PUSHL	R5	:	Save the UCB address
		52	28	11C8	3132	MOVCL	R2,(R3),(R11)	:	Copy the string
		55	8ED0	11CC	3133	POPL	R5	:	restore the UCB address
	5B	53	D0	11CF	3134	MOVL	R3,R11	:	set address of next available byte
		5C	D5	11D2	3135	TSTL	AP	:	Any room left in slot ?
		C0	14	11D4	3136	BGTR	GET_MORE	:	Br if yes, look for more

```

11D6 3138      .SBTTL CONSUME CREDIT WHEN FILLING IN SLOT FIELD
11D6 3139      :
11D6 3140      : If any data was added to output slot, use credit locally, pass credit
11D6 3141      : to other end if needed, and set circuit mode to unbalanced.
11D6 3142      :
11D6 3143      :
11D6 3144      USE_CREDIT: ENABL  LSB
64 A5 02 AB 11D6 3145      BISW  #UCBSM_INT,UCBSL_STS(R5); Set interrupt expected, locks
11DA 3146      : out transmits from STARTIO
OF 11 11DA 3147      BRB   10$      : Continue
11DC 3148      :
64 A5 02 AA 11DC 3149      USE_CREDIT1:
05 0143 C5 E0 11E0 3150      PICW  #UCBSM_INT,UCBSL_STS(R5); Clear interrupt expected
0142 C5 8A 11E2 3151      JBS   #UCBSV_LT_OVFLOW,-; Don't clear data waiting flag
11E6 3152      UCBSB_LT_LATSTS(R5),10$; if we're waiting for OVFLOW to clear
11E8 3153      BICB  #UCBSM_LT_DATA,-; No more output data waiting
11EB 3154      UCBSB_LT_DATAW(R5)
11EB 3155      10$:
11EB 3156      ::BB
11EB 3157      SUBL3 AP,R6,AP; form character count in AP
11EB 3158      DECB  UCBSB_LT_TCRED(R5); Use up a credit
11EB 3159      BISB  #FLAG_M_RRF,-; Set response requested flag
SC 56 5C A8 88 11EB 3160      BISB  #FLAG_M_RRF,-; Set response requested flag
5C 56 5C C3 11ED 3161      CSB B FLAG(R8)
0146 C5 97 11EF 3162      SUBL3 AP,R6,AP; Calculate character count in AP
0147 C5 81 11F3 3163      BEQL  20$; Br if no data, DON'T use credit
50 01 08 11F5 3164      DECB  UCBSB_LT_TCRED(R5); Else, use up a credit
0147 C5 81 11F9 3165      ASSUME SLT_C_DA_SLOT EQ 0
50 01 08 11F9 3166      ADDB3 UCBSB_LT_XCRED(R5)-; Calculate credits to send to remote
1201 3167      #LATSC_MAX_RCRED,R0; system as DATA_A slot
0147 C5 81 1201 3168      BRB   40$; Send credits and data
50 01 08 1201 3169      ASSUME SLT_C_DA_SLOT EQ 0
0148 C5 50 80 1201 3170      ADDB3 UCBSB_LT_XCRED(R5)-; See if we need to send
0147 C5 50 82 1205 3171      #LATSC_MAX_RCRED,R0; a credit to the remote
1207 3172      BEQL  60$; Br if no, send nothing
1209 3173      40$:
120E 3174      ADDB  R0,UCBSB_LT_RCRED(R5); Else, add in credits to send
1213 3175      SUBB  R0,UCBSB_LT_XCRED(R5); Credits have been extended
1213 3176      :
1213 3177      : Setup slot header and advance pointer to next transmit slot header
1213 3178      :
1213 3179      ASSUME UCBSB_LT_LOCID EQ UCBSB_LT_REMID+1
1213 3180      ASSUME SLT_B_SRCID EQ SLT_B_DSTID+1
0140 C5 80 1213 3181      MOVW  UCBSB_LT_REMID(R5),-; Set up slot connection ids
69 1217 3182      SLT B DSTID(R9)
02 A9 5D A8 96 1218 3183      INCB  CSB B RUN_SLOTS(R8); Update slot count
0148 C5 90 1218 3184      MOVB  AP,SLT_B_COUNT(R9); Load character count
03 A9 90 121F 3185      MOVB  UCBSB_LT_RCRED(R5),-; Load slot type and credits
1223 3186      SLT B CRED(R9)
0148 C5 94 1225 3187      ASSUME SLT_C_DA_SLOT EQ 0
50 59 D0 1225 3188      CLRB  UCBSB_LT_RCRED(R5); Set RUN state, no more credits
59 58 D0 1229 3189      MOVL  R9, R0; Copy slot starting address
59 59 D6 122C 3190      MOVL  R11, R9; Address of start of next slot
59 01 CA 122F 3191      INCL  R9; Make word address
50 59 50 C3 1231 3192      BICL  #1, R9; which is even
5A 50 C3 1234 3193      SUBL3 R0, R9, R0; Compute bytes used by slot
5A 50 C2 1238 3194      SUBL2 R0, R10; Adjust bytes left for other slots

```

```

FE0E 31 123B 3195 60$: BRW ONE_LESS ; go do next UCB
123E 3196
123E 3197 LOOP_EXIT:
123E 3198 :
123E 3199 : Compute and save the number of bytes in the transmit buffer.
123E 3200 :
123E 3201 : R9 -> end of data in the transmit buffer
123E 3202 :
123E 3203 :
123E 3204 : There is a special case where the XMIT buffer can be taken,
123E 3205 : and that is when the circuit is halted. So if the circuit is
123E 3206 : in the HALT state, then we can assume we no longer have the
123E 3207 : XMIT buffer that we copied data into.
123E 3208 :
123E 3209 >DISPATCH CSB_B STATE(R8),TYPE=B,- ; Dispatch on our CSB state
123E 3210 <- : state : action
123E 3211 <CSB_C_STATE_HALT LTSREPLY_DONE>,- ; All done, exit
123E 3212 >
1245 3213
5D AB 95 1245 3214 TSTB CSB_B_NUM_SLOTS(R8) ; Are there any slots?
04 12 1248 3215 BNEQ 80$ ; Br if yes, continue
01 8A 124A 3216 BICB #FLAG M RRF,- ; Else, clear RRF flag
5C AB 124C 3217 CSB_B FLAG(R8)
51 34 AB D0 124E 3218 80$: MOVL CSB_B XBUFQ(R8), R1 ; Address of buffer
1C A1 59 D0 1252 3219 MOVL R9,-CXBSL_T_ENDADR(R1) ; Save end address of data
1256 3220 .DSABL LSB

```

```

1256 3222      .SBTTL SEND REPLY RESPONSE TO RECEIVED MESSAGE
1256 3223
1256 3224 LTSREPLY_MSG:
1256 3225 :
1256 3226 :   We come here to transmit a new frame in response to a received frame
1256 3227 :
1256 3228 :   R8 = CSB address
1256 3229 :
59 34 AB D0 1256 3230      MOVL CSB_Q_XBUFQ(R8),R9      ; Transmit buffer address
1256 3231
1256 3232 LTSREPLY_ALT:
1256 3233 :IF DEFINED LT_XMT_CHECK
1256 3234 :
1256 3235 :   Debug code
1256 3236 :
1256 3237 :   check the format of the slots and make sure that they are
1256 3238 :   well formatted.
1256 3239 :
1256 3240 :   R8 = CSB address
1256 3241 :   R9 = transmit buffer address
1256 3242 :   R0-R3 clobbered
1256 3243 :
1256 3244      MOVAQ CSB_Q_XBUFQ(R8), R0      ; Address of transmit head
1256 3245      CMPL R9, R0                  ; Is there a transmit buffer
1256 3246      BNEQ 5$                     ; Nope, its the queue head
1256 3247      DEBUG                        ; Can't be the queue head
1256 3248
1256 3249 5$:      MOVQ CSB_T_CIRCHDR(R8),-   ; Move the circ header
1256 3250      XMT_T_DATA(R9)              ; to look at real msg
1256 3251      CMPB XMT_B_FLAG(R9),-       ; Only analyse running msgs
1256 3252      #MYP_C_RUN@FLAG_V_MTYPE    ; with slots
1256 3253      BNEQ 50$                    ; Not a running message
1256 3254      MOVL CXBSL_T_ENDADR(R9), R2
1256 3255      MOVAB XMT_T_MDATA(R9), R0   ; First slot address
1256 3256      MOVZBL XMT_B_NUM_SLOTS(R9), R1 ; number of valid slots
1256 3257      BNEQ 10$                   ; We have some
1256 3258      ; No slot - message must only have a circuit header!
1256 3259      CMPL R2, R0                 ; Only circuit header?
1256 3260      BEQL 50$                   ; Br if yes, okay
1256 3261      DEBUG                        ; Not correct
1256 3262      BRB 50$
1256 3263
1256 3264 10$:     MOVZBL SLT_B_COUNT(R0), R3 ; Look to next slot
1256 3265      BGEQ 30$                  ; Must be greater or zero
1256 3266      DEBUG                        ; Maxc not valid
1256 3267 30$:     ADDL #SLT_T_DATA, R3 ; Slot header length
1256 3268      ADDL R3, R0
1256 3269      INCL R0                    ; Round to wrd boundary
1256 3270      BICL #1, R0
1256 3271      CMPL R0, R2
1256 3272      BLEQU 40$                 ; Still inside buffer?
1256 3273      DEBUG                        ; Yep
1256 3274 40$:     SOBGTR R1, 10$ ; Slot beyond buffer
1256 3275 50$:     ; Check all slots
1256 3276      .ENDC ;; DEFINED LT_XMT_CHECK
1256 3277
54 58 D0 1256 3278      MOVL R8, R4 ; CSB address to r4
    
```

```

080F 30 125D 3279          BSBW  LTSXMIT          ; Transmit the buffer
      1260 3280
      1260 3281 LTSREPLY_DONE: ; All done
3FFE BF BA 1260 3282      POPR  #^M<R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,AP,FP> ; Restore registers
      05 1264 3283          RSB
      1265 3284
      1265 3285
      1265 3286 LTSREPLY_REXMIT:
      1265 3287          ; Retransmit last message
      1265 3288          ;
      1265 3289          ;
      1265 3290          ; R8 = CSB address
      1265 3291          ;
      1265 3292          ;
54 58 D0 1265 3293      MOVL  R8, R4          ; CSB address to R4
      0869 30 1268 3293      BSBW  LTSREXMIT          ; Transmit all waiting buffers
      F3 11 1268 3294      BRB   LTSREPLY_DONE       ; Finish up
      126D 3295

```

```

126D 3297 .SBTTL LT$CREATECSB - Create a CSB
126D 3298
126D 3299 :++ LT$CREATECSB - Create a CSB
126D 3300 :
126D 3301 : Allocate a CSB for use and initialize it partially.
126D 3302 :
126D 3303 : Inputs:
126D 3304 : R4 -> next entry in CSB table beyond the one that is free
126D 3305 : R1 = length of UCB list at end of CSB
126D 3306 :
126D 3307 : Outputs:
126D 3308 : R8 = new CSB address
126D 3309 : R0 = success or failure
126D 3310 :
126D 3311 : CSB table entry set to its address
126D 3312 :
126D 3313 : R1-R2 are destroyed.
126D 3314 :--
126D 3315
126D 3316 LT$CREATECSB:
00000070 8F C0 126D 3317 ADDL #CSB_C_FIXLENGTH+4,R1 ; Add in size of fixed block +
51 1273
1274 3318 ; one 'no access' UCB slot
00000000 53 DD 1274 3319 PUSHL R3 ; Save R3
GF 16 1276 3320 JSB G^EXE$ALONONPAGED ; Get block
53 8ED0 127C 3321 POPL R3 ; Restore R3
4D 50 E9 127F 3322 BLBC R0,10$ ; Got block
3E BB 1282 3323 PUSHR #^M<R1,R2,R3,R4,R5> ; Save registers
00 6E 00 2C 1284 3324 MOVCS #0,(SP),#0,R1,(R2) ; Zero block
62 51 1288
3E BA 128A 3325 POPR #^M<R1,R2,R3,R4,R5> ; Restore registers
08 A2 51 B0 128C 3326 MOVW R1,CSB_W_SIZE(R2) ; Save size of CSB in CSB
0A A2 33 90 1290 3327 MOVB #DYN$C-CDB,- ; Store an unlikely type in the
1294 3328 CSB_B_TYPE(R2) ; the block
50 EDA0 CF DE 1294 3329 MOVAL GHB_L_CSBTABLE, R0 ; Create the slave index from the
50 54 50 C3 1299 3330 SUBL3 R0,-R4, R0 ; address in the CSB table
50 50 04 C6 129D 3331 DIVL2 #4, R0 ; from 1 to n
60 A2 50 90 12A0 3332 MOVB R0, CSB_B_LOCIDN(R2) ; and save it in the CSB
68 A2 50 90 12A4 3333 MOVB R0, CSB_B_INX(R2) ; Save it here also
EEOB CF40 90 12A8 3334 MOVB GHB_B_LOC_CCHK[R0],- ; Obtain the check field for this
61 A2 12AD 3335 CSB_B_LOCIDN(R2) ; CSB slot
12AF 3336
12AF 3337 ; Initialize queue headers
12AF 3338
12AF 3339 MOVAB CSB_Q_XBUFQ(R2), R0 ; Initialize the queue headers for the
50 34 A2 9E 12AF 3340 MOVL R0, -(R0) ; transmit buffers
60 50 DO 12B3 3341 MOVL R0, 4(R0)
04 A0 50 DO 12B6 3342 MOVAB CSB_Q_XWAITQ(R2), R0
50 3C A2 9E 12BA 3343 MOVL R0, -(R0)
60 50 DO 12BE 3344 MOVL R0, 4(R0)
04 A0 50 DO 12C1 3345 MOVL R2, R8
58 52 DO 12C5 3346 MOVL R2, -(R4)
74 52 DO 12C8 3347 MOVZBL #$$$_NORMAL, R0
50 00 8F 9A 12CB 3348 108: RSB
05 12CF 3348

```

```

12D0 3350      .SBTTL  LTSNEW_CIRCUIT - Get Concentrator State Block
12D0 3351      :++
12D0 3352      : LTSNEW_CIRCUIT - Get concentrator state block and initialize it
12D0 3353      :
12D0 3354      : Functional description:
12D0 3355      :
12D0 3356      : Whenever a frame with a zero svci is seen and the circuit state is
12D0 3357      : starting, we attempt to allocate a CSB and start a circuit. First
12D0 3358      : we check for a CSB with the same source address. If any are found
12D0 3359      : and the mvci matches and the state is starting, then we retransmit
12D0 3360      : our starting message for that circuit. If the other things are not
12D0 3361      : true, then the circuit is clobbered to that source since the mvci
12D0 3362      : does not match or the circuit is already running and the concentrator
12D0 3363      : is trying to restart it perhaps because it rebooted.
12D0 3364      :
12D0 3365      : Inputs:
12D0 3366      :   R3 = CXB address
12D0 3367      :   R6 = receive buffer address
12D0 3368      :   R10 = IDB address
12D0 3369      :
12D0 3370      : Outputs:
12D0 3371      :   R6 = receive buffer address
12D0 3372      :   R8 = CSB address
12D0 3373      :   R0,R1,R2,R3,R4,R7 and R9 are modified
12D0 3374      :--
12D0 3375      :
12D0 3376      LTSNEW_CIRCUIT:
12D0 3377      MOVAB  GHB_L_CSBTABLE,R4      : Find free entry in CSB_TABLE
12D0 3378      MOVZBL #LATSC_MAX_CSBS,R2   : Length of table in longwords
12D0 3379      10$:  MOVL  (R4)+,R8        : Get CSB address
12D0 3380      BEQL  30$                    : Ignore unallocated ones for pass
12D0 3381      :
12D0 3382      : If this is the same source address, then we are looking for a duplicate
12D0 3383      : starting message. It will have the same local index and a state of
12D0 3384      : starting. If this is not the case, then just clobber the circuit.
12D0 3385      :
12D0 3386      CMPL  CXBSQ_STATION(R3),-    : NI source address match?
12D0 3387      CSB_Z_DST(R8)                :
12D0 3388      BNEQ  30$                    : Br if not, look for more
12D0 3389      CMPW  CXBSQ_STATION+4(R3),- : Rest match too?
12D0 3390      CSB_Z_DST+4(R8)              :
12D0 3391      BNEQ  30$                    : Br if not, look for more
12D0 3392      CMPW  RCV_W_DSTID(R6),-      : Is this a duplicate starting msg
12D0 3393      CSB_W_LOCID(R8)              : with the same local vc indx/seq?
12D0 3394      BNEQ  20$                    : Br if not, kill circuit
12D0 3395      CMPB  CSB_B_STATE(R8),-     : Is the state still starting?
12D0 3396      #CSB_C_STATE_START          :
12D0 3397      BNEQ  20$                    : Br if not, kill circuit
12D0 3398      MOVL  #2, R0                 : Else, tell higher code to retrans
12D0 3399      RSB                          : our starting message
12D0 3400      :
12D0 3401      20$:  BSBW  LTSCIRCDEAD      : Get rid of UCBs
12D0 3402      MNEGL #1, CSB_Z_DST(R8)    : Form illegal address
12D0 3403      MNEGW #1, CSB_Z_DST+4(R8)  : to stop next compare
12D0 3404      :
12D0 3405      30$:  SOBGTR R2,10$       : Try whole table
12D0 3406      :

```

54	ED64	CF	9E	12D0	3377		
	52	20	9A	12D5	3378		
	58	84	D0	12DB	3379	10\$:	
		2A	13	12DB	3380		
				12DD	3381		
				12DD	3382		
				12DD	3383		
				12DD	3384		
				12DD	3385		
	28	A3	D1	12DD	3386		
	2E	A8		12E0	3387		
		23	12	12E2	3388		
	2C	A3	B1	12E4	3389		
	32	A8		12E7	3390		
		1C	12	12E9	3391		
	02	A6	B1	12EB	3392		
	60	A8		12EE	3393		
		0A	12	12F0	3394		
	08	A8	91	12F2	3395		
		01		12F5	3396		
		04	12	12F6	3397		
	50	02	D0	12F8	3398		
			05	12FB	3399		
				12FC	3400		
		042B	30	12FC	3401	20\$:	
	2E	A8	01	12FF	3402		
	32	A8	01	1303	3403		
				1307	3404		
				1307	3405	30\$:	
	CE	52	F5	130A	3406		

:JAY0002
-1

```

130A 3407
130A 3408
130A 3409
OF EE89 CF E0 130C 3410
54 ED24 CF DE 1310 3411
52 20 9A 1315 3412
84 D5 1318 3413
06 13 131A 3414
F9 52 F5 131C 3415
131F 3416
131F 3417
131F 3418
50 D4 1321 3419
05 1322 3420
1322 3421
1322 3422
1322 3423
1322 3424
1322 3425
50 08 A6 9E 1322 3426
05 02 A0 91 1326 3427
132A 3428
F3 12 132A 3429
132C 3430
132C 3431
132C 3432
132C 3433
132C 3434
06 A0 95 132C 3435
EE 15 132F 3436
1331 3437
1331 3438
1331 3439
1331 3440
1331 3441
1331 3442
1331 3443
1331 3444
55 EDD7 CF 9A 1331 3445
55 04 A0 91 1336 3447
04 1E 133A 3448
55 04 A0 90 133C 3449
51 55 02 78 1340 3450
1344 3451
FF26 30 1344 3452
D5 50 E9 1347 3453
69 A8 55 90 134A 3454
134E 3455
134E 3456
134E 3457
134E 3458
50 08 A6 9E 134E 3459
51 0C A0 9A 1352 3460
52 0D A0 9E 1356 3461
52 51 C0 135A 3462
51 62 9A 135D 3463

```

:JAY0002
:JAY0002
:JAY0002
-3

```

: Search failed, try to find a free CSB
BBS #GHB_STS V SHUT - ; Br if we are shutting down,
GHB_B STATUS 50$ ; don't let connection succeed
MOVAL GHB [ CSBTABLE, R4 ; Look again for a free CSB
MOVZBL #LATSC_MAX_CSBS, R2 ; Set max CSBs
TSTL (R4)+ ; Is this slot free?
BEQL 60$ ; Br if yes, great
SOBGTR R2, 40$ ; Loop til done
: Return resource error
CLRL R0 ; Return failure indicating resource
RSB ; failure
60$:
: Check the protocol version of the incoming message and reject all but
our own.
MOVAB RCV_T_MDATA(R6), R0 ; Address of first slot
CMPB STRT_B_PVER(R0), - ; Check version number
#LATSC_CUR_VER
BNEQ 50$ ; Must be equal
: Check the circuit timer value (in multiples of 10 ms).
We will accept anything in the range 10 to 1270 ms; legal
values are 1 to 127 in this byte field.
TSTB STRT_B_CIR_TIMER(R0) ; Is this field positive and non-zero?
BLEQ 50$ ; Br if not, illegal value
ECO is not checked
R4 -> CSB table address -4
: Compute size needed for UCB list at end of CSB,
size is based on MIN [GHB_B_MAX_SLOTS, STRT_B_MAXSLOTS(R0)]
MOVZBL W^GHB_B_MAX_SLOTS R5 ; Assume ours is MIN value
CMPB STRT_B_MAXSLOTS(R0), R5 ; Is rcvd slot count larger?
BGEQU 70$ ; Br if yes, use ours
MOV B STRT_B_MAXSLOTS(R0), R5 ; Else, use smaller of two
ASHL #2, R5, R1 ; Multiply by 4, and save in R1
70$:
BSBW LT$CREATECSB ; Create a CSB
BLBC R0, 50$ ; Br if resource error
MOV B R5, CSB_B_MAX_SLOTS(R8) ; Save the maximum allowed slot count
: Save the server's name
ASSUME STRT_B_SYS_LEN EQ STRT_T_NODE+GHB$K_NAMELEN
MOVAB RCV_T_MDATA(R6), R0 ; Get address of first slot
MOVZBL STRT_B_NODE_LEN(R0), R1 ; Get length of dest node name
MOVAB STRT_T_NODE(R0), R2 ; Get address of dest node name
ADDL R1, R2 ; Skip dest node name, point to src name
MOVZBL (R2), R1 ; Get length of src (server) name

```

10	51	D1	1360	3464	CMP	R1,#GHBSK_NAMELEN	: Is the name length okay?		
		06	1363	3465	BLEQ	758	: Br if yes, continue		
51	10	9A	1365	3466	MOVZBL	#GHBSK_NAMELEN,R1	: Else, use maximum allowed		
62	51	90	1368	3467	MOV	R1,(R2)	: Make size of name valid		
	51	D6	1368	3468	INCL	R1	: Account for length byte		
	09	BB	136D	3469	PUSHR	#*M<R0,R3>	: Save registers		
62	51	28	136F	3470	MOV	R1,(R2),CSB_B_SERVER(RB)	: Copy the server name		
	1C	AB	1372						
	09	BA	1374	3471	POPR	#*M<R0,R3>	: Restore registers		
			1376	3472					
			1376	3473					
			1376	3474					
			1376	3475					
			1376	3476	MOV	#3600,CSB_W_TIMERSET(RB)	: Assume a default of one hour		
0E10	BF	B0	1376	3476					
	66	AB	137A						
51	07	A0	137C	3477	MOVZBL	STRT_B_KPA_TIMER(R0),R1	: Make timer reset value		
		13	1380	3478	BEQ	808	: Not a good value		
51	05	C4	1382	3479	MULL2	#5,R1	: Look for five times the period,		
	51	D6	1385	3480	INCL	R1	: and one more		
55	51	3C	1387	3481	MOVZWL	R1,R5	: Check for overflow		
55	51	D1	138A	3482	CMP	R1,R5	: Did we overflow?		
	04	12	138D	3483	BNEQ	808	: Br if yes, use default value		
66	AB	51	138F	3484	MOV	R1,CSB_W_TIMERSET(RB)	: Set the value in the CSB		
			1393	3485					
			1393	3486					
			1393	3487					
05D4	BF	B0	1393	3488	MOV	#XMT_C_MAXDATA,-	: Assume ours is maximum allowed		
	6A	AB	1397	3489		CSB_W_MAX_MSGSIZ(RB)	: message size		
	08	A2	1399	3490	SUB	#XMT_C_HDRLEN,-	: Account for header is remotes size		
	60		139B	3491		STRT_B_MSGSIZ(R0)			
	0A	15	139C	3492	BLEQ	908	: Br if not large enough		
	60	B1	139E	3493	CMP	STRT_W_MSGSIZ(R0),-	: Is remotes buffer larger?		
	6A	AB	13A0	3494		CSB_W_MAX_MSGSIZ(RB)			
	04	1E	13A2	3495	BGEQ	908	: Br if yes, use ours		
	60	B0	13A4	3496	MOV	STRT_W_MSGSIZ(R0),-	: Else, use smaller of two		
	6A	AB	13A6	3497		CSB_W_MAX_MSGSIZ(RB)			
			13A8	3498					
			13A8	3499					
			13A8	3500					
5E	AB	04	A6	B0	13A8	3501	MOV	RCV_W_SRCID(R6),CSB_W_REMID(RB)	: In case of retransmit
		5D	AB	94	13AD	3502	CLRB	CSB_B_NUM_SLOTS(RB)	: Mode default is balanced
		52	AB	B4	13B0	3503	CLRW	CSB_W_XMTTMO(R9)	: Not timed state
		66	AB	B0	13B3	3504	MOV	CSB_W_TIMERSET(RB),-	: Set a nonzero timeout value
		50	AB		13B6	3505		CSB_W_TIMEOUT(RB)	: even if the normal value is zero.
		2D	AB	94	13B8	3506	CLRB	CSB_B_REFC(RB)	: No UCBs here
		14	B0		13B8	3507	MOV	#LATSC_PROGRESS,-	: Set for allowing progress
		54	AB		13BD	3508		CSB_W_PROGRESS(RB)	: while we start up.
					13BF	3509			
		28	A3	D0	13BF	3510	MOVL	CXBSQ_STATION(R3),-	: Store remotes's NI address.
		2E	AB		13C2	3511		CSB_Z_DST(RB)	
		2C	A3	B0	13C4	3512	MOV	CXBSQ_STATION+4(R3),-	: ...
		32	AB		13C7	3513		CSB_Z_DST+4(RB)	
		00	90		13C9	3514	MOV	#CSB_C_STATE_HALT,-	: Circuit is halted
		08	AB		13CB	3515		CSB_B_STATE(RB)	
62	AB	01	AE		13CD	3516	MNEGW	#1,CSB_W_XSEQ(RB)	: Initialize transmit sequence
		FF00	BF	B0	13D1	3517	MOV	#<-128!05,CSB_W_RSEQ(RB)	: Initialize receive sequence
		64	AB		13D5				

			13D7	3518	:	
			13D7	3519	:	
			13D7	3520	:	Obtain all necessary transmit buffers for use
			13D7	3521	:	
	58 AB	94	13DA	3522	:	CLRB CSB_B_XMTBSY(R8) : Not busy transmitting
	59 AB	94	13DD	3523	:	CLRB CSB_B_XMTCNT(R8) : No buffers waiting to transmit
54	34 AB	DE	13E1	3524	:	MOVAL CSB_Q_XBUFQ(R8), R4 : Address of the transmit queue head
	55 02	9A	13E4	3525	:	MOVZBL #LAT\$C_XMT_BUFFERS, R5 : Set number of buffers
			13E4	3526	:	
51	0624 8F	3C	13E9	3527	100\$:	MOVZWL #XMT_C_MAXLEN, R1 : Size of the buffers
	00000000 GF	16	13EF	3528	:	JSB G^EXESXLOWONPAGED : Allocate the buffer
	25 50	E9	13F2	3529	:	BLBC R0, 110\$: No such buffer
	08 A2 51	B0	13F6	3530	:	MOVW R1, CXBSW_SIZE(R2) : Store the size,
			13FA	3531	:	ASSUME CXBSB_CODE EQ CXBSB_TYPE+1
	0A A2 1B	B0	13FA	3532	:	MOVW #DYN\$C_CXB, - : the block type
			13FA	3533	:	CXBSB_TYPE(R2) : and indicate not multicast buffer
			13FA	3534	:	
	0C A2 58	D0	13FE	3535	:	MOVL R8, CXBSL_T(CSB(R2)) : Save CSB address in CXB
	2E AB 7D	7D	1401	3536	:	MOVQ CSB_Z_DST(R8), - : Initialize the destination address
	28 A2		1403	3537	:	CXBSB_STATION(R2)
	48 A2 7C	7C	1406	3538	:	CLRQ XMT_T_DATA(R2) : Zero circuit header
	0048 8F	B0	140A	3539	:	MOVW #XMT_T_DATA, - : Store offset to start of data
	18 A2		140C	3540	:	CXBSB_BOFF(R2)
	64 62	0E	140F	3541	:	INSQUE (R2), (R4) : Queue the buffer
	D2 55	F5	1412	3542	:	SOBGTR R5, 100\$: for each of the buffers
			1412	3543	:	
50	00'8F	9A	1416	3544	:	MOVZBL #SSB_NORMAL, R0
		05	1417	3545	:	RSB
			1417	3546	:	
			1417	3547	:	
			1417	3548	:	Resource error return
			1417	3549	:	
54	58	D0	141A	3550	110\$:	MOVL R8, R4 : Move the CSB address for the call
	0003	30	141D	3551	:	BSBW LT\$DEALOCB : Clean up the CSB before we leave
	50	D4	141F	3552	:	CLRL R0 : Return an error
		05	1420	3553	:	RSB

```

1420 3555 .SBTTL LT$DEALOCB - Deallocate a CSB
1420 3556 :++
1420 3557 : LT$DEALOCB
1420 3558 :
1420 3559 : FUNCTIONAL DESCRIPTION:
1420 3560 :
1420 3561 : Deallocate the transmit buffers.
1420 3562 :
1420 3563 : INPUTS:
1420 3564 : R4 = CSB address
1420 3565 :
1420 3566 : OUTPUTS:
1420 3567 : R0-R3 are destroyed
1420 3568 :
1420 3569 :--
1420 3570 :
1420 3571 LT$DEALOCB:
50 51 01 9A 1420 3572 MOVZBL #LAT$C_XMT_BUFFERS-1,R1 : Get number of transmits -1
50 44 A441 D0 1423 3573 10$: MOVL CSB_L_XCB(R4)(R1), R0 : Is an FFI transmit in progress ?
1428 3574 BEQL 20$ : no
142A 3575 CLRL CXB$L_T_CSB(R0) : Delete backpointer to CSB in CXB
50 F3 51 F4 142D 3576 20$: SOBGEQ R1,10$ : Loop if more
50 34 B4 OF 1430 3577 25$: REMQUE @CSB_Q_XBUFQ(R4), R0 : Now dump all the buffers
1434 3578 BVS 30$ : No more buffers
1436 3579 BSBW LT$DEALPOOL : Dump a buffer
1439 3580 BRB 25$ :
143B 3581
50 3C B4 OF 143B 3582 30$: RFMQUE @CSB_Q_XWAITQ(R4), R0 : Now dump all the waiters
143F 3583 BVS 40$ : No more buffers
1441 3584 BSBW LT$DEALPOOL : Dump a buffer
1444 3585 BRB 30$ :
1446 3586
53 68 A4 9A 1446 3587 40$: MOVZBL CSB_B_INX(R4), R3 : Find our place in the table
144A 3588 BNEQ 50$ : Great
144C 3589 DEBUG : Not properly initialized CSB
1452 3590
50 EC61 CF43 96 1452 3591 50$: INCB GHB_B_LOC(CCHK[R3]) : Bump the slot check digit
50 EBD8 CF43 DE 1457 3592 MOVAL GHB_L_CSBTABLE-4[R3],R0 : Address in table
54 60 D1 145D 3593 CMPL (R0),R4 : Check this address with the one
1460 3594 BEQL 60$ : we have.
1462 3595 DEBUG : Table is clobbered or CSB addr bad
1468 3596
60 60 D4 1468 3597 60$: CLRL (R0) : No CSB address here anymore
146A 3598 :
146A 3599 : We will now save this CSB on the OLD CSB queue, but we will
146A 3600 : have to remove an old entry if we exceed our limit.
146A 3601 :
50 54 D0 146A 3602 MOVL R4,R0 : Copy CSB address just in case
146D 3603 : we want to deallocate it
146E 3604 TSTL CSB_Z_LCB+LCB_L_MSG_XMT(R4) : Any information in counters?
1470 3605 BNEQ 70$ : Br if yes, save it
1472 3606 TSTL CSB_Z_LCB+LCB_L_MSG_RCV(R4) : Any information in counters?
1475 3607 BNEQ 70$ : Br if yes, save it
1477 3608 BRW 180$ : Else, nothing interesting here
147A 3609 :
147A 3610 70$: : We will now scan the CSB list for any old CSBs that match this
147A 3611 : one, that way we will only save 1 entry per server.

```

```

51  ECC6 CF 9E 147A 3612      ;
      52 51 D0 147A 3613      MOVAB GHBSQ_OLD_CSBS+GHB_AREA,R1 ; Get listhead of old CSBs
      52 62 D0 147F 3614      MOVL R1,R2 ; ..twice
      51 52 D1 1482 3615 80$: MOVL (R2),R2 ; Get next in queue
      68 13 1485 3616      CMPL R2,R1 ; Are we back to start?
      2E A4 D1 1488 3617      BEQL 100$ ; Br if yes, no more - insert new entry
      2E A2 148A 3618      CMPL CSB_Z_DST(R4),- ; Does the Ethernet address match?
      F1 12 148D 3619      CSB_Z_DST(R2)
      32 A4 B1 148F 3620      BNEQ 80$ ; Br if no, try next
      32 A2 1491 3621      CMPW CSB_Z_DST+4(R4),- ; Full address match?
      EA 12 1494 3622      CSB_Z_DST+4(R2)
      53 1C A4 12 1496 3623      BNEQ 80$ ; Br if no, try next
      1C A2 53 9A 1498 3624      MOVZBL CSB_B_SERVER(R4),R3 ; Get length of server name
      E0 12 149C 3625      CMPB R3,CSB_B_SERVER(R2) ; Lengths match?
      07 BB 14A0 3626      BNEQ 80$ ; Br if no, try next
      1D A4 53 29 14A2 3627      PUSHR #*M<R0,R1,R2> ; Save registers
      1D A2 07 BA 14A4 3628      CMPC3 R3,CSB_T_SERVER(R4),- ; Does the server name match?
      07 07 14A8 3629      CSB_T_SERVER(R2)
      04 12 14AA 3630      POPR #*M<R0,R1,R2>
      14AE 3631      BNEQ 80$ ; Br if no, try next
      14AE 3632      ;
      14AE 3633      ; Take sum of counters and store in CSB @ R2
      14AE 3634      ;
      14AE 3635      ADD_CTR CSB_Z_LCB+LCB_L_MSG_XMT(R4),- ; XMITs
      14AE 3636      CSB_Z_LCB+LCB_L_MSG_XMT(R2)
      14B9 3637      ADD_CTR CSB_Z_LCB+LCB_L_MSG_RCV(R4),- ; RCVs
      14B9 3638      CSB_Z_LCB+LCB_L_MSG_RCV(R2)
      14C4 3639      ADD_CTR CSB_Z_LCB+LCB_L_MSG_REXMT(R4),- ; REXMTs
      14C4 3640      CSB_Z_LCB+LCB_L_MSG_REXMT(R2)
      14CF 3641      ADD_CTR CSB_Z_LCB+LCB_W_SEQ_ERR(R4),- ; SEQ_ERRs
      14CF 3642      CSB_Z_LCB+LCB_W_SEQ_ERR(R2),W
      14DA 3643      ADD_CTR CSB_Z_LCB+LCB_B_INV_MSG(R4),- ; INV_MSGs
      14DA 3644      CSB_Z_LCB+LCB_B_INV_MSG(R2),B
      14E5 3645      ADD_CTR CSB_Z_LCB+LCB_B_INV_SLOT(R4),- ; INV_SLOTS
      14E5 3646      CSB_Z_LCB+LCB_B_INV_SLOT(R2),B
      14 11 14F0 3647 100$: BRB 180$ ; And delete the CSB @ R4
      14F2 3648      ;
      14F2 3649      ; Insert new CSB into list
      14F2 3650      ;
      EC51 DF 64 OE 14F2 3651      INSQUE (R4), @GHBSQ_OLD_CSBS+ - ; Save CSB at end of CSB queue
      14F7 3652      GHB_AREA+4 ; since queue is time ordered
      EC9F CF 97 14F7 3653      DECB GHB_B_OLD_CSBCNT ; Use up one more slot on queue
      EC9F OC 18 14FB 3654      BGEQ 190$ ; Br if still within CSB limits
      50 EC99 CF 96 14FD 3655      INCB GHB_B_OLD_CSBCNT ; Else, restore count
      EC3F DF OF 1501 3656      REMQUE @GHBSQ_OLD_CSBS+ - ; Remove oldest entry
      06D0 30 1506 3657      GHB_AREA,R0
      05 05 1506 3658 180$: BSBW LT$DEALPOOL ; Deallocate the CSB block
      1509 3659 190$: RSB

```

```

150A 3661      .SBTTL LT$CREATEUCB - Create an LT UCB
150A 3662      :++
150A 3663      : LT$CREATEUCB - Create an LT UCB
150A 3664      :
150A 3665      : Create a UCB by cloning it from the master UCB zero and
150A 3666      : initialize it as part of the io database. We can't lock
150A 3667      : the io database for write access, because we aren't a
150A 3668      : process. So we just do it and hope for the best. The
150A 3669      : modifications are such that code scanning the database will
150A 3670      : not be fooled if they are straight forward. It is possible
150A 3671      : to write code that will be confused when a new ucb appears,
150A 3672      : but we don't have any solution other than waiting for the
150A 3673      : the request to come here when the database is not locked.
150A 3674      : That would probably mean that many connection attempts would
150A 3675      : time out.
150A 3676      :
150A 3677      : NB !!
150A 3678      :
150A 3679      : The I/O data base is not locked!!
150A 3680      :
150A 3681      : Inputs:
150A 3682      : none
150A 3683      :
150A 3684      : Outputs:
150A 3685      : R5 = new UCB address
150A 3686      : R0 = success or failure
150A 3687      :
150A 3688      : R1-R4 clobbered
150A 3689      :--
150A 3690      :
150A 3691      LT$CREATEUCB:
150A 3692      MOVL   GHB_L UCBO, R5      ; Get the UCB to clone from
150A 3693      JSB    G^IOCB$CLONE_UCB   ; Clone the UCB
150A 3694      BLBC   R0,90$             ; Exit on error
150A 3695      MOVL   R2,R5              ; Else, copy new UCB address
150A 3696      CLRW   UCBSW_REF(C,R5)    ; No refs yet!
150A 3697      RSB
55    EC72 CF   DO 150A 3692
00000000'GF 16 150F 3693
          06 50  E9 1515 3694
          55 52  DO 1518 3695
          5C A5  B4 151B 3696
          05 151E 3697 90$: RSB

```

```

151F 3699 .SBTTL LT$DESTROYUCB - Destroy an LT UCB
151F 3700 :++
151F 3701 : LT$DESTROYUCB - Destroy an LT UCB
151F 3702 :
151F 3703 : Delete a ucb from the io data base that has been disconnected.
151F 3704 : This routine must be called at high IPL when the I/O database
151F 3705 : is not in use by anyone. We are going to deallocate a block
151F 3706 : and if anyone has its address in a register, we are going
151F 3707 : to go crash with a bang. This situation is not quite the
151F 3708 : same as the creation case where one can be much more cavalier.
151F 3709 :
151F 3710 : Also note, that we must ensure that the LT UCB has been detached
151F 3711 : from any VT UCB, and that the REFC has gone to zero. There is
151F 3712 : a condition where the class driver will not detach a VT UCB
151F 3713 : even though the disconnect entry point is called. This occurs if
151F 3714 : logout has not been run yet, but is about to be run. In this
151F 3715 : case, the class driver will not have detached the LT UCB from the
151F 3716 : VT UCB and we can check for that condition.
151F 3717 :
151F 3718 : NB !!!
151F 3719 : The I/O database is NOT locked for write access, BUT the MUTEX
151F 3720 : is available!
151F 3721 :
151F 3722 : Inputs:
151F 3723 : R5 = UCB to destroy
151F 3724 :
151F 3725 : Outputs:
151F 3726 : R5 = UCB address
151F 3727 : R0-R1 are destroyed.
151F 3728 :--
151F 3729 :
151F 3730 LT$DESTROYUCB:
00C0 C5 55 D1 151F 3731 CMPL R5,UCBSL_TT_LOGUCB(R5) ; Is this UCB ready for deletion?
12 12 1524 3732 BNEQ 50$ ; Br if no, requeue it
SC A5 B5 1526 3733 TSTW UCBSW_REFC(R5) ; All refs gone?
OD 12 1529 3734 BNEQ 50$ ; Br if no, requeue it
152B 3735 SETBIT #UCBSV_DELETEUCB, - ; Else, allow UCB to be deleted
152B 3736 UCBSL_STS(R5)
00000000 GF 16 1530 3737 JSB G^IOC$DELETE_UCB ; Delete the UCB
OE 11 1536 3738 BRB 90$ ; leave
1538 3739
0144 C5 04 90 1538 3740 50$: MOVB #UCBSC_LT_STATE_KILL,- ; Enter the KILL UCB state
153A 3741 UCBSB_LT_STATE(R5)
153D 3742 CLRBIT #UCBSV_LT_DEAD,- ; Clear the dead indicator
153D 3743 UCBSB_LT_LATSIS(R5)
014A 30 1543 3744 BSBW LT$HANGUP_UCB ; And hangup the UCB again!
05 1546 3745 90$: RSB ; Return

```

```

1547 3747 .SBTTL LTSALC_UCB - Allocate New LT UCB
1547 3748 :++
1547 3749 : LTSALC_UCB - Allocate a new LT UCB
1547 3750 :
1547 3751 : Subroutine to select a UCB for a slot.
1547 3752 :
1547 3753 : Inputs:
1547 3754 : R7 = receive buffer slot address
1547 3755 : R8 = CSB address
1547 3756 :
1547 3757 : Outputs:
1547 3758 : R0 = success or failure
1547 3759 : R5 = UCB address
1547 3760 : R7 = receive buffer slot address
1547 3761 : R8 = CSB address
1547 3762 : R1,R2,R3,R4 modified
1547 3763 :--
1547 3764
1547 3765 LTSALC_UCB:
50 D4 1547 3766 CLRL R0 ; Return failure if no UCB
1549 3767
1549 3768 ::88 *TEMP* concentrator bug
1549 3769
54 6C AB DE 1549 3770 MOVAL CSB_L_UCBLST(R8), R4 ; Search the list for this CSB
53 54 DO 154D 3771 MOVL R4, R3 ; Save the address
51 69 AB 9A 1550 3772 MOVZBL CSB_B_MAX_SLOTS(R8), R1 ; Get maximum in list
55 84 DO 1554 3773 1$: MOVL (R4)+, R5 ; Look in the table
OB 13 1557 3774 BEQL 2$ ; A free slot, my word
01 A7 91 1559 3775 CMPB SLT_B_SRCID(R7), - ; Is this the same terminal calling
0140 C5 155C 3776 UCBSB_LT_REMID(R5) ; again?
03 12 155F 3777 BNEQ 2$ ; No, then continue
010C 31 1561 3778 BRW 95$ ; Yes, then setup message again
ED 51 F5 1564 3779 2$: SOBGTR R1, 1$ ; look again
1567 3780
1567 3781 ::88 *TEMP* concentrator bug
1567 3782
54 6C AB DE 1567 3783 MOVAL CSB_L_UCBLST(R8), R4 ; Search the list for this CSB
53 54 DO 156B 3784 MOVL R4, R3 ; save the address
51 69 AB 9A 156E 3785 MOVZBL CSB_B_MAX_SLOTS(R8), R1 ; Get maximum in list
55 84 DO 1572 3786 10$: MOVL (R4)+, R5 ; Look in the table
06 13 1575 3787 BEQL 20$ ; A free slot, my word
F8 51 F5 1577 3788 SOBGTR R1, 10$ ; look again
00F2 31 157A 3789 15$: BRW 90$ ; None to be had here, so fail
157D 3790
02 A7 91 157D 3791 20$: CMPB SLT_B_COUNT(R7), - ; Is slot big enough?
05 1580 3792 #<SLT_B_DST_NAMLEN-SLT_f_DATA>+2
0A 18 1581 3793 BGEQ 30$ ; Br if yes, okay
ED 11 1583 3794 INC_CTR CSB_Z_LCB+LCB_B_INV_SLOT(R8) ; Else, say invalid slot
158B 3795 BRB 15$ ; And exit
158D 3796
54 54 DD 158D 3797 30$: PUSHL R4 ; Save the address in table
54 53 C2 158F 3798 SUBL2 R3, R4 ; Compute the index for this CSB
54 04 C6 1592 3800 DIVL2 #4, R4 ; as a small integer
54 54 DD 1595 3801 PUSHL R4 ; Save the index too
FF70 30 1597 3802 BSBW LTSCREATEUCB ; Create a raw lt UCB
54 8ED0 159A 3803 POPL R4 ; Obtain the index

```



```

53 8ED0 159D 3804      POPL      R3      ; and its address in CSB table
      15A0 3805      ;
      D7 50  E9 15A0 3806      BLBC      R0, 15$      ; Do we have one?, nope, too bad
      73 55  D0 15A3 3807      MOVL      R5, -(R3)      ; Store its address in CSB table
      F4F2 30 15A6 3808      BSBW      LT$UCB INIT      ; Initialize the ucb
0141 C5 54 90 15A9 3809      MOVB      R4, UCBSB_LT_LOCID(R5) ; Set local slot index
      01  A7 90 15AE 3810      MOVB      SLT_B_SRCID(R7), -      ; Load remote circuit id
      0140 C5 15B1 3811      UCBSB_LT_REMID(R5)      ;
      04 00  EF 15B4 3812      EXTZV     #0, #4, SLT_B_CRED(R7), R0 ; Get credit total extended
      50 03  A7 15B7      ;
      50 01  81 15BA 3813      ADDB3     #1, R0, UCBSB_LT_TCRED(R5); Give ourself an extra credit for
      0146 C5 15BD      ;
      15C0 3814      ; sending the start slot
      91 8F 90 15C0 3815      MOVB      #SLT_C_STR SLOTS@4!LATSC_MAX_RCRED, - ; Let slot type and credits
0148 C5 15C3 3816      UCBSB_LT_RCRED(R5)      ;
      01 01  8E 15C6 3817      MNEGB     #LATSC_MAX_RCRED, -      ; Set credits extended to remote end
0147 C5 15C8 3818      UCBSB_LT_XCRED(R5)      ; kept as negative value
0082 C5  B4 15CB 3819      CLRW     UCBSW_ERRCNT(R5)      ; Reset the error counter that
      AB 15CF 3820      ; we bump as we shut down.
      15D0 3821      BISW      #<UCBSM INT!-      ; Expecting interrupts & online
      15D0 3822      UCBSM_ONLINE>, -      ;
0134 C5 58  D0 15D3 3823      MOVL      R8, UCBSL_LT_CSB(R5)      ; Load CSB address
      2D  AB 96 15DB 3824      INCB     CSB_B_REFC(R8)      ; One more UCB here
      15DB 3825      ;
      15DB 3826      ;
      15DB 3827      ; Compute maximum allowed slot size
      15DB 3828      ;
      FF 8F 90 15DB 3829      MOVB      #LATSC_MAX_SLOTSIZ, -      ; Assume ours is the maximum
0149 C5 15DE 3830      UCBSB_LT_SLOTSZ(R5)      ; slot size
      06  A7 91 15E1 3831      CMPB     SLT_B_DT_SLTSIZ(R7), -      ; Is remotes size smaller?
0149 C5 15E4 3832      UCBSB_LT_SLOTSZ(R5)      ;
      06 06  1E 15E7 3833      BGEQU     40$      ; Br if no, use ours
      06  A7 90 15E9 3834      MOVB      SLT_B_DT_SLTSIZ(R7), -      ; Else, take remotes
0149 C5 15EC 3835      UCBSB_LT_SLOTSZ(R5)      ;
      0144 C5 02 90 15EF 3836 40$: MOVB      #UCBSC_LT_STATE_RUN, -      ; Set the initial state to RUN
      05 05 90 15F1 3837      UCBSB_LT_STATE(R5)      ;
      014B C5 05 90 15F4 3838      MOVB      #GHB_C_STRT_SLOTL, -      ; Build a start slot
      55  DD 15F6 3839      UCBSB_LT_CORC(R5)      ;
      EE 05 28 15F9 3840      PUSHL    R5      ; Save UCB address
      014C C5 05 28 15FB 3841      MOVCS    #GHB_C_STRT_SLOTL, -      ; Make it look like output data
      EE  CF 15FD 3842      GHB_T_STRT_SLOT, -      ;
      014C C5 55 8ED0 1600 3843      UCBSB_LT_CBUF(R5)      ;
      01 01 88 1603 3844      POPL     R5      ; Save UCB address
0142 C5 1606 3845      BISB     #UCBSM_LT_DATA, -      ; Force data through output slot
      1608 3846      UCBSB_LT_DATAW(R5)      ;
      1608 3847      ;
      1608 3848      ; Check for the disable login flag
      1608 3849      ;
      51 07  A7 9E 160B 3850      MOVAB     SLT_B_DST_NAMLEN(R7), R1 ; Skip to dest. slot name field
      50 81 9A 160F 3851      MOVZBL   (R1)+, R0      ; Get size of dest. name field
      51 50  C0 1612 3852      ADDL     R0, R1      ; Skip the dest. name field
      50 81 9A 1615 3853      MOVZBL   (R1)+, R0      ; Get size of src name field
      51 50  C0 1618 3854      ADDL     R0, R1      ; Skip the src name field
      50 04  A7 9E 161B 3855      MOVAB     SLT_T_DATA(R7), R0      ; Compute size of slot skipped
      52 51 50  C3 161F 3856      SUBL3    R0, R1, R2      ;
      50 02  A7 9A 1623 3857      MOVZBL   SLT_B_COUNT(R7), R0      ; Get size of entire slot
      50 52  C2 1627 3858      SUBL     R2, R0      ; Compute bytes remaining

```

```

52 30 15 162A 3859 BLEQ 70$ ; Br if no more data, do login
    81 9A 162C 3860 60$: MOVZBL (R1)+,R2 ; Get parameter code
    2B 13 162F 3861 BEQL 70$ ; Br if none, assume auto-login
    50 D7 1631 3862 DECL R0 ; One less byte left
    27 15 1633 3863 BLEQ 70$ ; Br if no more data, do login
53 81 9A 1635 3864 MOVZBL (R1)+,R3 ; Get size of parameter
50 53 C2 1638 3865 SUBL R3,R0 ; Get remaining byte count
    1F 15 163B 3866 BLEQ 70$ ; Br if no more data, do login
54 51 D0 163D 3867 MOVL R1,R4 ; Save current parameter address
51 53 C0 1640 3868 ADDL R3,R1 ; Skip to next parameter
01 52 D1 1643 3869 CMPL R2,#SLT_C_START_FLAG ; Is this the start slot flag byte?
    E4 12 1646 3870 BNEQ 60$ ; Br if no, look for next
OC 64 02 E1 1648 3871 BBC #SLT_FLAG_V_BIND,(R4),65$ ; Br if not a bind request
    164C 3872 CLRBIT #TT2$V_HANGUP,- ; Else, set terminal to NOHANGUP
    164C 3873 UCBSL TT_DECHA1(R5) ; on logou!
    1652 3874 SETBIT #UCBS$V_LT_BOUND,- ; Don't delete UCB after STOP slot
    1652 3875 UCBSB [T_CATSTS(R5) ; ... only if we must KILL the UCB:
OF 64 01 E0 1658 3876 65$: BBS #SLT_FLAG_V_LOGIN,(R4),80$ ; Br if login is disabled
    165C 3877 ;
    165C 3878 ; Ask the class driver to log us in
    165C 3879 ;
51 0114 C5 D0 165C 3880 70$: MOVL UCBSL TT_CLASS(R5), R1 ; Class dispatch vector
00002000 8F DU 1661 3881 MOVL #1@13, R3 ; Framing error with zero byte
    53 1667
    14 B1 16 1668 3882 JSB @CLASS_READERROR(R1) ; Cause read error
50 00'8F 9A 1668 3883 80$: MOVZBL #SS$_NORMAL, R0 ; Do we have a UCB for you.
    05 166F 3884 90$: RSB
    1670 3885
    1670 3886 ;:88
    1670 3887
    1670 3888 95$: SETBIT #GHBSV_REPCREATE,- ; Set mask bit
    1670 3889 GHBSL_PROTOMASK+GHB_AREA ; Repeat create of slot
    FF6C 31 1676 3890 INC_CTR GHBSL_PROTOCOL+GHB_AREA ; Increment protocol error counter
    1680 3891 BRW 40$
    1683 3892 ;:88
*TEMP* concentrator bug

```

```

1683 3894 .SBTTL LASHANGUP_UCB - Cause Terminal Hangup On UCB
1683 3895 :++
1683 3896 : LASHANGUP_UCB - Hangup an LT UCB
1683 3897 :
1683 3898 : Cause all the hangup effects on a terminal device. This should cause
1683 3899 : process rundown and deallocation of all channels to the UCB.
1683 3900 :
1683 3901 : Inputs:
1683 3902 :     R5 = UCB address
1683 3903 :
1683 3904 : Outputs:
1683 3905 :     none.
1683 3906 :
1683 3907 :--
1683 3908 .ENABL LSB
1683 3909 LASHANGUP_UCB_NOW:
1683 3910 :
1683 3911 : Dispatch on UCB (slot) state
1683 3912 :
1683 3913 $DISPATCH UCBSB_LT_STATE(R5),TYPE=B,-
1683 3914 <- : state : action
1683 3915 <UCBSC_LT_STATE_STOP LASKILLUCB>,- : Already disconnected,
1683 3916 >
1688 3917 :
1688 3918 : Else
1688 3919 :
1688 3920 MOVB #UCBSC_LT_STATE_KILL,- : Enter the KILL UCB state
168D 3921 UCBSB_LT_STATE(R5)
1690 3922 :
1690 3923 LASHANGUP_UCB:
1690 3924 POSHR #*M<R0,R1,R2,R3,R4> : Save registers
1692 3925 BICW #UCBSM_INT,UCBSL_STS(R5) : No interrupt expected (allows
1696 3926 : STARTIO to be called again
1696 3927 : to flush data).
1696 3928 SETBIT #UCBSV_LT_HANGUP,- : Indicate that we're shutting
1696 3929 UCBSB_LT_LATSTS(R5) : down
169C 3930 MOVL UCBSL_TT_CLASS(R5),R0 : Class driver vector address
16A1 3931 JSB @CLASS_DISCONNECT(R0) : Call routine
16A4 3932 POPR #*M<R0,R1,R2,R3,R4> : Restore the regs
16A6 3933 90$: RSB
16A7 3934 :
16A7 3935 .DSABL LSB
    
```

```

16A7 3937 .SBTTL LT$DISCONNECT - Port Disconnect
16A7 3938 :++
16A7 3939 : LT$DISCONNECT - Disconnect the terminal Port driver
16A7 3940 :
16A7 3941 : Port driver entry to disconnect a terminal. called when
16A7 3942 : Last reference is gone. here we free the slot in the buffer
16A7 3943 : And bump the sequence number. also we set the UCB offline.
16A7 3944 :
16A7 3945 : Inputs:
16A7 3946 : R5 = UCB address
16A7 3947 : R0 -> lbc = delete UCB
16A7 3948 : R0 -> lbs = do not delete UCB
16A7 3949 :
16A7 3950 : Outputs:
16A7 3951 : R5 = UCB address
16A7 3952 : All other registers are preserved.
16A7 3953 :
16A7 3954 : Implicit inputs:
16A7 3955 : UCBSB_LT_STATE = KILL -> kill UCB immediately.
16A7 3956 : STOP -> send stop slot, then kill UCB.
16A7 3957 : ??? -> all others, set to STOP state.
16A7 3958 :
16A7 3959 :--
16A7 3960 :
16A7 3961 : LT$DISCONNECT:
16A7 3962 :
16A7 3963 : Do not touch the UCB if it is the template UCBO
16A7 3964 :
16A7 3965 : CMPL R5,GMB_L_UCBO ; Is this the template ucb ?
EAD4 CF 55 D1 16A7 3965 : BEQL 90$ ; Br if yes, don't touch it!
27 13 16AC 3966 :
16AE 3967 :
16AE 3968 : Dispatch on our UCB state
16AE 3969 :
16AE 3970 : $DISPATCH UCBSB_LT_STATE(R5),TYPE=B,-
16AE 3971 : <- : state : action
16AE 3972 : <UCBSB_LT_STATE_KILL 50$>,- ; KILLED, just kill the UCB now
16AE 3973 : <UCBSB_LT_STATE_STOP 90$>,- ; STOPed, we've been here before
16AE 3974 : >
16B8 3975 : ; ALL other states.
16B8 3976 :
16B8 3977 : If the circuit is gone, or there is no CSB for the UCB
16B8 3978 : then just delete the UCB immediately
16B8 3979 :
0134 C5 D5 16B8 3980 : fSTL UCBSL_LT_CSB(R5) ; Do we have a circuit?
14 13 16BC 3981 : BEQL 50$ ; Br if none
16BE 3982 :
16BE 3983 : Conditionally skip deallocation of UCB, on 'hangup'.
16BE 3984 :
14 50 E8 16BE 3985 : BLBS R0,90$ ; Br if not to 'hangup' UCB
10 AA 16C1 .1 ; No longer ONLINE
64 A5 16C3 .2 ; so no one can use again
03 90 16C5 3986 : MOVB #UCBSB_LT_STATE_STOP,- ; Else, set STOP state,
0144 C5 16C7 3987 : UCBSB_LT_STATE(R5) ; to send STOP slot
16CA 3988 :
16CA 3989 : Make sure that no data is given to CLASS driver, after
16CA 3990 : we've been here!
16CA 3991 :

```

: JAY0001
: JAY0001

		16CA	3992		SETBIT	#UCBSV_LT_HANGUP,-	:	Indicate that we're shutting
		16CA	3993			UCBSB_LT_LATSTS(R5)	:	down
03	11	16D0	3994		BRB	90\$:	Delete UCB when STOP is sent!
		16D2	3995				:	
0001	30	16D2	3996	50\$:	BSBW	LT&KILLUCB	:	Get rid of the UCB now
		16D5	3997				:	
	05	16D5	3998	90\$:	RSB		:	Done
		16D6	3999				:	

```

16D6 4001 .SBTTL LTSKILLUCB - Delete a dried up UCB
16D6 4002 :++
16D6 4003 : LTSKILLUCB - Delete an LT UCB
16D6 4004 :
16D6 4005 : The UCB is all dried up one way or another. So we must delete it if
16D6 4006 : possible, else put it on the deadlink list to be deleted on the next
16D6 4007 : timer tick.
16D6 4008 :
16D6 4009 : Inputs:
16D6 4010 : R5 = UCB address
16D6 4011 :
16D6 4012 : Outputs:
16D6 4013 : R5 = UCB address
16D6 4014 : All other registers preserved.
16D6 4015 :
16D6 4016 :--
16D6 4017 :
16D6 4018 LTSKILLUCB:
1F BB 16D6 4019 PUSH R0,R1,R2,R3,R4 : Save some volatile regs
AA 16D8 4020 BICW #<UCBSM_INT! - : No interrupt expected
64 AS 12 16D9 4021 UCBSM_ONLINE>,- : Not online, but free now
F57A 30 16D9 4022 UCBSL_STS(R5) : For use again
0027 30 16DC 4023 LTSFLUSH DATA : Flush all output data
01 E3 16DF 4024 LTSIDELINEUCB : Set the UCB aside from the circuit
12 0143 C5 16E2 4025 #UCBSV_LT_DEAD,- : If we already linked UCB to deadlist
16E4 4026 UCBSB [T [ATSTS(R5),20S : then we have made an error
16E8 4027 SETBIT #GHB$V_REPDISC,- : KILLUCB called twice
16E8 4028 GHB$L_PROTOCOL+GHB_AREA :
OC 11 16EE 4029 INC_CTR GHB$L_PROTOCOL+GHB_AREA : Increment protocol error counter
16FB 4030 BRB 90S :
16FA 4031 20S: :
16FA 4032 : We must destroy the UCB (not called from $DASSGN, so queue the
16FA 4033 : UCB to the deadlist so it can be deallocated by the timer service.
16FA 4034 : This is because the CLASS driver still uses the UCB even if we
16FA 4035 : deallocate it!
16FA 4036 :
EAB6 CF DO 16FA 4037 MOVL GHB_L_DEADLINK,- : Link UCB into chain of dead UCBs
0138 C5 16FE 4038 UCBSL_LT_DEADLINK(R5) : for disposal at timeout
EA7E CF 55 DO 1701 4039 MOVL R5, GHB [ _DEADLINK :
1F BA 1706 4040 90S: POPR #*M<R0,R1,R2,R3,R4> : Restore the regs
05 OS 1708 4041 RSB

```

```

1709 4043 .SBTTL LTSSIDELINEUCB - Set a UCB aside
1709 4044 :++
1709 4045 : LTSSIDELINEUCB
1709 4046 :
1709 4047 : Functional description:
1709 4048 :
1709 4049 : Set a UCB aside to dry up on its own. Disconnect it from a running
1709 4050 : circuit.
1709 4051 :
1709 4052 : Inputs:
1709 4053 :     R5 = UCB address
1709 4054 :
1709 4055 : Outputs:
1709 4056 :     none
1709 4057 :     All registers preserved.
1709 4058 :
1709 4059 :--
1709 4060
1709 4061 LTSSIDELINEUCB:
54 0134 18 BB 1709 4062 PUSH  #^M<R3,R4> : Save registers
68 A443 15 DC 1709 4063 MOVL  UCBSL_LT_CSB(R5), R4 : Get the CSB address
53 0141 13 13 1710 4064 BEQL  10$ : None, this is already done
0134 C5 9A 1712 4065 MOVZBL UCBSB_LT_LOCID(R5), R3 : Use the local slot index in UCB
68 A443 D4 1717 4066 CLRL  CSB_L_UCBLST-4(R4)[R3] : to adjust state of UCB list in CSB
0134 C5 D4 1718 4067 CLRL  UCBSL_LT_CSB(R5) : Remember we have been here
2D A4 97 171F 4068 DECB  CSB_B_REFC(R4) : One less UCB on this circuit
03 12 1722 4069 BNEQ  10$ : We have more UCBs here
0044 30 1724 4070 BSBW  LT$STOPCIRC : Then don't wait for return traffic
18 BA 1727 4071 10$: POPR  #^M<R3,R4> : Restore registers
05 05 1729 4072 RSB : Return to caller

```

```

172A 4074 .SBTTL LT$CIRCDEAD - Declare circuit dead
172A 4075 :++
172A 4076 : LT$CIRCDEAD - Declare circuit dead.
172A 4077 :
172A 4078 : Functional description:
172A 4079 :
172A 4080 : Log off all the processes on each UCB and waiting for them
172A 4081 : to free their UCBs. This will cause the CSB to go free too.
172A 4082 : If this CSB has no ucb's attached, then the CSB is just cleaned up.
172A 4083 :
172A 4084 : Inputs:
172A 4085 : RB = CSB address
172A 4086 :
172A 4087 : Outputs:
172A 4088 : All UCB's hungup.
172A 4089 :
172A 4090 :--
172A 4091 :
172A 4092 LT$CIRCDEAD:
03F 8F BB 172A 4093 PUSHR #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9> ; Save registers
2D AB 95 172E 4094 TSTB CSB_B_REFc(R8) ; Do we have any UCBs?
57 69 AB 9A 1731 4095 BEQL 100$ ; Br if no, so deallocate CSB
56 6C AB DE 1737 4096 MOVZBL CSB_B_MAX_SLOTS(R8), R7 ; Else, number of UCBs to scan
55 86 D0 173B 4097 MOVAL CSB_L_UCB[ST(R8), R6] ; List of the UCBs
FF40 03 18 173E 4098 20$: MOVL (R6)+, R5 ; Get next UCB address
F5 57 FS 1740 4099 BGEQ 60$ ; Not system address
03FF 8F BA 1746 4100 BSBW LT$HANGUP_UCB_NOW ; Hangup the process on this UCB
05 174A 4101 60$: SOBGTR R7, 20$ ; For all the units here
174B 4102
174B 4103 90$: POPR #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9> ; Restore registers
174B 4104 RSB ; Return to caller
174B 4105
174B 4106 ; Refc has gone to zero, send a stop message
174B 4107
54 58 D0 174B 4108 100$: MOVL R8, R4 ; Setup CSB address
18 10 174E 4109 BSBW LT$STOPCIRC ; Send a stop message on circ
F4 11 1750 4110 BRB 90$ ; And exit
1752 4111

```



```

1752 4113 .SBTTL LT$STOPCIRC - Transmit stop message
1752 4114 :++
1752 4115 : LT$STOPCIRC - Transmit stop message
1752 4116 :
1752 4117 : Functional description:
1752 4118 :
1752 4119 : Format and transmit stop message
1752 4120 :
1752 4121 : Inputs
1752 4122 :     R4 = CSB address
1752 4123 :     CSB_L_STOPREASON      addr of stop reason block
1752 4124 :
1752 4125 : Outputs:
1752 4126 :     none
1752 4127 :
1752 4128 :--
1752 4129 :
1752 4130 DEFREASN:
1752 4131 .ASCIC /Node terminated circuit./

20 65 64 6F 4E 00' 1752 4132
6E 69 6D 72 65 74 1752 4133
63 20 64 65 74 61 1752 4134
74 69 75 63 72 69 1752 4135
2E 1752 4136
18 1752 4137
176B 4132
176B 4133 LT$STOPCIRC:
03FF 8F BB 176B 4134 PUSH R4,R5,R6,R7,R8,R9 ; Save registers
58 54 D0 176F 4135 MOV R4,R8 ; Set CSB addr in r8
2D AB 95 1772 4136 TSTB CSB_B_REFC(R8) ; We are not supposed to send
06 13 1775 4137 BEQL 10$ ; stop messages with
1777 4138 DEBUG ; outstanding ucbs on circ
0B AB 91 177D 4139 10$: CMPB CSB_B_STATE(R8),- ; Have we been here before?
00 ; #CSB_C_STATE_HALT
03 12 1780 4140 BNEQ 15$ ; Br if no
008A 31 1783 4141 BRW 90$ ; Else, exit
66 AB 01 B0 1786 4142 15$: MOVW #1,CSB_W_TIMRESET(R8) ; Set timers for quick timeout
50 AB 01 B0 178A 4143 MOVW #1,CSB_W_TIMEOUT(R8)
00 90 178E 4144 MOVB #CSB_C_STATE_HALT,- ; and set state to halted
0B AB 1790 4145 CSB_B_STATE(R8)
60 AB B4 1792 4146 CLRW CSB_W_LOCID(R8) ; Zero the local ID
50 34 AB 7E 1795 4147 MOVAQ CSB_Q_XBUFQ(R8), R0 ; Transmit buffer queue
57 00 B0 0F 1799 4148 REMQUE @(R0), R7 ; Do we have a free buffer?
0F 1C 179D 4149 BVC 30$ ; Yep, got it
57 3C B8 0F 179F 4150 REMQIE @CSB_Q_XWAITQ(R8), R7 ; Get a waiting bfr
06 1C 17A3 4151 BVC 20$ ; Ok, so use it
17A5 4152 DEBUG ; All buffers gone????
59 AB 97 17AB 4153 20$: DECB CSB_B_XMTCNT(R8) ; One less buffer on waitq
60 67 0E 17AE 4154 30$: INSQUE (R7), (R0) ; Make it a free buffer
59 57 D0 17B1 4155 MOVL R7, R9 ; Save bfr header adr
57 48 A7 9E 17B4 4156 MOVAB XMT_T_DATA(R7), R7 ; Stop message data
08 90 17B8 4157 MOVB #MTP_C_HALT@FLAG_V_MTYPE,- ; Send a HALT message
5C AB 17BA 4158 CSB_B_FCAG(R8)
5D AB 94 17BC 4160 CLRB CSB_B_NUM_SLOTS(R8) ; No slot data
53 50 A9 9E 17BF 4161 MOVAB XMT_T_MDATA(R9), R3 ; Set default end of data
83 94 17C3 4162 CLRB (R3)+ ; Set default reason code
50 8A AF 9E 17C5 4163 MOVAB DEFREASN, R0 ; Store a default text string
51 60 9A 17C9 4164 MOVZBL (R0), R1 ; the count

```

63	60	51	D6	17CC	4165	INCL	R1	:	including the count
		51	28	17CE	4166	MOVCL	R1, (R0), (R3)	:	move count and string
56	4C	A8	D0	17D2	4167	MOVL	CSB_L_STOPREASON(R8), R6	:	Do we have a reason
		29	13	17D6	4168	BEQL	40\$:	No reason
50	02	A6	9A	17D8	4169	MOVZBL	2(R6), R0	:	Set the reason bit
				17DC	4170	SETBIT	R0, GHB\$L_PROTOMASK+GHB_AREA	:	and count the error
				17E2	4171	INC_CTR	GHB\$L_PROTOCOL+GHB_AREA	:	as a halt
50	A9	03	A6	90	17EC	MOVBL	3(R6), -	:	Store the reason code
				17F1	4173		STOP_B_RCODE+XMT_T_MDATA(R9)	:	
	50	66	3C	17F1	4174	MOVZWL	(R6), R0	:	Offset to the string
	50	56	C0	17F4	4175	ADDL	R6, R0	:	Set the address
	51	60	9A	17F7	4176	MOVZBL	(R0), R1	:	Get the count
		51	D6	17FA	4177	INCL	R1	:	and set the count
	60	51	28	17FC	4178	MOVCL	R1, (R0), -	:	copy the counted string
	51	A9		17FF				:	
				1801	4179		STOP_B_RLEN+XMT_T_MDATA(R9)	:	Leave r3 pointing beyond
				1801	4180	40\$:			
1C	A9	53	D0	1801	4181	MOVL	R3, CXB\$L_T_ENDADR(R9)	:	and store in buffer
	54	58	D0	1805	4182	MOVL	R8, R4	:	copy CSB address
		0264	30	1808	4183	BSBW	LT\$XMIT	:	and transmit the msg
	03FF	8F	BA	180B	4184	50\$:	POP_R	#^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9>	: Restore registers
			05	180F	4185	RSB		:	Return to caller
				1810	4186				
				1810	4187	:			Stop message sent once
				1810	4188	:			Just wipe CSB now
				1810	4189				
54	58	D0	1810	4190	90\$:	MOVL	R8, R4	:	CSB address to R4
	FCOA	30	1813	4191		BSBW	LT\$DEALOCB	:	Deallocate CSB to pool
	F3	11	1816	4192		BRB	50\$:	And exit
			1818	4193					

```

1818 4195 .SBTTL LT$SET_TIMER - Start Timer
1818 4196 :++
1818 4197 : LT$SET_TIMER - Start the timer
1818 4198 :
1818 4199 : Functional description:
1818 4200 :
1818 4201 : Start timer ticking to scan the CSB's for dead circuits
1818 4202 :
1818 4203 : Inputs:
1818 4204 : R0-R2 available
1818 4205 :
1818 4206 : Implicit Inputs:
1818 4207 :
1818 4208 : IPL = LAT$C_IPL (8)
1818 4209 :
1818 4210 : Outputs:
1818 4211 : R0-R2 are destroyed.
1818 4212 :
1818 4213 :--
1818 4214 :
1818 4215 LT$SET_TIMER:
50 E8C0 CF 9E 1818 4216 MOVAB GHB G_TQE, R0 ; Address of timer queue element
1F E976 CF 01 E2 181D 4217 BBSS #GHB_STS_V_ACTIVE,- ; Br if timer already active
000D'CF 04 88 181F 4218 GHB_B_STATUS,10$ ; and mark as active
08 A0 30 B0 1823 4219 BISB #DPT$M_NOUNLOAD,DPT$TAB+DPT$B_FLAGS, ; Make driver unloadable
0A A0 90 1828 4220 MOVW #TQESC_LENGTH,- ; Setup timer queue entry block
55 50 D0 182A 4221 TQESW SIZE(R0) ; For its first use
0F 90 182C 4222 MOVB #DYN$C_TQE,- ; With both the size and the type
0A A0 182E 4223 TQESB_TYPE(R0) ; Code.
38 BB 1830 4224 PUSHR #*M<R3,R4,R5> ; Save the fork context
55 50 D0 1832 4225 MOVL R0, R5 ; Use TQE as a fork block
1835 4226 :
1835 4227 : Use the TQE as a fork block to start the timer. This is because the
1835 4228 : timer must be started at or below QUEUEAST to synchronize access to the
1835 4229 : TIMER QUEUE.
1835 4230 :
1835 4231 MOVB #IPL$_QUEUEAST,- ; Set delivery IPL
0B A5 90 1837 4232 FKBSB_FIPL(R5) ; of fork block
08 10 1839 4233 BSBB 40$ ; Create fork process
E8F8 CF 01 CE 183B 4234 MNEGL #1,GHBSL_TIM_ACT+GHB_AREA ; Mark the timer as active for LATCP
38 BA 1840 4235 POPR #*M<R3,R4,R5> ; Restore our context
05 1842 4236 10$: RSB ; And return
1843 4237 :
1843 4238 :
1843 4239 : Fork process to start the timer. We need to go to synch from
1843 4240 : fork ipl to start the timer. We do this by going through
1843 4241 : Queueast IPL to be safe - using the TQE as a fork block.
1843 4242 :
1843 4243 :
00000000'GF 16 1843 4244 40$: JSB G*EXES$FORK ; Fork to start the timer ticking
1849 4245 DSBINT #IPL$_SYNCH ; Raise to synch
188C'CF 9E 184F 4246 MOVAB W*LT$TICK,- ; Address of tick routine
0C A5 1853 4247 TQESL FPC(R5) ; is the fork pc
00000000'GF D0 1855 4248 MOVL G*TTY$GL_DELTA,- ; The tick time is from tty sysgen
20 A5 185B 4249 TQESQ DELTA(R5) ; parameter for modem control
05 90 185D 4250 MOVB #TQESC_SSREPT,- ; Repeating tick
0B A5 185F 4251 TQESB_RQTYPE(R5) ;

```



```

52 AB 02 B0 18D1 4316      MOVW    #LATSC_HOST_TIMER,CSB_W_XMTTMO(R8) ; Reset the timeout
          18D5 4317      :
          18D5 4318      : Timeout circuit if needed
          18D5 4319      :
          50 AB 07 18D5 4320 60$:  DECW    CSB_W_TIMEOUT(R8)          ; Check the circuit timeout
          12      12 18DB 4321      BNEQ    90$                    ; Br if not expired
          FE43 30 18DA 4322      INC     CTR GHB$CIRCDOWN+GHB_AREA ; Count circ down event
          66 AB 00 18E4 4323      BSBW   LT$CIRCDEAD             ; Declare the circuit dead
          50 AB 00 18E7 4324      MOVW    CSB_W_TIMRESET(R8),- ; Reset the timeout to wait
          B5 5B F5 18EA 4325      CSB_W_TIMEOUT(R8) ; a while.
          18EC 4326 90$:  SOBGTR  R11, 20$ ; Loop till we are done
          18EF 4327      :
          18EF 4328      :
          18EF 4329      :
          18EF 4330      :
          18EF 4331      :
          18EF 4332      :
          18EF 4333      :
          00000000'GF DE 18EF 4334 100$: MOVAL   G^IOCSGL_MUTEX, R0 ; Is anyone owning the io data
          50      50 18F5          :
          FFFF 8F 60 B1 18F6 4335      CMPW    MTX$W_OWCNT(R0), #-1 ; base? If so, then we must
          1A      12 18FB 4336      BNEQ    130$                    ; try again later.
          54  E883 CF D0 18FD 4337      MOVL   GHB_L_DEADLINK, R4 ; Good we own it then
          E87E CF D4 1902 4338      CLRL   GHB_L_DEADLINK ; Reset list pointer
          54      D5 1906 4339 110$:  TSTL   R4 ; Any work to do?
          OD      13 1908 4340      BEQL   130$                    ; All done
          55      54 D0 190A 4341      MOVL   R4, R5 ; UCB to delete
          54  0138 C5 D0 190D 4342      MOVL   UCBS$L_LT_DEADLINK(R5), R4 ; Save link to next one
          FCOA 30 1912 4343      BSBW   LT$DESTROYUCB ; Destroy this UCB
          EF      11 1915 4344      BRB    110$ ; and go back for more
          1917 4345      :
          1917 4346 130$:  :
          1917 4347      :
          1917 4348      :
          1917 4349      :
          1917 4350      :
          1917 4351      :
          1917 4352      :
          1917 4353      :
          EAD3 CF B7 1917 4354      DECW    GHB_W_MULTIMR ; Time to transmit?
          03      15 1918 4355      BLEQ    150$ ; Br if yes, do it
          0147 31 191D 4356 140$:  BRW    300$ ; Else, still time to go
          1920 4357      :
          52  EACC CF D0 1920 4358 150$:  MOVL   GHB_L_MULTIBFR, R2 ; Get multicast buffer
          OD      13 1925 4359      BEQL   155$ ; Br if none
          1927 4360      ASSUME  GHB_STS_V_MULTI EQ 0 ;
          F1  E86D CF E8 1927 4361      BLBS   GHB_B_STATUS,140$ ; Ignore, if already in progress
          E869 CF 91 192C 4362      CMPB   GHB_B_INCARN, - ; Has it been rebuilt?
          4E  A2      1930 4363      :
          06      13 1932 4364      BEQL   160$ ; if so, incarn has been
          FO2E 30 1934 4365 155$:  BSBW   LT$SETENTRY ; bumped to say so
          E3 50  E9 1937 4366      BLBC   R0, 140$ ; Go rebuild the message
          E860 CF 98 193A 4367 160$:  MOVZBW <XMT_B_MC_MULTIMR-XMT_B_MC_SET>- ; Br if failure
          EAAD CF 193E          : ; Reset multicast timer
          05      12 1941 4368      BNEQ    +GHB_T_MC_DATA,GHB_W_MULTIMR ; from LATCP set data
          1941 4369          : ; Br if value is okay

```

Now check to see if there are dead ucbs waiting to be cleaned up and if noone is using the io database right now, we can delete them all in a flash.

Transmit a configuration message if its time. We tell all concentrators our node name, announcement string every so often so they know we exist.


```

      53 04 C0 19E7 4421
      53 04 D6 19E7 4422 EIGHT: ADDL #4,R3 ; Rating will be highest
      53 04 D6 19EA 4423 FOUR: INCL R3 ; next rating
      53 04 D6 19EC 4424 THREE: INCL R3 ; etc.
      53 04 D6 19EE 4425 TWO: INCL R3 ;
      50 53 C4 19F0 4426 ONE: ;
00000320 8F C6 19F0 4427 MULL R3,R0 ; Compute adjusted rating
      50 8F C6 19F3 4428 DIVL #800,R0 ; Divide by highest rating + %
53 E7A0 CF 9A 19FA 4429 MOVZBL <XMT_B_MC_MULTIMR-XMT_B_MC_SET>- ; Get multicast timer
      50 53 C6 19FF 4430 +GHB_T_MC_DATA,R3
      50 OD 13 1A02 4431 DIVL R3,R0 ; Normalize rating by time
000000FF 8F 50 D1 1A02 4432 BEQL 165$ ; Br if zero rating
      50 06 15 1A04 4433 CMPL R0,#255 ; Ridiculous rating ?
      50 FE 8F 9A 1A06 4434 BLEQ 167$ ; Br if no, continue
      50 FE 8F 9A 1A0B 4435 MOVZBL #254,R0 ; Else, set highest rating
      50 FE 8F 9A 1A0D 4436 ; let INCL up it to 255
      50 FE 8F 9A 1A11 4437 165$: ;
      50 FE 8F 9A 1A11 4438 ; Don't let rating fall to zero, unless the joblimit is reached
      50 FE 8F 9A 1A11 4439 ; or the jobcnt if greater than or equal to the joblim.
      50 FE 8F 9A 1A11 4440 ;
00000000 GF 50 D6 1A11 4441 INCL R0 ; Else, set to minimum
00000000 GF 50 B1 1A13 4442 167$: CMPW G^SYSSGW_IJOBcnt,G^SYSSGW_IJOBlim ; Are we maxed out?
      50 02 1F 1A1E 4443 BLSSU 169$ ; Br if no, continue
      50 50 D4 1A20 4444 CLRL R0 ; Else, set to minimum
      61 50 91 1A22 4445 169$: CMPB R0,(R1) ; New second service rating?
      61 05 13 1A25 4446 BEQL 170$ ; Br if no
      61 50 90 1A27 4447 MOVB R0,(R1) ; Else, set new 2nd svc rating
      61 0A 11 1A2A 4448 BRB 175$
00000000 GF 91 1A2C 4449 170$: CMPB G^SYSSGW_IJOBcnt, 1(R4) ; New number of interactive
      01 A4 1A32 4450 ; users?
      01 A4 1A34 4451 ;
00000000 GF 13 13 1A34 4452 BEQL 180$ ; Br if no
      01 A4 90 1A36 4453 175$: MOVB G^SYSSGW_IJOBcnt, 1(R4) ; Use the interactive count
      4E A2 96 1A3E 4454 ; as the second rating
      E754 CF 96 1A3E 4455 INCB XMT_B_MC_INCARN(R2) ; set new incarnation
      4F A2 08 8C 1A41 4456 INCB GHB_B_INCARN ; no call LTISETENTRY next time
      54 E73B CF D0 1A45 4457 XORB #XMT_CHFLG_M_RATE,XMT_B_MC_CHG_FLAG(R2) ; flag rate change
      17 13 1A49 4458 180$: MOVL GHB_L_FFI, R4 ; Get FFI block address
      53 52 D0 1A49 4459 BEQL 300$ ; Br if none
      E98B CF 7D 1A4E 4460 SETBIT #GHB_STS_V_MULTI,GHB_B_STATUS ; Indicate that multicast is active
      28 A3 7D 1A50 4461 MOVL R2,R3 ; Copy CXB address
      14 A3 18 A3 D0 1A59 4463 MOVQ GHB_Q_MULTIADD, ; Copy the destination address
      10 B4 16 1A5D 4464 CXBSQ_STATION(R3)
      OFE0 8F BA 1A5F 4465 ASSUME CXBSW_BCNT EQ CXBSW_BOFF+2
      10 B4 16 1A5F 4466 MOVL CXBSW_BOFF(R3),CXBSW_T_SAVE(R3) ; Save BOFF and BCNT
      OFE0 8F BA 1A64 4467 JSB @FFISL_XMIT(R4) ; Start XMIT operation
      OFE0 8F BA 1A67 4469 300$: POPR #*M<R5,R6,R7,R8,R9,R10,R11> ; Restore the regs
      05 1A6B 4471 ENBINT ; Restore priority
      05 1A6E 4472 RSB

```



```

1A6F 4476 .SBTTL LTSXMIT - Transmit a message
1A6F 4477 :++
1A6F 4478 : LTSXMIT - Transmit a message
1A6F 4479 :
1A6F 4480 : FUNCTIONAL DESCRIPTION:
1A6F 4481 :
1A6F 4482 : If the CXB is outstanding then record the fact that there is
1A6F 4483 : a waiting message. If the CXB is not outstanding, then ready
1A6F 4484 : it for transmission and send it to the other driver as a write.
1A6F 4485 :
1A6F 4486 : Transmissions must make progress. That is the sequence and acks
1A6F 4487 : of the messages must change over time or we decide that the
1A6F 4488 : circuit is really deadlocked and we take it down. We are generous
1A6F 4489 : with the counter for detecting this deadlock.
1A6F 4490 :
1A6F 4491 : Input:
1A6F 4492 : R4 = CSB address
1A6F 4493 :
1A6F 4494 : Output:
1A6F 4495 : R0-R5 clobbered
1A6F 4496 :-
1A6F 4497 :
1A6F 4498 .ENABLE LOCAL_BLOCK
1A6F 4499
1A6F 4500 LTSXMIT:
53 34 B4 OF 1A6F 4501 REMQUE @CSB_Q_XBUFQ(R4), R3 : Remove the buffer from queue
06 1C 1A73 4502 BVC 5$ : We had better have a buffer here.
1A75 4503 DEBUG : Fatal error, no buffer
59 A4 96 1A7B 4504 5$: INCB CSB_B_XMTCNT(R4) : One more buffer to transmit
62 A4 96 1A7E 4505 INCB CSB_B_XSEQ(R4) : Next message to send
5C A4 7D 1A81 4506 MOVQ CSB_T_CIRCHDR(R4),- : Set the circuit header for the
48 A3 1A84 4507 XMT_T_DATA(R3) : message
50 48 A3 9E 1A86 4508 MOVAB XMT_T_DATA(R3), R0 : Start of data to send
1C A3 50 C2 1A8A 4509 SUBL R0,-CXBSL_T_ENDADR(R3) : Make length from end address
1C A3 1A8E 4510 MOVW CXBSL_T_ENDADR(R3),- : Copy length to CXB format
1A  A3 1A91 4511 CXBSW_BCNT(R3)
40 B4 63 OE 1A93 4512 INSQUE (R3),@CSB_Q_XWAITQ+4(R4) : Queue to tail of waiting buffers
58 A4 95 1A97 4513 TSTB CSB_B_XMTBSY(R4) : Transmitter busy?
3D 13 1A9A 4514 BEQL 60$ : Br if no, start transmitting
58 A4 96 1A9C 4515 10$: INCB CSB_B_XMTBSY(R4) : Else, count another transmit
1A9F 4516 20$: RSB : we just placed one on the wait queue
05 1A9F 4517
1AA0 4518
1AA0 4519 :
1AA0 4520 : Transmit just the one buffer, not all the ones waiting.
1AA0 4521 :
1AA0 4522 LTSXMITONE:
53 34 B4 OF 1AA0 4523 REMQUE @CSB_Q_XBUFQ(R4), R3 : Get the one buffer of interest
06 1C 1AA4 4524 BVC 30$ : Got it
1AA6 4525 DEBUG : No buffer for this transmit
59 A4 96 1AAC 4526 30$: INCB CSB_B_XMTCNT(R4) : Count one more on wait queue
62 A4 96 1AAF 4527 INCB CSB_B_XSEQ(R4) : Next message to send
5C A4 7D 1AB2 4528 MOVQ CSB_T_CIRCHDR(R4),- : Set the circuit header for the
48 A3 1AB5 4529 XMT_T_DATA(R3) : message
50 48 A3 9E 1AB7 4530 MOVAB XMT_T_DATA(R3), R0 : Start of data to send
1C A3 50 C2 1ABB 4531 SUBL R0,-CXBSL_T_ENDADR(R3) : Make length from end address
1C A3 1ABF 4532 MOVW CXBSL_T_ENDADR(R3),- : Copy length to CXB format

```

```

1A A3      1AC2 4533      CXBSW_BCNT(R3)
58 A4      95 1AC4 4534      TSTB   CSB_B_XMTBSY(R4)      ; Are we busy??
   05      12 1AC7 4535      BNEQ   40$                ; Br if yes, so carry on
58 A4      96 1AC9 4536      INCB   CSB_B_XMTBSY(R4)      ; Just one buffer
   25      11 1ACC 4537      BRB    90$                ; Go
40 B4      63 0E 1ACE 4539 40$:  INSQUE (R3), @CSB_Q_XWAITQ+4(R4) ; Place it on the queue
   CB      11 1AD2 4540      BRB    10$                ; And transmit later
   1AD4 4541
   1AD4 4542 LTSREXMIT:
58 A4      95 1AD4 4543      TSTB   CSB_B_XMTBSY(R4)      ; Entry to start a retransmit
   C6      12 1AD7 4544      BNEQ   20$                ; Is the transmit busy??
58 A4      59 A4 90 1AD9 4545 60$:  MOVW   CSB_B_XMTCNT(R4), - ; Br if yes, just leave
   1ADE 4546      CSB_B_XMTBSY(R4)          ; Start a series of retransmits
   1ADE 4547
   1ADE 4548
   1ADE 4549 ;: ** debug code
59 A4      91 1ADE 4550      CMPB   CSB_B_XMTCNT(R4), - ; All buffers on the transmit queue?
   02      1AE1 4551      #LATSC_XMT_BUFFERS
   09      19 1AE2 4552      BLSS   80$                ; Nope
   06      13 1AE4 4553      BEQL   70$                ; Yes all of them
   01 1AEC 4555 70$:  NOP                          ; More than all of them
   1AED 4556 80$:
   1AED 4557 ;: ** debug code
   1AED 4558
   1AED 4559 LTSXMTGO:
53 3C B4    0F 1AED 4560      REMQUE @CSB_Q_XWAITQ(R4), R3 ; Enter to transmit for waiters
   AC      1D 1AF1 4561      BVS    20$                ; Obtain the next waiting buffer
56 A4      B1 1AF3 4562 90$:  CMPW   CSB_W_PROGSEQ(R4), - ; Br if no buffer to transmit
62 A4      1F 1AF6 4563      CSB_W_XSEQ(R4)            ; Are we making progress?
   1F      12 1AF8 4564      BNEQ   100$              ; are the seq and ack fields changing?
54 A4      B7 1AFA 4565      DECM   CSB_W_PROGRESS(R4) ; Br if yes, splendid
   23      14 1AFD 4566      BGTR   110$              ; Nope, give them some time then
34 A4      63 0E 1AFF 4567      INSQUE (R3), CSB_Q_XBUFQ(R4) ; We have more time then
   58      DD 1B00 4568      INC_CTR GMBL_CIRCDOWN+GMB_AREA ; Put the buffer back into the CSB
58 54      D0 1B0F 4570      PUSHL R8                 ; Count a circuit timeout
   FC15    30 1B12 4571      MOVL   R4, R8             ; No more time. Clobber the circuit
   58 BED0 05 1B15 4572      BSBW   LATSC_CIRCDEAD    ; Since it appears to be deadlocked.
   1B18 4573      RSB    R8                 ; and clean it all up
   1B19 4574
62 A4      B0 1B19 4575 100$: MOVW   CSB_W_XSEQ(R4), - ; Reset the current sequence and ack
56 A4      14 B0 1B1C 4576      CSB_W_PROGSEQ(R4)        ; to check against
   14      B0 1B1E 4577      MOVW   #LATSC_PROGRESS, - ; and reset the progress counter
54 A4      1820 4578      CSB_W_PROGRESS(R4)
   1B22 4579
   1B22 4580 110$:
   1B22 4581
   1B22 4582
   1B22 4583
   1B22 4584 Build the CXB to transmit
55 E662 CF  D0 1B22 4584      MOVL   GMB_L_FFI, R5      ; Get the FFI block address
   2A      13 1B27 4585      BEQL   150$              ; Br if none
51 5A A4    9A 1B29 4586      MOVZBL CSB_B_XCXB_INX(R4), R1 ; Get XMIT CXB index slot
44 A441 53  D0 1B2D 4587      MOVL   R3, CSB_L_XCXB(R4)[R1] ; Save address of CXB
   51      96 1B32 4588      INCB   R1                 ; Skip to next slot
51 FE BF 8A 1B34 4589      BICB   #^C<LATSC_XMT_BUFFERS-1>, R1 ; Modulo maximum

```

5A	A4	51	90	1B38	4590	MOVW	R1, CSB_B_X(CXB_INX(R4))	:	Save for next time
		54	DD	1B3C	4591	PUSHL	R4	:	Save R4
	54	55	DD	1B3E	4592	MOVL	R5, R4	:	Else, copy the FFI block address
14	A3	18	A3	DD	1B41	4593	ASSUME	CXBSW_BCNT EQ CXBSW_BOFF+2	
					1B41	4594	MOVL	CXBSW_BOFF(R3), CXBSW_T_SAVE(R3)	: Save BOFF and BCNT
					1B46	4595			: progress
					1B46	4596			
					1B46	4597			
					1B46	4598			
					1B46	4599			
50	48	A3	9E	1B46	4600	MOVAB	XMT_B_FLAG(R3), R0	:	Address of data to save
		EBCA	30	1B4A	4601	BSBW	LT\$HISTORY	:	Save the data
				1B4D	4602				
				1B4D	4603				
				1B4D	4604				
	10	B4	16	1B4D	4605	JSB	@FFISL_XMIT(R4)	:	Start the XMIT operation
		54	8ED0	1B50	4606	POPL	R4	:	Restore R4
			05	1B53	4607	RSB		:	Return to caller
				1B54	4608				
				1B54	4609				
				1B54	4610				

1508:

03	12	1BBF	4669		BNEQ	70\$: Br if not
F85C	30	1BC1	4670		BSBL	LTSDEALOCB		: Get rid of the circuit block
36	BA	1BC4	4671	70\$:	POPR	#*M<R1,R2,R4,R5>		: Restore registers
	05	1BC6	4672		RSB			
		1BC7	4673		:			
		1BC7	4674		:	Multicast buffer completed		
		1BC7	4675		:			
		1BC7	4676	110\$:	CLRBIT	#GHB_STS_V_MULTI,-		: Multicast is not busy now
		1BC7	4677			GHB_B_STATUS		
F5	11	1BCD	4678		BRB	70\$: And exit
		1BCF	4679					

-1
CL
CL
DE
LI

```
1BCF 4681 .SBTTL LT$FFIPOSTDONE - Garbage transmit posting routine
1BCF 4682 :++
1BCF 4683 : LT$FFIPOSTDONE - Garbage transmit posting routine
1BCF 4684 :
1BCF 4685 : FUNCTIONAL DESCRIPTION:
1BCF 4686 :
1BCF 4687 : Final post routine to dry up a CXB I/O when we are done
1BCF 4688 : with the channel. We simply deallocate any transmit buffer returned.
1BCF 4689 :
1BCF 4690 : INPUTS:
1BCF 4691 : R3 = CXB address
1BCF 4692 : R4 = FFI block address
1BCF 4693 :
1BCF 4694 : OUTPUTS:
1BCF 4695 : R0 clobbered
1BCF 4696 :--
1BCF 4697 :
1BCF 4698 LT$FFIPOSTDONE:
50 36 BB 1BCF 4699 PUSHR #*M<R1,R2,R4,R5> ; Save registers
53 DO 1BD1 4700 MOVL R3,R0 ; Copy CXB address
03 10 1BD4 4701 BSBB LT$DEALPOOL ; Drop the buffer
36 BA 1BD6 4702 POPR #*M<R1,R2,R4,R5> ; Restore registers
05 1BD8 4703 LT$DROPCXB: ; Drop all RECEIVE CXBs
1BD8 4704 RSB
1BD9 4705
00000000'GF 17 1BD9 4706 LT$DEALPOOL:
1BD9 4707 JMP G*EXE$DEANONPAGED ; Deallocate the FFI block
1BDF 4708
1BDF 4709 LT$SEND: ; End of driver
1BDF 4710
1BDF 4711 .END
```

LTDRIVER
Symbol table

- Local Area Terminal Port Driver C 7

8-JAN-1985 17:22:25 VAX/VMS Macro V04-00
5-SEP-1984 11:40:22 [LAT.BUGSRC]LTDRIVER.MAR;1

\$\$\$	= 00000020	R	02	CSB_W_PROGRESS	00000054		
\$\$BASE	= 00000001			CSB_W_PROGSEQ	00000056	G	
\$\$DISPL	= 0700000A			CSB_W_REMID	0000005E	G	
\$\$GENSW	= 00000001			CSB_W_RSEQ	00000064	G	
\$\$HIGH	= 00000009			CSB_W_SIZE	00000008	G	
\$\$LIMIT	= 00000008			CSB_W_TIMEOUT	00000050	G	
\$\$LOW	= 00000001			CSB_W_TIMRESET	00000066	G	
\$\$MNSW	= 00000001			CSB_W_XMTTMO	00000052	G	
\$\$MXSW	= 00000001			CSB_W_XSEQ	00000062	G	
\$\$OP	= 00000002			CSB_Z_DST	0000002E	G	
ATS_NULL	*****	X	02	CSB_Z_LCB	0000000C	G	
BIT...	= 00000004			CXBSB_CODE	= 0000000B		
BUGS_UNSUPRTCPU	*****	X	03	:XBSB_TYPE	= 0000000A		
BUILD_TQF	= 000008E2	R	03	(XBSB_HEADER	= 00000048		
CLASS_DDT	= 00000010			(XBSL_END_ACTION	= 0000001C		
CLASS_DISCONNECT	= 00000018			(XBSL_IRP	= 00000014		
CLASS_GETNXT	= 00000000			CXBSL_T_CSB	0000000C	G	
CLASS_PUTNXT	= 00000004			CXBSL_T_ENDADR	0000001C	G	
CLASS_READERROR	= 00000014			CXBSL_T_SAVE	00000014	G	
CLASS_SETUP_UCB	= 00000008			CXBSQ_STATION	= 00000028		
CRBSL_INTD	= 00000024			CXBSW_BCNT	= 0000001A		
CRBSW_REFC	= 0000000C			CXBSW_BOFF	= 00000018		
CREDIT_CHECK	= 000010E2	R	03	CXBSW_LENGTH	= 0000000C		
CSB_B_FLAG	= 0000005C	G		CXBSW_OFFSET	= 0000000E		
CSB_B_INX	= 00000068	G		CXBSW_SIZE	= 00000008		
CSB_B_LOCIDN	= 00000060	G		DCS_TERM	*****	X	02
CSB_B_LOCIDS	= 00000061	G		DDBSL_DDT	= 0000000C		
CSB_B_MAX_SLOTS	= 00000069	G		DDBSL_UCB	= 00000004		
CSB_B_NUM_SLOTS	= 0000005D	G		DEFREASN	= 00001752	R	03
CSB_B_RACK	= 00000065	G		DEVSM_AVL	= 00040000		
CSB_B_REFC	= 0000002D	G		DEVSM_CCL	= 00000002		
CSB_B_REMIDN	= 0000005E	G		DEVSM_IDV	= 04000000		
CSB_B_REMIDS	= 0000005F	G		DEVSM_NNM	= 00000200		
CSB_B_RSEQ	= 00000064	G		DEVSM_ODV	= 08000000		
CSB_B_SERVER	= 0000001C	G		DEVSM_REC	= 00000001		
CSB_B_SPARE	= 0000005B	G		DEVSM_TRM	= 00000004		
CSB_B_STATE	= 0000000B	G		DPTSB_FLAGS	= 0000000D		
CSB_B_TYPE	= 0000000A	G		DPTSC_LENGTH	= 00000038		
CSB_B_XACK	= 00000063	G		DPTSC_VERSION	= 00000004		
CSB_B_XCXB_INX	= 0000005A	G		DPTSIRITAB	= 00000038	R	02
CSB_B_XMTBSY	= 00000058	G		DPTSM_NOUNLOAD	= 00000004		
CSB_B_XMTCNT	= 00000059	G		DPTSREINITAB	= 000000D3	R	02
CSB_B_XSEQ	= 00000062	G		DPTSTAB	= 00000000	R	02
CSB_C_FIXLENGTH	= 0000006C	G		DPTSW_VECTOR	= 0000001E		
CSB_C_STATE_HALT	= 00000000	G		DYNDC_BUFIO	= 00000013		
CSB_C_STATE_RUN	= 00000002	G		DYNDC_CDB	= 00000033		
CSB_C_STATE_START	= 00000001	G		DYNDC_CRB	= 00000005		
CSB_L_STOPREASON	= 0000004C	G		DYNDC_CXB	= 0000001B		
CSB_L_UCBLST	= 0000006C	G		DYNDC_DDB	= 00000006		
CSB_L_XCXB	= 00000044	G		DYNDC_DPT	= 0000001E		
CSB_Q_XBUFO	= 00000034	G		DYNDC_ORB	= 00000049		
CSB_Q_XWAITQ	= 0000003C	G		DYNDC_TQE	= 0000000F		
CSB_S_CIRCHDR	= 00000008	G		DYNDC_UCB	= 00000010		
CSB_T_CIRCHDR	= 0000005C	G		ECHO_ERROR	= 0000101A	R	03
CSB_T_SERVER	= 0000001D	G		EIGHT	= 000019E7	R	03
CSB_W_LOCID	= 00000060	G		EXESALLOCTQE	*****	X	03
CSB_W_MAX_MSGSIZ	= 0000006A	G		EXESALONONPAGED	*****	X	03

LTDRIVER
Symbol table

- Local Area Terminal Port Driver

D 7

8-JAN-1985 17:22:25 VAX/VMS Macro V04-00
5-SEP-1984 11:40:22 [LAT.BUGSRC]LTDRIVER.MAR;1

Page 114
(66)

EXESAL_TQENOREPT	*****	X	03	GHBSL_STRETRY	=	00000010		
EXESDEANONPAGED	*****	X	03	GHBSL_TIM_ACT	=	00000018		
EXESFORK	*****	X	03	GHBSL_UCB	=	00000014		
EXESGB_CPUDATA	*****	X	03	GHBSL_XCOUNT	=	00000038		
EXESGB_CPUTYPE	*****	X	03	GHBSL_XERR	=	0000003C		
EXESGL_MP	*****	X	03	GHBSL_XERRCOD	=	00000040		
EXESGR_SYSTIME	*****	X	03	GHBSQ_OLD_CSBS	=	00000024		
EXESINSTIMO	*****	X	03	GHBS\$COMM AREA	=	00000060		
FFISB_SPARE	= 0000000B			GHBST_CSBCTR	=	0000000C		
FFISB_TYPE	= 0000000A			GHB\$V_BADCREDITS	=	00000008		
FFISC_LENGTH	= 0000003E			GHB\$V_CSBINVALID	=	00000003		
FFISL_BL	= 00000004			GHB\$V_CSBRANGE	=	00000002		
FFISL_CTX_DL	= 0000000C			GHB\$V_CSBSTALE	=	00000004		
FFISL_DL_OCB	= 00000034			GHB\$V_CSBZERO	=	00000001		
FFISL_ERROR	= 0000001C			GHB\$V_HALT	=	00000005		
FFISL_FL	= 00000000			GHB\$V_INVALIDLOCID	=	00000007		
FFISL_PID	= 00000038			GHB\$V_INVALIDREMID	=	00000006		
FFISL_RECV_DONE	= 00000018			GHB\$V_INVALIDSEQ	=	0000000A		
FFISL_SHUT_DONE	= 00000020			GHB\$V_REPCREATE	=	00000009		
FFISL_SPARE0	= 00000024			GHB\$V_REPDISC	=	0000000B		
FFISL_SPARE1	= 00000028			GHB\$V_START	=	00000000		
FFISL_SPARE2	= 0000002C			GHB\$W_VMSVERSION	=	00000000		
FFISL_SPARE3	= 00000030			GHB_AREA		00000120	R	03
FFISL_XMIT	= 00000010			GHB_B_INCARN		00000199	R	03
FFISL_XMIT_DONE	= 00000014			GHB_B_LOC_CCHK		000000B8	R	03
FFISW_CHAN	= 0000003C			GHB_B_MAX_SLOTS		0000010C	R	03
FFISW_SIZE	= 00000008			GHB_B_OLD_CSBCNT		0000019A	R	03
FKBSB_FIPL	= 0000000B			GHB_B_RATE		0000019B	R	03
FLAG_M_MASTER	= 00000002	G		GHB_B_STATUS		00000198	R	03
FLAG_M_MTYPE	= 000000FC	G		GHB_C_STRT_SLOTL	=	00000005		
FLAG_M_RRF	= 00000001	G		GHB_G_TQE		000000DC	R	03
FLAG_S_MASTER	= 00000001	G		GHB_L_AREA		000004B8	R	03
FLAG_S_MTYPE	= 00000006	G		GHB_L_CSBTABLE		00000038	R	03
FLAG_S_RRF	= 00000001	G		GHB_L_DEADLINK		00000184	R	03
FLAG_V_MASTER	= 00000001	G		GHB_L_FFI		00000188	R	03
FLAG_V_MTYPE	= 00000002	G		GHB_L_LASTCPU		00000190	R	03
FLAG_V_RRF	= 00000000	G		GHB_L_MULTIBFR		000003F0	R	03
FOUR	= 000019EA	R	03	GHB_L_NULLCPU		0000018C	R	03
FUNCTAB_LEN	= 00000000			GHB_L_NULLSEC		00000194	R	03
GET MORE	= 00001196	R	03	GHB_L_UCB0		00000180	R	03
GHBSB_LATECO	= 00000003			GHB_Q_MULTIADD		000003E8	R	03
GHBSB_LATVERSION	= 00000002			GHB_STS_M_ACTIVE	=	00000002	G	
GHBSK_IDLEN	= 00000040			GHB_STS_M_MULTI	=	00000001	G	
GHBSK_NAMELEN	= 00000010			GHB_STS_M_SHUT	=	00000004	G	
GHBSL_CIRCDOWN	= 00000048			GHB_STS_M_TQE	=	00000008	G	
GHBSL_CSBLST	= 00000020			GHB_STS_S_ACTIVE	=	00000001	G	
GHBSL_DUPLMSG	= 00000034			GHB_STS_S_MULT:	=	00000001	G	
GHBSL_HISTORY	= 00000008			GHB_STS_S_SHUT	=	00000001	G	
GHBSL_NODE	= 0000001C			GHB_STS_S_TQE	=	00000001	G	
GHBSL_NOXBFR	= 00000058			GHB_STS_V_ACTIVE	=	00000001	G	
GHBSL_PROTOCOL	= 0000004C			GHB_STS_V_MULTI	=	00000000	G	
GHBSL_PROTOMASK	= 00000050			GHB_STS_V_SHUT	=	00000002	G	
GHBSL_RCOUNT	= 0000002C			GHB_STS_V_TQE	=	00000003	G	
GHBSL_RESOURCE	= 00000054			GHB_T_MAX_SLOT_TABLE		0000010D	R	03
GHBSL_RETRANS	= 00000044			GHB_T_MC_DATA		0000019E	R	03
GHBSL_SEENTRY	= 00000004			GHB_T_MC_MSG		000003F4	R	03
GHBSL_SHUTENTRY	= 0000000C			GHB_T_STRT_MSG		00000400	R	03

LTDRIVER
Symbol table

- Local Area Terminal Port Driver F 7

8-JAN-1985 17:22:25 VAX/VMS Macro V04-00
5-SEP-1984 11:40:22 [LAT.BUGSRC]LTDRIVER.MAR;1

Page 116
(66)

LTSSIDELINEUCB	00001709	R	03	PRS_IPL	= 00000012		
LTSSSTARTIO	00000ADE	R	03	PRS_SID_TYP730	= 00000003		
LTSSSTATE_ENTER_RUN	00000EA6	R	03	PRS_SID_TYP750	= 00000002		
LTSSSTATE_HALT	00000E5C	R	03	PRS_SID_TYP780	= 00000001		
LTSSSTATE_RUN	00000EAE	R	03	PRS_SID_TYP785	= 00000009		
LTSSSTATE_START	00000E82	R	03	PRS_SID_TYP790	= 00000004		
LTSSSTOPCIRC	0000176B	R	03	PRS_SID_TYPMAX	= 00000008		
LTSSSTRETRY	0000075D	R	03	PRS_SID_TYPUV1	= 00000007		
LTSTICK	0000188C	R	03	PRS_SID_TYPUV2	= 00000008		
LTSUCB_INIT	00000A9B	R	03	RCV_B_ACK	00000007	G	
LTSUNIT_INIT	00000A2F	R	03	RCV_B_DSTIDN	00000002	G	
LTSVEC	000004BC	R	03	RCV_B_DSTIDS	00000003	G	
LTSVECEND	000004F4	R	03	RCV_B_FLAG	00000000	G	
LTSVECTOR	000004BC	R	03	RCV_B_NUM_SLOTS	00000001	G	
LTSXMIT	00001A6F	R	03	RCV_B_SEQ	00000006	G	
LTSXMITONE	00001AA0	R	03	RCV_B_SRCIDN	00000004	G	
LTSXMTGO	00001AED	R	03	RCV_B_SRCIDS	00000005	G	
LTSXMT_FFIDONE	00001B54	R	03	RCV_T_DATA	00000000	G	
LTSXOFF	00000CCE	R	03	RCV_T_MDATA	00000008	G	
LTSXON	00000C7E	R	03	RCV_W_DSTID	00000002	G	
LT_HISTORY	= 00000001			RCV_W_SRCID	00000004	G	
LT_RSN_BADCREDITS	000005E9	R	03	RSNS_AWAIT	= 00000001		
LT_RSN_CSBINVALID	00000561	R	03	SCH\$GL_CURPCB	*****	X	03
LT_RSN_CSBRANGE	00000541	R	03	SCH\$GL_NULLPCB	*****	X	03
LT_RSN_CSBSTALE	0000057C	R	03	SCH\$RAVAIL	*****	X	03
LT_RSN_CSZERO	00000529	R	03	SCH\$RWAIT	*****	X	03
LT_RSN_HALT	00000598	R	03	SEND_STOP_SLOT	00001020	R	03
LT_RSN_INVALIDLOCID	000005B3	R	03	SHUT_TIMER	00000936	R	03
LT_RSN_INVALIDIDREMID	000005CE	R	03	SIZ...	= 00000001		
LT_RSN_REPCREATE	0000060C	R	03	SLOT_LOOP_IN	00000ED9	R	03
LT_RSN_REPDISC	00000630	R	03	SLOT_LOOP_OUT	00001044	R	03
LT_RSN_REPSTART	000004F9	R	03	SLOT_SIZE	= 00000020		
MAX_PR_SID	= 00000010			SLT_ATT_M_ABORT	= 00000020	G	
MP\$GL_MAXPFN	*****	X	03	SLT_ATT_S_ABORT	= 00000001	G	
MP\$GL_NULLCPU	*****	X	03	SLT_ATT_V_ABORT	= 00000005	G	
MTXSW_OUNCNT	= 00000000			SLT_B_ATT_CONTROL	00000004	G	
MTYP_C_CONF	= 0000000A	G		SLT_B_ATT_SLTSIZ	00000005	G	
MTYP_C_HALT	= 00000002	G		SLT_B_COUNT	00000002	G	
MTYP_C_RUN	= 00000000	G		SLT_B_CRED	00000003	G	
MTYP_C_START	= 00000001	G		SLT_B_DB_CONTROL	00000004	G	
NEXT_SLOT	00001007	R	03	SLT_B_DB_IFOF	00000007	G	
NEXT_SLOTO	00000F90	R	03	SLT_B_DB_IFON	00000008	G	
ONE	000019F0	R	03	SLT_B_DB_OFOF	00000005	G	
ONE_LESS	0000104C	R	03	SLT_B_DB_OFON	00000006	G	
ORBSB_FLAGS	= 00000008			SLT_B_DSTID	00000000	G	
ORBSL_OWNER	= 00000000			SLT_B_DST_NAMLEN	00000007	G	
ORBSM_PROT_16	= 00000001			SLT_B_DT_SLTSIZ	00000006	G	
ORBSW_PROT	= 00000018			SLT_B_PARAMS	00000029	G	
PCBSL_PHD	= 0000006C			SLT_B_REAS	00000003	G	
PHDSL_CPUTIM	= 00000038			SLT_B_REJ_RLEN	00000004	G	
PORT_ABORT	= 00000020			SLT_B_SRCID	00000001	G	
PORT_DISCONNECT	= 00000004			SLT_B_SRC_NAME	00000019	G	
PORT_LENGTH	= 00000038			SLT_B_SRC_NAMLEN	00000018	G	
PORT_SET_LINE	= 00000008			SLT_B_STP_RLEN	00000004	G	
PORT_STARTIO	= 000C0000			SLT_B_SVCC	00000004	G	
PORT_XOFF	= 00000014			SLT_B_TYPE	00000003	G	
PORT_XON	= 00000010			SLT_C_ATT_SLOT	= 00000008	G	

LTDRIVER
Symbol table

- Local Area Terminal Port Driver 6 7

8-JAN-1985 17:22:25 VAX/VMS Macro V04-00
5-SEP-1984 11:40:22 [LAT.BUGSRC]LTDRIVER.MAR;1

```

SLT_C_DA_SLOT      = 00000000 G
SLT_C_DB_SLOT      = 0000000A G
SLT_C_INVSLLOT     = 00000003 G
SLT_C_INVSVC       = 00000004 G
SLT_C_REJ_SLOT     = 0000000C G
SLT_C_RESOURCE     = 00000005 G
SLT_C_START_FLAG   = 00000001 G
SLT_C_STP_SLOT     = 0000000D G
SLT_C_STR_SLOT     = 00000009 G
SLT_C_SVC_INTT     = 00000001 G
SLT_C_SYS          = 00000002 G
SLT_C_USD          = 00000001 G
SLT_DB_M_BREAK     = 00000010 G
SLT_DB_M_IFDIS     = 00000002 G
SLT_DB_M_IFENA     = 00000001 G
SLT_DB_M_OFDIS     = 00000008 G
SLT_DB_M_OFENA     = 00000004 G
SLT_DB_S_BREAK     = 00000001 G
SLT_DB_S_IFDIS     = 00000001 G
SLT_DB_S_IFENA     = 00000001 G
SLT_DB_S_OFDIS     = 00000001 G
SLT_DB_S_OFENA     = 00000001 G
SLT_DB_V_BREAK     = 00000004 G
SLT_DB_V_IFDIS     = 00000001 G
SLT_DB_V_IFENA     = 00000000 G
SLT_DB_V_OFDIS     = 00000003 G
SLT_DB_V_OFENA     = 00000002 G
SLT_FLAG_M_BIND    = 00000004 G
SLT_FLAG_M_LOGIN   = 00000002 G
SLT_FLAG_M_REMOTE  = 00000001 G
SLT_FLAG_S_BIND    = 00000001 G
SLT_FLAG_S_LOGIN   = 00000001 G
SLT_FLAG_S_REMOTE  = 00000001 G
SLT_FLAG_V_BIND    = 00000002 G
SLT_FLAG_V_LOGIN   = 00000001 G
SLT_FLAG_V_REMOTE  = 00000000 G
SLT_S_ATT_LEN      = 00000001 G
SLT_S_ATT_SLOT     = 00000005 G
SLT_S_DB_LEN       = 00000005 G
SLT_S_DB_SLOT      = 00000009 G
SLT_S_REJ_LEN      = 00000000 G
SLT_S_REJ_SLOT     = 00000004 G
SLT_S_START_SLOT   = 00000025 G
SLT_S_STP_LEN      = 00000001 G
SLT_S_STP_SLOT     = 00000005 G
SLT_T_DATA         = 00000004 G
SLT_T_DST_NAME     = 00000008 G
SLT_T_REJ_REAS     = 00000004 G
SLT_T_START        = 00000000 G
SLT_T_STP_REAS     = 00000005 G
SSB_IRSFMEM        = ***** X 03
SSB_NORMAL         = ***** X 03
STOP_B_RCODE       = 00000000 G
STOP_B_RLEN        = 00000001 G
STOP_TDE           = 0000187C R 03
STOP_T_REAS        = 00000002 G
STRT_B_CIR_TIMR    = 00000006 G

```

```

STRT_B_DL_BFRS     = 00000005 G
STRT_B_ID_LEN      = 0000002E G
STRT_B_KPA_TIMR    = 00000007 G
STRT_B_LOC_LEN     = 0000006F G
STRT_B_MAXSLOTS    = 00000004 G
STRT_B_MODE_LEN    = 0000000C G
STRT_B_PARAM       = 00000080 G
STRT_B_PECO        = 00000003 G
STRT_B_PVER        = 00000002 G
STRT_B_SYS_LEN     = 0000001D G
STRT_B_VAR         = 0000000C G
STRT_C_LENGTH      = 0000000C G
STRT_C_VAR_LEN     = = 000000A5 G
STRT_T_ID          = 0000002F G
STRT_T_LOC         = 00000070 G
STRT_T_MODE        = 0000000D G
STRT_T_SYS         = 0000001E G
STRT_W_FAC_NUM     = 00000008 G
STRT_W_MSGSIZ      = 00000000 G
STRT_W_PROD_TYP    = 0000000A G
SYSSGW_IJOBENT     = ***** X 03
SYSSGW_IJOBIM      = ***** X 03
THREE              = 000019EC R 03
TQESB_RQTYPE       = = 00000008 G
TQESB_TYPE         = = 0000000A G
TQESC_LENGTH       = = 00000030 G
TQESC_SSREPT       = = 00000005 G
TQESC_SSSNGL       = = 00000001 G
TQESL_FPC          = = 0000000C G
TQESH_REPEAT       = = 00000004 G
TQESQ_DELTA        = = 00000020 G
TQESW_SIZE         = = 00000008 G
TTSV_HOSTSYNC     = = 00000004 G
TTSV_PASSALL       = = 00000000 G
TTSV_REMOTE        = = 0000000D G
TTSV_TTSYNC        = = 00000005 G
TTS_UNKNOWN        = = 00000000 G
TT2SV_AUTOBAUD    = = 00000001 G
TT2SV_HANGUP       = = 00000002 G
TTY$GB_DEFSPEED    = ***** X 02
TTY$GB_PARITY      = ***** X 02
TTY$GB_RSPEED      = ***** X 02
TTY$GL_DEFCHAR     = ***** X 02
TTY$GL_DEFCHAR2    = ***** X 02
TTY$GL_DELTA       = ***** X 03
TTY$GL_DPT         = ***** X 02
TTY$GL_OWNJIC      = ***** X 02
TTY$GW_DEFBUF      = ***** X 02
TTY$GW_PROT        = ***** X
TTY$V_PC_NOTIME    = = 00000000 G
TTY_C_BECL         = = 00000007 G
TWO                = 000019EE R 03
UCBSB_DEVCLASS     = = 00000040 G
UCBSB_DEVTYPE      = = 00000041 G
UCBSB_DIPL         = = 0000005E G
UCBSB_ERTCNT       = = 00000080 G
UCBSB_FIPL         = = 00000008 G

```

```

X 03
X 03
R 03
X 02
X 02
X 02
X 03
X 02
X 02
R 03

```

LTDRIVER
Symbol table

- Local Area Terminal Port Driver H 7

8-JAN-1985 17:22:25 VAX/VMS Macro V04-00
5-SEP-1984 11:40:22 [LAT.BUGSRC]LTDRIVER.MAR;1

Page 118
(66)

UCBSB_LT_CBUF	0000014C	G	UCBSS_LT_DATA	= 00000001	G
UCBSB_LT_CURC	0000014B	G	UCBSS_LT_DEAD	= 00000001	G
UCBSB_LT_DATAW	00000142	G	UCBSS_LT_FLOW	= 00000001	G
UCBSB_LT_EBUF	0000019C	G	UCBSS_LT_HANGUP	= 00000001	G
UCBSB_LT_LATSTS	00000143	G	UCBSS_LT_INPUT	= 00000001	G
UCBSB_LT_LOCID	00000141	G	UCBSS_LT_OVFLOW	= 00000001	G
UCBSB_LT_MAXC	0000014A	G	UCBSV_DELETEUCB	= 00000010	
UCBSB_LT_RCRED	00000148	G	UCBSV_LT_ABORT	= 00000002	G
UCBSB_LT_REASON	00000145	G	UCBSV_LT_BOUND	= 00000003	G
UCBSB_LT_REMID	00000140	G	UCBSV_LT_DATA	= 00000000	G
UCBSB_LT_SLOTSZ	00000149	G	UCBSV_LT_DEAD	= 00000001	G
UCBSB_LT_STATE	00000144	G	UCBSV_LT_FLOW	= 00000001	G
UCBSB_LT_TCRED	00000146	G	UCBSV_LT_HANGUP	= 00000000	G
UCBSB_LT_XCRED	00000147	G	UCBSV_LT_INPUT	= 00000004	G
UCBSB_TT_DEPARI	= 000000EC		UCBSV_LT_OVFLOW	= 00000002	G
UCBSB_TT_DETYPE	= 000000F0		UCBSV_POWER	= 00000005	
UCBSB_TT_OUTYPE	= 0000010B		UCBSV_TEMPLATE	= 0000000D	
UCBSC_LT_LENGTH	0000019C	G	UCBSW_DEVBUFSIZ	= 00000042	
UCBSC_LT_STATE_KILL	= 00000004	G	UCBSW_DEVSTS	= 00000068	
UCBSC_LT_STATE_RUN	= 00000002	G	UCBSW_ERRCNT	= 00000082	
UCBSC_LT_STATE_START	= 00000001	G	UCBSW_MB_SEED	= 00000000	
UCBSC_LT_STATE_STOP	= 00000003	G	UCBSW_REFC	= 0000005C	
UCBSC_TT_LENGTH	= 00000134		UCBSW_TT_DESIZE	= 000000F1	
UCBSL_CRB	= 00000024		UCBSW_TT_DESPEE	= 000000E8	
UCBSL_DDB	= 00000028		UCBSW_TT_OUTLEN	= 00000120	
UCBSL_DDT	= 00000088		UCBSW_TT_PRTCTL	= 00000122	
UCBSL_DEVCHAR	= 00000038		UCBSW_TT_SPEED	= 000000F4	
UCBSL_DEVCHAR2	= 0000003C		UCBSW_UNIT	= 00000054	
UCBSL_DEVDEPEND	= 00000044		USE_CREDIT	000011D6	R 03
UCBSL_LT_CSB	00000134	G	USE_CREDIT1	000011DC	R 03
UCBSL_LT_DEADLINK	00000138	G	VECSL_INITIAL	= 0000000C	
UCBSL_LT_INPBUF	0000013C	G	VECSL_START	= 0000001C	
UCBSL_STS	= 00000064		VECSL_UNITINIT	= 00000018	
UCBSL_TT_CLASS	= 00000114		XMT_B_ACK	0000004F	G
UCBSL_TT_DECHA1	= 000000C8		XMT_B_DSTIDN	0000004A	G
UCBSL_TT_DECHAR	= 000000C4		XMT_B_DSTIDS	0000004B	G
UCBSL_TT_DEVDP1	= 00000048		XMT_B_FLAG	00000048	G
UCBSL_TT_GETNXT	= 0000010C		XMT_B_MC_CHG_FLAG	0000004F	G
UCBSL_TT_LOGUCB	= 000000C0		XMT_B_MC_CIR_TIMER	00000049	G
UCBSL_TT_OUTADR	= 0000011C		XMT_B_MC_CUR_ECO	0000004D	G
UCBSL_TT_PORT	= 00000118		XMT_B_MC_CUR_VER	0000004C	G
UCBSL_TT_PUTNXT	= 00000110		XMT_B_MC_GRP_LEN	00000054	G
UCBSL_TT_RTIMOU	= 000000B4		XMT_B_MC_HI_VER	0000004A	G
UCBSL_TT_WBLINK	= 000000D0		XMT_B_MC_ID_LEN	00000099	G
UCBSL_TT_WFLINK	= 000000CC		XMT_B_MC_INCRN	0000004E	G
UCBSM_INT	= 00000002		XMT_B_MC_LO_VER	0000004B	G
UCBSM_LT_ABORT	= 00000004	G	XMT_B_MC_MUXTIMR	00000052	G
UCBSM_LT_BOUND	= 00000008	G	XMT_B_MC_NAME_LEN	00000003	G
UCBSM_LT_DATA	= 00000001	G	XMT_B_MC_NODE_LEN	00000075	G
UCBSM_LT_DEAD	= 00000002	G	XMT_B_MC_NUM SVCS	00000006	G
UCBSM_LT_FLOW	= 00000002	G	XMT_B_MC_RATE	00000007	G
UCBSM_LT_HANGUP	= 00000001	G	XMT_B_MC_SET	00000052	G
UCBSM_LT_INPUT	= 00000010	G	XMT_B_MC_STATUS	00000053	G
UCBSM_LT_OVFLOW	= 00000004	G	XMT_B_MC_SVC	0000027A	G
UCBSM_ONLINE	= 00000010		XMT_B_MC_SVC_LEN	00000279	G
UCBSS_LT_ABORT	= 00000001	G	XMT_B_NUM_SLOTS	00000049	G
UCBSS_LT_BOUND	= 00000001	G	XMT_B_SEQ	0000004E	G

LTDRIVER
Symbol table

- Local Area Terminal Port Driver 1 7

8-JAN-1985 17:22:25 VAX/VMS Macro V04-00
5-SEP-1984 11:40:22 [LAT.BUGSRC]LTDRIVER.MAR;1

Page 119
(66)

```

XMT_B_SRCIDN      0000004C  G
XMT_B_SRCIDS      0000004D  G
XMT_CHFLG_M_ALL  = 000000FF  G
XMT_CHFLG_M_CLASS = 00000020  G
XMT_CHFLG_M_GROUP = 00000001  G
XMT_CHFLG_M_NDESC = 00000002  G
XMT_CHFLG_M_OTHER = 00000080  G
XMT_CHFLG_M_RATE  = 00000008  G
XMT_CHFLG_M_SDESC = 00000010  G
XMT_CHFLG_M_SVCNAM = 00000004  G
XMT_CHFLG_S_CLASS = 00000001  G
XMT_CHFLG_S_GROUP = 00000001  G
XMT_CHFLG_S_NDESC = 00000001  G
XMT_CHFLG_S_OTHER = 00000001  G
XMT_CHFLG_S_RATE  = 00000001  G
XMT_CHFLG_S_SDESC = 00000001  G
XMT_CHFLG_S_SVCNAM = 00000001  G
XMT_CHFLG_V_CLASS = 00000005  G
XMT_CHFLG_V_GROUP = 00000000  G
XMT_CHFLG_V_NDESC = 00000001  G
XMT_CHFLG_V_OTHER = 00000007  G
XMT_CHFLG_V_RATE  = 00000003  G
XMT_CHFLG_V_SDESC = 00000004  G
XMT_CHFLG_V_SVCNAM = 00000002  G
XMT_C_MORCEN      = 00000008  G
XMT_C_MAXDATA     = 000005B4  G
XMT_C_MAXLEN      = 00000674  G
XMT_C_MC_LENGTH   = 0000000A  G
XMT_C_MC_SVC_SIZE = 00000053  G
XMT_C_MC_TOTAL    = 0000023A  G
XMT_G_DST         = 0000003A  G
XMT_T_DATA        = 00000048  G
XMT_T_MC_GROUP    = 00000053  G
XMT_T_MC_ID       = 0000009A  G
XMT_T_MC_MORE_SVC = 0000000A  G
XMT_T_MC_NAME     = 00000089  G
XMT_T_MC_NODE     = 00000076  G
XMT_T_MC_DATA     = 00000050  G
XMT_W_DSTID       = 0000004A  G
XMT_W_MC_MSG_SIZE = 00000050  G
XMT_W_SRCID       = 0000004C  G

```

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
.ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABS\$	00000282 (642.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$105_PROLOGUE	0000000E (222.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$115_DRIVER	00001BDF (7135.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
-----	-----	-----	-----
Initialization	96	00:00:00.19	00:00:02.34
Command processing	137	00:00:00.50	00:00:03.20
Pass 1	1002	00:00:31.72	00:01:40.19
Symbol table sort	2	00:00:03.08	00:00:07.20
Pass 2	428	00:00:09.41	00:00:29.42
Symbol table output	2	00:00:00.39	00:00:00.73
Psect synopsis output	2	00:00:00.01	00:00:00.01
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1671	00:00:45.31	00:02:23.10

The working set limit was 3000 pages.
233724 bytes (457 pages) of virtual memory were used to buffer the intermediate code.
There were 160 pages of symbol table space allocated to hold 2613 non-local and 352 local symbols.
4681 source lines were read in Pass 1, producing 49 object records in Pass 2.
105 pages of virtual memory were used to define 97 macros.

! Macro library statistics !

Macro library name	Macros defined
-----	-----
\$255SDUA18:[LAT.OBJ]LAT.MLB;2	1
\$255SDUA18:[SYS.OBJ]LIB.MLB;1	39
\$255SDUA18:[SYSLIB]STARLET.MLB;3	10
TOTALS (all libraries)	50

2819 GETS were required to define 50 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS:LTDRIVER/OBJ=OBS:LTDRIVER MSRCS:LTDRIVER/UPDATE=(BUGS:LTDRIVER)+EXECMLS/LIB+LIBS:LAT/LIB

0444 AH-EF71A-SE
VAX/VMS V4.1 SRC LST MCRF UPD

Grid of source code listings for various modules including:

- LTDRIVER MAP
- LTDRIVER LIS
- TRUNC LIS
- RWUB LIS
- SMALOC LIS
- SNDER LIS
- LAT

