```
LLL                 AAAAAAAAA     TTTTTTTTTTTTTTT
LLL                 AAAAAAAAA     TTTTTTTTTTTTTTT
LLL                 AAAAAAAAA     TTTTTTTTTTTTTTT
LLL              AAA         AAA        TTT
LLL              AAA         AAA        TTT
LLL              AAA         AAA        TTT
LLL              AAA         AAA        TTT
LLL              AAA         AAA        TTT
LLL              AAA         AAA        TTT
LLL              AAA         AAA        TTT
LLL              AAA         AAA        TTT
LLL              AAA         AAA        TTT
LLL              AAAAAAAAAAAAAAA        TTT
LLL              AAAAAAAAAAAAAAA        TTT
LLL              AAAAAAAAAAAAAAA        TTT
LLL              AAA         AAA        TTT
LLL              AAA         AAA        TTT
LLL              AAA         AAA        TTT
LLLLLLLLLLLLLLLL AAA         AAA        TTT
LLLLLLLLLLLLLLLL AAA         AAA        TTT
LLLLLLLLLLLLLLLL AAA         AAA        TTT
```

```
LL          TTTTTTTTTT  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
LL          TTTTTTTTTT  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
LL              TT      DD      DD  RR      RR      II      VV      VV  EE          RR      RR
LL              TT      DD      DD  RR      RR      II      VV      VV  EE          RR      RR
LL              TT      DD      DD  RR      RR      II      VV      VV  EE          RR      RR
LL              TT      DD      DD  RR      RR      II      VV      VV  EE          RR      RR
LL              TT      DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
LL              TT      DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
LL              TT      DD      DD  RR  RR          II      VV      VV  EE          RR  RR
LL              TT      DD      DD  RR  RR          II      VV      VV  EE          RR  RR
LL              TT      DD      DD  RR    RR        II        VV  VV    EE          RR    RR        ....
LL              TT      DD      DD  RR    RR        II        VV  VV    EE          RR    RR        ....
LLLLLLLLLL      TT      DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR      ....
LLLLLLLLLL      TT      DDDDDDDD    RR      RR    IIIIII        VV      EEEEEEEEEE  RR      RR      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

(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$STRENTRY - Entry to start driver
(28)	1371	LT$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$NEW_CIRCUIT - Get Concentrator State Block
(52)	3555	LT$DEALOCSB - 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

```
0000          .TITLE  LTDRIVER - Local Area Terminal Port Driver
0000          .IDENT  'V04-000'
0000
0000    ;*****************************************************************************
0000    ;*                                                                           *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000    ;*  ALL RIGHTS RESERVED.                                                     *
0000    ;*                                                                           *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    ;*  TRANSFERRED.                                                             *
0000    ;*                                                                           *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    ;*  CORPORATION.                                                             *
0000    ;*                                                                           *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    ;*                                                                           *
0000    ;*                                                                           *
0000    ;*****************************************************************************
0000
0000    ; Do not remove items from this list.  Add X when item is done.
0000    ; to be done:
0000    ;
0000    ; - LATCP should be allowed to specify the Ethernet device.
0000    ; - fix slot stop and circuit stop  code for correct message format
0000    ;       including reason code and text.
0000    ; - transmit two slots/message from host transmit loop.
0000    ; - allocate and be able to send additional buffers based about the servers
0000    ;       value of XTRABFRs (in START message).
0000    ; - send credits in DATA_B slots.
0000    ;
```

```
0000
0000  ;
0000  ;*****************************************************************************
0000  ;*                                                                           *
0000  ;*  Copyright (c) 1982, 1984                                                 *
0000  ;*  by digital equipment corporation, maynard, mass.                          *
0000  ;*                                                                           *
0000  ;*  this software is furnished under a license and may be used and  copied   *
0000  ;*  only  in  accordance  with  the  terms  of  such  license and with the   *
0000  ;*  inclusion of the above copyright notice.  this software or  any  other   *
0000  ;*  copies  thereof may not be provided or otherwise made available to any   *
0000  ;*  other person.  no title to and ownership of  the  software  is  hereby   *
0000  ;*  transferred.                                                              *
0000  ;*                                                                           *
0000  ;*  the information in this software is subject to change  without  notice   *
0000  ;*  and  should  not  be  construed  as  a commitment by digital equipment   *
0000  ;*  corporation.                                                              *
0000  ;*                                                                           *
0000  ;*  digital assumes no responsibility for the use or  reliability  of  its   *
0000  ;*  software on equipment which is not supplied by digital.                   *
0000  ;*                                                                           *
0000  ;*****************************************************************************
0000  ;
0000  ;++
0000  ; Facility:
0000  ;
0000  ;       VAX/VMS Terminal Driver
0000  ;
0000  ; Abstract:
0000  ;
0000  ;       Local Area Terminal Port Driver
0000  ;
0000  ;       This is an alternate port driver for use with the VMS terminal
0000  ;       driver.  It forms the host end of the Local Area Terminal (LAT)
0000  ;       protocol and communicates with a LAT concentrator to provide
0000  ;       terminal communications on a local area network.
0000  ;
0000  ;       This driver performs IO using a datalink driver via the FFI
0000  ;       mechanism.  Its connection with the datalink driver is controlled
0000  ;       by a program called LATCP.
0000  ;
0000  ;
0000  ;
0000  ; Authors:
0000  ;
0000  ;       Bruce Mann
0000  ;       Darrell Duffy
0000  ;       Joe Marchesani
0000  ;       Rich Bailey
0000  ;
0000  ; Revision history:
0000  ;
0000  ;       V03-036 RNG0036         Rod Gamache             23-Aug-1984
0000  ;               Fix LT$XOFF to recognize when data input st·am has been
0000  ;               interrupted or whether the call was an asyc  onous event.
0000  ;               Fix problem with XON dropping a character (if AP=2).
0000  ;               Fix problem with output slot processing, that incorrectly
```

```
0000    ;               moved data up in the UCB output buffer area.
0000    ;
0000    ;       V03-035 RNG0035         Rod Gamache             7-Aug-1984
0000    ;               Fix service rating reaching zero.
0000    ;
0000    ;       V03-034 RNG0034         Rod Gamache             3-Aug-1984
0000    ;               Make sure LT UCBs can be deleted by insuring that the
0000    ;               Logical UCB pointer is pointing at the LT UCB and that
0000    ;               the REFC has gone to zero.
0000    ;
0000    ;       V03-033 RNG0033         Rod Gamache             2-Aug-1984
0000    ;               Set service rating to zero if IJOBLIM is reached, else
0000    ;               don't let service rating reach zero.
0000    ;               Clear the hangup bit in the ALLOCATE UCB code rather than
0000    ;               unit init.
0000    ;               Fix code to correctly format a reject slot.
0000    ;
0000    ;       V03-032 RNG0032         Rod Gamache             31-Jul-1984
0000    ;               Change DISCONNECT routine to not use DELETEUCB bit to
0000    ;               force $DASSGN code to delete UCBs, since if there are logical
0000    ;               UCBs (ie VTs), then the LT UCBs will never get deallocated.
0000    ;               Put back code to set HANGUP bit, in DISCONNECT routine.
0000    ;
0000    ;       V03-031 RNG0031         Rod Gamache             30-Jul-1984
0000    ;               Fix previous fix where all uses of CLASS/PORT iterface were
0000    ;               not fixed.
0000    ;
0000    ;       V03-030 RNG0030         Rod Gamache             25-Jul-1984
0000    ;               Fix incorrect useage of CLASS/PORT driver interface for the
0000    ;               DDCMP CLASS DRIVER. Fix credit check on sending DATA_B slot.
0000    ;
0000    ;       V03-029 LMP0275         L. Mark Pilant,         24-Jul-1984  15:49
0000    ;               Initialize the ACL info in the ORB to be a null descriptor
0000    ;               list rather than an empty queue.  This avoids the overhead
0000    ;               of locking and unlocking the ACL mutex, only to find out
0000    ;               that the ACL was empty.
0000    ;
0000    ;       V03-028 RNG0028         Rod Gamache             21-May-1984
0000    ;               Remove call to SIDELINEUCB which caused LT units to remain
0000    ;               "hung". Also fix test of service rating. Change the way
0000    ;               LT units are deleted.
0000    ;               Change use of history buffer.
0000    ;               Remove bug check generated when UCB state is "halted" on
0000    ;               call to LT$DISCONNECT (this happens when user doesn't log
0000    ;               in and unit is disconnected).
0000    ;               Disable UCB deletion if server is doing a BIND request.
0000    ;               Change XOFF and XON routines to save R0, and change number
0000    ;               of characters checked for input buffering from 4 to 1.
0000    ;
0000    ;       V03-027 RNG0027         Rod Gamache             10-May-1984
0000    ;               Change LAT Multicast Address from AB-00-03-00-00-00 to
0000    ;               09-00-2B-00-00-0F, as per LAT Architecture ECO.
0000    ;
0000    ;       V03-026 RNG0026         Rod Gamache             18-Apr-1984
0000    ;               Make better estimate of service rating on a 782 processor.
0000    ;               Always call EXE$DEANONPAGED rather than COM$DRVDEALMEM.
0000    ;               Fixup server name length when needed before copying into CSB.
```

```
0000   ;
0000   ;       V03-025 LMP0234         L. Mark Pilant,         17-Apr-1984  14:04
0000   ;               Copy the ORB from the template rather than zeroing the new
0000   ;               one.
0000   ;
0000   ;       V03-024 RNG0024         Rod Gamache             9-Apr-1984
0000   ;               Add SERVER name to CSB. Count active users on CSB.
0000   ;
0000   ;       V03-023 LMP0221         L. Mark Pilant,         27-Mar-1984  14:49
0000   ;               Change UCB$L_OWNUIC to ORB$L_OWNER and UCB$W_VPROT to
0000   ;               ORB$W_PROT.  Also, allocate and link the ORB into the
0000   ;               UCB at creation.  Deallocate it at deletion.
0000   ;
0000   ;       V04-022 RNG0022         Rod Gamache             22-Mar-1984
0000   ;               Modify algorithm that computes the service rating.
0000   ;               Use XOFF, XON calls to buffer typeahead data.
0000   ;
0000   ;       V04-021 RNG0021         Rod Gamache             19-Mar-1984
0000   ;               Fix call to CLASS driver's SETUP_UCB routine to not
0000   ;               get called for UNIT #0.
0000   ;
0000   ;       V04-019 RNG0019         Rod Gamache             27-Dec-1983
0000   ;               General cleanup for V4 release.
0000   ;
0000   ;--
```

```
0000          .SBTTL  Global Definitions

0000          .ENABLE SUPPRESSION

0000    ;
0000    ;     Global definitions
0000    ;

0000          $LATDEF                         ; LAT communication area defs
0000          $ACBDEF                         ; Ast control block
0000          $ADPDEF                         ; Define adapter
0000          $CRBDEF                         ; Define crb
0000          $CXBDEF                         ; Complex buffer
0000          $DDBDEF                         ; Define ddb
0000          $DEVDEF                         ; Define device characteristics
0000          $DPTDEF                         ; Define dpt
0000          $DYNDEF                         ; Dynamic memory block codes
0000          $FFIDEF                         ; Fast Interface block
0000          $FKBDEF                         ; Fork block definitions
0000          $IPLDEF                         ; IPL levels
0000          $MTXDEF                         ; Mutex offsets
0000          $ORBDEF                         ; Object's Rights Block offsets
0000          $PCBDEF                         ; Process control block
0000          $PHDDEF                         ; Process header block
0000          $PRDEF                          ; Processor register definitions
0000          $RSNDEF                         ; Resouce wait codes
0000          $TQEDEF                         ; Timer queue elements
0000          $UCBDEF                         ; Define ucb
0000          $VECDEF                         ; Define vector for crb

0000          $TTYDEF                         ; Define terminal driver symbols
0000          $TTDEF                          ; Define terminal types
0000          $TT2DEF                         ; Define extended characteristics
0000          $TTYMACS                        ; Define terminal driver macros
0000          $TTYDEFS                        ; Define terminal driver symbols
```

```
                0000   .SBTTL  Local Definitions
                0000 ;
                0000 ;       Macros
                0000 ;
                0000         .MACRO  ALIGN_LONG
                0000         .iif ne <.&3>,  .BLKB  <4-<.&3>> ; align to longword boundary, if needed
                0000         .ENDM   ALIGN_LONG
                0000
                0000 ;
                0000 ;       Conditional assembly parameters
                0000 ;
                0000
00000001        0000         LT_HISTORY      = 1              ; keep history of received/transitted ni fra
                0000 ;;&&    LT_CRED_CHECK   = 1              ; validate credit total for UCBs
                0000 ;;&&    LT_XMT_CHECK    = 1              ; validate transmit frame slots dynamically
                0000
                0000 ;
                0000 ;       Constants defined for the VMS LAT host implementation
                0000 ;
00000007        0000         TTY_C_BELL      = 7              ; Control_G (^G) ... bell character
00000003        0000         LAT$C_VMS       = 3              ; VAX/VMS product type code
00000004        0000         LAT$C_VMS_VER   = 4              ; Part of VAX/VMS V4.0
00000008        0000         LAT$C_IPL       = 8              ; Fork IPL to synchronize this driver
00000014        0000         LAT$C_PROGRESS  = 20             ; Must make progress every 20 msgs
00000002        0000         LAT$C_XMT_BUFFERS = 2            ; Number of transmit buffers
00000001        0000         LAT$C_MAX_RCRED = 1              ; Maximum number of credits extended
00000050        0000         LAT$C_UCB_BUFSIZ = 80            ; Size of UCB buffering
00000004        0000         LAT$C_UCB_SLOTSIZ = 4            ; Size of formatted slot buffer in UCB
00000002        0000         LAT$C_UCB_TIMOUT = 2             ; Allow 2 seconds before deleting  UCB's
00000020        0000         LAT$C_MAX_CSBS  = 32             ; Maximum number of circuits handled
00000040        0000         LAT$C_MAX_SLOTS = 64             ; Maximum slots in a message, this also
                0000                                          ;  implies max connections per circuit
00000005        0000         LAT$C_HI_VER    = 5              ; Highest supported LAT protocol version
00000005        0000         LAT$C_LO_VER    = 5              ; Lowest supported LAT protocol version
00000005        0000         LAT$C_CUR_VER   = 5              ; Current LAT protocol version
00000000        0000         LAT$C_CUR_ECO   = 0              ; Current LAT protocol ECO version
000005DC        0000         LAT$C_MAX_MSGSIZ = 1500          ; Maximum Data Link message size
000000FF        0000         LAT$C_MAX_SLOTSIZ = 255          ; Maximum Slot size
00000040        0000         LAT$C_LOC_LEN   = 64             ; Maximum length of LOCation field
00000020        0000         LAT$C_GRP_LEN   = 32             ; Maximum number of group codes
00000008        0000         LAT$C_SVC_LEN   = 8              ; Maximum number of service classes
00000006        0000         LAT$C_MAX_SERVICES = 6           ; Maximum number of services
00000009        0000         LAT$C_MULTIADD1 = ^X0009         ; LAT Multicast address
0000002B        0000         LAT$C_MULTIADD2 = ^X002B         ;       ...
00000F00        0000         LAT$C_MULTIADD3 = ^X0F00         ;       ...
00000002        0000         LAT$C_HOST_TIMER = 2             ; Host circuit timeout value (in SECs)
0000000A        0000         LAT$C_MULTI_TIMER = 10           ; Multicast message timer
00000050        0000         LAT$C_CIR_TIMER = 80             ; Host preferred circuit timer (in MS)
004C4B40        0000         LAT$C_SHUTDELAY = 5*1000*1000    ; Delay time = 1/2 second
0000000A        0000         LAT$C_KEEP_CSB  = 10             ; Keep last 10 CSBs
                0000
                0000 ;
                0000 ;       Port driver ucb extension to class driver structures
                0000 ;
                0000
                0000         $DEFINI UCB     GLOBAL
00000134        0000         . = UCB$C_TT_LENGTH
```

```
0134          ALIGN_LONG                                  ; align to longword boundary
0134  $DEF    UCB$L_LT_CSB     .BLKL   1                  ; CSB address
0138  $DEF    UCB$L_LT_DEADLINK .BLKL 1                   ; Link for dead ucbs
013C  $DEF    UCB$L_LT_INPBUF  .BLKL   1                  ; Pointer to LAT type ahead buffer
0140  $DEF    UCB$B_LT_REMID   .BLKB   1                  ; Remote slot index number
0141  $DEF    UCB$B_LT_LOCID   .BLKB   1                  ; Local slot index number
0142  $DEF    UCB$B_LT_DATAW   .BLKB   1                  ; Data waiting flags
0143   $VIELD UCB,0,<-
0143          <LT_DATA,1,M>-                              ;* Output data waiting
0143          <LT_FLOW,1,M>-                              ;* Send current flow control values
0143          <LT_ABORT,1,M>-                             ;* Send abort request
0143          >
0143
0143  $DEF    UCB$B_LT_LATSTS .BLKB   1                   ; Status for local terminal connection
0144   $VIELD UCB,0,<-
0144          <LT_HANGUP,1,M>-                            ;* We are hanging up the slot
0144          <LT_DEAD,1,M>-                              ;* We have linked the UCB on deadlist
0144          <LT_OVFLOW,1,M>-                            ;* We had an overflow on input (XOFF)
0144          <LT_BOUND,1,M>-                             ;* Terminal is bound, don't delete UCB
0144          <LT_INPUT,1,M>-                             ;* Input stream in progress
0144          >
0144
0144  $DEF    UCB$B_LT_STATE   .BLKB   1                  ; Slot state
0145   $EQU   UCB$C_LT_STATE_START     1                  ;* State is starting
0145   $EQU   UCB$C_LT_STATE_RUN       2                  ;* State is running
0145   $EQU   UCB$C_LT_STATE_STOP      3                  ;* State is stopping (send stop slot)
0145   $EQU   UCB$C_LT_STATE_KILL      4                  ;* Kill UCB immediately
0145
0145  $DEF    UCB$B_LT_REASON .BLKB   1                   ; Reason for stopping slot
0146  $DEF    UCB$B_LT_TCRED  .BLKB   1                   ; Available credits for this slot
0147  $DEF    UCB$B_LT_XCRED  .BLKB   1                   ; Credits extended to concentrator
0148  $DEF    UCB$B_LT_RCRED  .BLKB   1                   ; Credits to be returned to concentrator
0149  $DEF    UCB$B_LT_SLOTSZ .BLKB   1                   ; Slot size (negotiated)
014A  $DEF    UCB$B_LT_MAXC   .BLKB   1                   ; Size of UCB buffering
014B  $DEF    UCB$B_LT_CURC   .BLKB   1                   ; Current character count
014C  $DEF    UCB$B_LT_CBUF   .BLKB LAT$C_UCB_BUFSIZ ; Startio data buffer
019C  $DEF    UCB$B_LT_EBUF                               ; Startio data buffer end
019C  $DEF    UCB$C_LT_LENGTH                             ; Length of block
019C          $DEFEND UCB
```

```
                 0000
                 0000  ;+
                 0000  ;       Circuit state block
                 0000  ;
                 0000  ;  For each concentrator, one virtual circuit is maintained.
                 0000  ;  Protocol state is maintained in circuit state blocks.
                 0000  ;  The CSB addresses are stored in a vector starting
                 0000  ;  at CSB_TABLE.
                 0000  ;-
                 0000
                 0000          $DEFINI CSB     GLOBAL
00000004         0000                          .BLKL   1
00000008         0004                          .BLKL   1
                 0008  $DEF    CSB_W_SIZE      .BLKW   1       ; Size of this block in bytes
                 000A  $DEF    CSB_B_TYPE      .BLKB   1       ; Type of block
                 000B  $DEF    CSB_B_STATE     .BLKB   1       ; Virtual circuit state:
                 000C   $EQU   CSB_C_STATE_HALT        0       ;* Circuit halted state
                 000C   $EQU   CSB_C_STATE_START       1       ;* Circuit starting state
                 000C   $EQU   CSB_C_STATE_RUN         2       ;* Circuit running state
                 000C  ;
                 000C  ;       Counter area - must follow standard block header!
                 000C  ;
                 000C          ASSUME  GHB$T_CSBCTR EQ .
                 000C  $DEF    CSB_Z_LCB       .BLKB   LCB_C_LENGTH ; Circuit block counter area
                 001C  ;
                 001C  ;       Server name area - must immediately follow counter area
                 001C  ;
                 001C  $DEF    CSB_B_SERVER    .BLKB   1       ; Length of server name
                 001D  $DEF    CSB_T_SERVER    .BLKB   GHB$K_NAMELEN ; Server name field
                 002D  $DEF    CSB_B_REFC      .BLKB   1       ; Number of UCBs attached to this CSB
                 002E  ;
                 002E  ;       Destination Ethernet address - must immediately follow server name area
                 002E  ;
                 002E  $DEF    CSB_Z_DST       .BLKW   3       ; Concentrator physical address
                 0034
                 0034  $DEF    CSB_Q_XBUFQ     .BLKQ   1       ; Current transmit buffer queue
                 003C  $DEF    CSB_Q_XWAITQ    .BLKQ   1       ; Transmit buffers waiting queue
                 0044  $DEF    CSB_L_XCXB      .BLKL   LAT$C_XMT_BUFFERS ; Transmit CXBs in progress
                 004C  $DEF    CSB_L_STOPREASON .BLKL  1       ; Address of the stop reason block
                 0050  $DEF    CSB_W_TIMEOUT   .BLKW   1       ; Timeout cell for CSB
                 0052  $DEF    CSB_W_XMTTMO    .BLKW   1       ; Timeout count for retransmit
                 0054  $DEF    CSB_W_PROGRESS  .BLKW   1       ; Progress counter
                 0056  $DEF    CSB_W_PROGSEQ   .BLKW   1       ; Transmit error control for progress
                 0058  $DEF    CSB_B_XMTBSY    .BLKB   1       ; Transmit busy and count of waiters
                 0059  $DEF    CSB_B_XMTCNT    .BLKB   1       ; Count of buffers to transmit
                 005A  $DEF    CSB_B_XCXB_INX  .BLKB   1       ; XCXB index
                 005B  $DEF    CSB_B_SPARE     .BLKB   1       ; SPARE
                 005C
                 005C  ; Start of circuit header as it appears in transmitted messages
                 005C
                 005C  $DEF    CSB_T_CIRCHDR                   ; Lat circuit header
                 005C  $DEF    CSB_B_FLAG      .BLKB   1       ; Message flags byte
                 005D  $DEF    CSB_B_NUM_SLOTS .BLKB   1       ; Number of slots
                 005E  $DEF    CSB_W_REMID                     ; Remote virtual circuit id
                 005E  $DEF    CSB_B_REMIDN    .BLKB   1       ; Remote virtual circuit id number
                 005F  $DEF    CSB_B_REMIDS    .BLKB   1       ; Remote virtual circuit id sequence
                 0060  $DEF    CSB_W_LOCID                     ; Local virtual circuit id
```

```
0060   $DEF    CSB_B_LOCIDN    .BLKB   1        ; Local virtual circuit id number
0061   $DEF    CSB_B_LOCIDS    .BLKB   1        ; Local virtual circuit id sequence
0062   $DEF    CSB_W_XSEQ                       ; Transmiter error control
0062   $DEF    CSB_B_XSEQ      .BLKB   1        ; Sequence number being transmitted
0063   $DEF    CSB_B_XACK      .BLKB   1        ; Highest sequence number received
0064    $EQU   CSB_S_CIRCHDR   <.-CSB_T_CIRCHDR> ; Size of header
0064
0064   ; End of circuit header
0064
0064   $DEF    CSB_W_RSEQ                       ; Sequence number to receive
0064   $DEF    CSB_B_RSEQ      .BLKB   1        ; Sequence number to receive
0065   $DEF    CSB_B_RACK      .BLKB   1        ; Highest sequence number acknowledged
0066   $DEF    CSB_W_TIMRESET  .BLKW   1        ; Timeout reset value
0068   $DEF    CSB_B_INX       .BLKB   1        ; CSB table index
0069   $DEF    CSB_B_MAX_SLOTS .BLKB   1        ; Maximum number of slots (negotiated)
006A   $DEF    CSB_W_MAX_MSGSIZ .BLKW  1        ; Maximum message size (negotiated)
006C   $DEF    CSB_C_FIXLENGTH                  ; Length of fixed block
006C   ;
006C   ;       UCB list area  (size = 4*min(lat$c_max_slots,server_slot_count) )
006C   ;
006C           ALIGN_LONG                       ; align to longword boundary
006C   $DEF    CSB_L_UCBLST
006C
006C           $DEFEND CSB
```

```
0000
0000  ;+
0000  ;       Terminal overflow input buffer definitions
0000  ;
0000  ;  The terminal overflow buffer is only used when a call is made to
0000  ;  the XOFF routine to stop the input flow. The overflow buffer is
0000  ;  allocated and the rest of the input data is copied to the overflow
0000  ;  buffer. When we are resumed at XON, the data from the input overflow
0000  ;  buffer is dumped into the class driver.
0000  ;
0000  ;-
0000
0000          $DEFINI INB     GLOBAL
0000
0000  $DEF    INB_W_BOFF      .BLKW   1       ; Offset to start of data
0002  $DEF    INB_W_BCNT      .BLKW   1       ; Count of data remaining
0004  $DEF    INB_L_SPARE     .BLKL   1       ; Spare longword
0008  $DEF    INB_W_SIZE      .BLKW   1       ; Size of structure
000A  $DEF    INB_B_TYPE      .BLKB   1       ; Type of structure
000B  $DEF    INB_B_SPARE     .BLKB   1       ; Spare byte
000C  $DEF    INB_C_LENGTH                    ; Size of structure header
000C
000C          $DEFEND INB
0000
```

```
                  0000
                  0000   ;+
                  0000   ;       Receive buffer offsets and bit definitions
                  0000   ;
                  0000   ;  The format of receive and transmit buffers is identical for fields
                  0000   ;  Starting with and following the destination address field. Ease of
                  0000   ;  implemenatation  dictates that the offsets  be defined relative to
                  0000   ;  the start of the buffer.
                  0000   ;
                  0000   ;  These are the offsets into the architecturally defined portion of
                  0000   ;  the receive buffer itself.
                  0000   ;-
                  0000
                  0000           $DEFINI RCV     GLOBAL
                  0000
00000000          0000   .=0
                  0000   $DEF    RCV_T_DATA                      ; Start of data is right here
                  0000   $DEF    RCV_B_FLAG      .BLKB   1       ; Flags byte and message type
                  0001           ;
                  0001           ; Flag bits:
                  0001           ;
                  0001   _VIELD  FLAG,0,<-                       ;* Flags byte bit definitions recv & xmit
                  0001           <RRF,1,M>,-                     ;* Response requested flag
                  0001           <MASTER,1,M>,-                  ;* Remote station always sets this flag
                  0001           <MTYPE,6,M>,-                   ;* Message type bits
                  0001           >
                  0001           ;
                  0001           ; Message types:
                  0001           ;
                  0001    $EQU   MTYP_C_RUN      0               ;* Run message
                  0001    $EQU   MTYP_C_START    1               ;* Start message
                  0001    $EQU   MTYP_C_HALT     2               ;* Halt message
                  0001    $EQU   MTYP_C_CONF     10              ;* Configuration message
                  0001
                  0001   $DEF    RCV_B_NUM_SLOTS .BLKB   1       ; Number of slots in buffer
                  0002   $DEF    RCV_W_DSTID                     ; Destination virtual circuit id
                  0002   $DEF    RCV_B_DSTIDN    .BLKB   1       ; Destination virtual circuit id number
                  0003   $DEF    RCV_B_DSTIDS    .BLKB   1       ; Destination virtual circuit id seq
                  0004   $DEF    RCV_W_SRCID                     ; Source virtual circuit id
                  0004   $DEF    RCV_B_SRCIDN    .BLKB   1       ; Source virtual circuit id number
                  0005   $DEF    RCV_B_SRCIDS    .BLKB   1       ; Source virtual circuit id
                  0006                                           ;  sequence number
                  0006
                  0006   $DEF    RCV_B_SEQ       .BLKB   1       ; Buffer sequence number
                  0007   $DEF    RCV_B_ACK       .BLKB   1       ; Sequence number acknowledgement
                  0008   $DEF    RCV_T_MDATA                     ; Start of type dependent data
                  0008
                  0008           $DEFEND RCV
```

```
0000
0000  ;
0000  ; Define START message offsets (used for both xmits and recvs)
0000  ;
0000          $DEFINI STRT     GLOBAL
0000  $DEF    STRT_W_MSGSIZ    .BLKW   1          ; Data link frame size
0002  $DEF    STRT_B_PVER      .BLKB   1          ; Circuit protocol version
0003  $DEF    STRT_B_PECO      .BLKB   1          ; Circuit Eng. change order level
0004  $DEF    STRT_B_MAXSLOTS  .BLKB   1          ; Maximum simultaneous slots
0005  $DEF    STRT_B_DL_BFRS   .BLKB   1          ; Number of extra rcv bfrs in server
0006  $DEF    STRT_B_CIR_TIMR  .BLKB   1          ; Local circuit timer in milliseconds
0007  $DEF    STRT_B_KPA_TIMR  .BLKB   1          ; Local keep alive timer in seconds
0008  $DEF    STRT_W_FAC_NUM   .BLKW   1          ; Facility number
000A  $DEF    STRT_W_PROD_TYP  .BLKW   1          ; Product type code
000C  $DEF    STRT_C_LENGTH                       ; Length of fixed portion of message
000C  ;
000C  ; Start of variable length parameters
000C  ;
000C  $DEF    STRT_B_VAR                  ; Start of variable portion of start message
000C  $DEF    STRT_B_NODE_LEN  .BLKB   1          ; Destination node name length
000D  $DEF    STRT_T_NODE      .BLKB   GHB$K_NAMELEN ; Destination node name text
001D  $DEF    STRT_B_SYS_LEN   .BLKB   1          ; Source node name length
001E  $DEF    STRT_T_SYS       .BLKB   GHB$K_NAMELEN ; Source node name text
002E  $DEF    STRT_B_ID_LEN    .BLKB   1          ; Source node descriptor length
002F  $DEF    STRT_T_ID        .BLKB   GHB$K_IDLEN ; Source node descriptor text
006F  $DEF    STRT_B_LOC_LEN   .BLKB   1          ; Source location length
0070  $DEF    STRT_T_LOC       .BLKB   LAT$C_LOC_LEN ; Source location text
00B0  $DEF    STRT_B_PARAM     .BLKB   1          ; Parameter code
00B1   $EQU   STRT_C_VAR_LEN   <.-STRT_B_VAR>     ; Length of variable portion of start
00B1                                              ;   message
00B1          $DEFEND STRT
0000
0000  ;
0000  ; Define STOP message offsets
0000  ;
0000          $DEFINI STOP     GLOBAL
0000  $DEF    STOP_B_RCODE     .BLKB              ; Circuit disconnect reason
0001  $DEF    STOP_B_RLEN      .BLKB              ; Reason length in bytes
0002  $DEF    STOP_T_REAS                         ; Reason ASCII text
0002          $DEFEND STOP
0000
```

```
                0000
                0000  ;+
                0000  ;       Transmit buffer offsets and bit definitions
                0000  ;
                0000  ;       The buffer header is defined to be the same as that used by
                0000  ;       the Ethernet datalink drivers.
                0000  ;-
                0000
                0000          $DEFINI CXB     GLOBAL          ; Transmit complex buffer
                0000  ;
                0000  ; Define the transmit/receive offsets in the CXB
                0000  ;
0000001C        0000  . = CXB$L_END_ACTION
                001C  $DEF    CXB$L_T_ENDADR  .BLKL   1       ; End address of data to transmit
                0020
0000000C        0020  . = CXB$W_LENGTH
                000C          ASSUME  CXB$W_OFFSET EQ CXB$W_LENGTH+2
                000C  $DEF    CXB$L_T_CSB                     ; CSB address
                000C
00000014        000C  . = CXB$L_IRP
                0014  $DEF    CXB$L_T_SAVE    .BLKL   1       ; BOFF & BCNT save area
                0018
                0018          $DEFEND CXB
                0000
                0000  ;
                0000  ; Start of architecturally defined offsets
                0000  ;
                0000          $DEFINI XMT     GLOBAL
                0000
0000003A        0000  . = CXB$C_HEADER - <6+6+2>
                003A  $DEF    XMT_G_DST       .BLKW   3       ; Destination address
                0040
00000048        0040  . = CXB$C_HEADER
                0048  $DEF    XMT_T_DATA                      ; Start of transmitted data
                0048  $DEF    XMT_B_FLAG      .BLKB   1       ; Flags byte and message type
                0049
                0049  ;
                0049  ; Run message format:
                0049  ;
                0049  $DEF    XMT_B_NUM_SLOTS .BLKB   1       ; Number of slots in buffer
                004A  $DEF    XMT_W_DSTID                     ; Destination virtual circuit id
                004A  $DEF    XMT_B_DSTIDN    .BLKB   1       ; Destination virtual circuit id number
                004B  $DEF    XMT_B_DSTIDS    .BLKB   1       ; Destination virtual circuit id seq
                004C  $DEF    XMT_W_SRCID                     ; Source virtual circuit id
                004C  $DEF    XMT_B_SRCIDN    .BLKB   1       ; Source virtual circuit id number
                004D  $DEF    XMT_B_SRCIDS    .BLKB   1       ; Source virtual circuit id
                004E                                          ;   sequence number
                004E
                004E  $DEF    XMT_B_SEQ       .BLKB   1       ; Buffer sequence number
                004F  $DEF    XMT_B_ACK       .BLKB   1       ; Sequence number acknowledgement
                0050  $DEF    XMT_T_MDATA                     ; Start of RUN message data
                0050
                0050  ;
                0050  ; Multicast (CONFIGURATION) message format:
                0050  ;
00000049        0050          . = XMT_B_FLAG+1
                0049  $DEF    XMT_B_MC_CIR_TIMER      .BLKB   1       ; Circuit timer
```

```
004A    $DEF    XMT_B_MC_HI_VER         .BLKB   1       ; Highest supported protocol version
004B    $DEF    XMT_B_MC_LO_VER         .BLKB   1       ; Lowest supported protocol version
004C    $DEF    XMT_B_MC_CUR_VER        .BLKB   1       ; version of this message
004D    $DEF    XMT_B_MC_CUR_ECO        .BLKB   1       ; eco of this message
004E    $DEF    XMT_B_MC_INCARN         .BLKB   1       ; Message incarnation
004F    $DEF    XMT_B_MC_CHG_FLAG       .BLKB   1       ; Changed fields in this msg
0050
0050            VIELD  XMT_CHFLG,0,<-
0050            <GROUP,1,M>,-                           ;* NODE group codes changed
0050            <NDESC,1,M>,-                           ;* NODE descriptor changed
0050            <SVCNAM,1,M>,-                          ;* Service name/number changed
0050            <RATE,1,M>,-                            ;* Service ratings changed
0050            <SDESC,1,M>,-                           ;* Service classes changed
0050            <CLASS,1,M>,-                           ;* Service classes changed
0050            <,1>,-                                  ;* RESERVED
0050            <OTHER,1,M>,-                           ;* One of the OTHER fields changed
0050            >
0050     $EQU   XMT_CHFLG_M_ALL <^XFF>                  ;* All codes changed
0050
0050    $DEF    XMT_W_MC_MSG_SIZ        .BLKW   1       ; Maximum frame size
0052     $EQU   XMT_C_MC_LENGTH         <.-XMT_T_DATA>  ; Size of fixed part of multicast me
0052
0052    $DEF    XMT_B_MC_SET            ; Start of LATCP set portion of multicast msg
0052    $DEF    XMT_B_MC_MULTIMR        .BLKB   1       ; Multicast message timer
0053    $DEF    XMT_B_MC_STATUS         .BLKB   1       ; Node status
0054    $DEF    XMT_B_MC_GRP_LEN        .BLKB   1       ; Number of group codes
0055    $DEF    XMT_T_MC_GROUP          .BLKB   LAT$C_GRP_LEN ; Group code field
0075    $DEF    XMT_B_MC_NODE_LEN       .BLKB   1       ; Node name length
0076    $DEF    XMT_T_MC_NODE           .BLKB   GHB$K_NAMELEN ; NODE name
0086    $DEF    XMT_B_MC_NUM_SVCS       .BLKB   1       ; Number of services
0087
0087    $DEF    XMT_B_MC_RATE           .BLKB   1       ; Service rating
0088    $DEF    XMT_B_MC_NAME_LEN       .BLKB   1       ; Service name length
0089    $DEF    XMT_T_MC_NAME           .BLKB   GHB$K_NAMELEN ; Service name
0099    $DEF    XMT_B_MC_ID_LEN         .BLKB   1       ; Length of Identification field
009A    $DEF    XMT_T_MC_ID             .BLKB   GHB$K_IDLEN ; Identification field
00DA     $EQU   XMT_C_MC_SVC_SIZE       <.-XMT_B_MC_RATE> ; Size of each service field
00DA
00DA    $DEF    XMT_T_MC_MORE_SVC       .BLKB   <<LAT$C_MAX_SERVICES-1>*XMT_C_MC_SVC_SIZE>
0279
0279    $DEF    XMT_B_MC_SVC_LEN        .BLKB   1       ; Number of service classes
027A    $DEF    XMT_B_MC_SVC            .BLKB   LAT$C_SVC_LEN ; Service classes
0282     $EQU   XMT_C_MC_TOTAL  <.-XMT_T_DATA>          ; Length of total message
0282            ASSUME  XMT_C_MC_TOTAL LE LAT$C_MAX_MSGSIZ
0282
0282     $EQU   XMT_C_HDRLEN    <XMT_T_MDATA - XMT_T_DATA>
0282                                                    ; Length of header
0282     $EQU   XMT_C_MAXDATA   <LAT$C_MAX_MSGSIZ - XMT_C_HDRLEN>
0282                                                    ; Data in slots
0282     $EQU   XMT_C_MAXLEN    <LAT$C_MAX_MSGSIZ + XMT_T_DATA>
0282                                                    ; Total size of a buffer
0282            $DEFEND XMT
```

```
0000   ;+
0000   ;       Buffer slot offsets
0000   ;
0000   ;       Each slot is formatted identically in both
0000   ;       transmit and receive buffers as defined below.
0000   ;-

0000           $DEFINI SLT     GLOBAL

0000   $DEF    SLT_T_START                     ; Start of slot
0000   $DEF    SLT_B_DSTID     .BLKB   1       ; Destination slot ID
0001   $DEF    SLT_B_SRCID     .BLKB   1       ; Source slot ID
0002   $DEF    SLT_B_COUNT     .BLKB   1       ; Character count in slot
0003   $DEF    SLT_B_REAS                      ; Stop/reject slot reason (low nibble)
0003   $DEF    SLT_B_CRED                      ; Credits extended (low order nibble)
0003   $DEF    SLT_B_TYPE      .BLKB   1       ; Slot type (high order nibble)
0004           ; STOP/REJECT reason codes:
0004    $EQU   SLT_C_USD       1               ; * User requested disconnect
0004    $EQU   SLT_C_SYS       2               ; * System shutdown
0004    $EQU   SLT_C_INVSLOT   3               ; * Invalid slot received
0004    $EQU   SLT_C_INVSVC    4               ; * Invalid service class received
0004    $EQU   SLT_C_RESOURCE  5               ; * Insufficient resources
0004           ; Slot type codes:
0004    $EQU   SLT_C_DA_SLOT   <^X00>          ; * DATA_A slot
0004    $EQU   SLT_C_STR_SLOT  <^X09>          ; * START slot
0004    $EQU   SLT_C_DB_SLOT   <^X0A>          ; * DATA_B slot
0004    $EQU   SLT_C_ATT_SLOT  <^X0B>          ; * ATTENTION slot
0004    $EQU   SLT_C_REJ_SLOT  <^X0C>          ; * REJECT slot
0004    $EQU   SLT_C_STP_SLOT  <^X0D>          ; * STOP slot
0004   $DEF    SLT_T_DATA                      ; Start of slot data

0004   ;
0004   ; Start slot data definitions
0004   ;
00000004  0004  . = SLT_T_DATA
0004   $DEF    SLT_B_SVCC      .BLKB   1       ; Service class
0005   $EQU    SLT_C_SVC_INTT  1               ; * Interactive Terminals
0005   $DEF    SLT_B_AT_SLTSIZ .BLKB   1       ; Maximum length of attention slot
0006   $DEF    SLT_B_DT_SLTSIZ .BLKB   1       ; Maximum length of Data slot
0007   $DEF    SLT_B_DST_NAMLEN .BLKB  1       ; Length of destination slot name
0008   $DEF    SLT_T_DST_NAME  .BLKB   GHB$K_NAMELEN
0018   $DEF    SLT_B_SRC_NAMLEN .BLKB  1       ; Length of source slot name
0019   $DEF    SLT_B_SRC_NAME  .BLKB   GHB$K_NAMELEN
0029    $EQU   SLT_S_START_SLOT <.-SLT_T_DATA> ; Length of Start slot
0029   $DEF    SLT_B_PARAMS                    ; Start of parameter area
0029    $EQU   SLT_C_START_FLAG        1       ;* Start slot parameter flag is code 1
0029           _VIELD  SLT_FLAG,0,<-
0029                   <REMOTE,1,M>,-          ;* Remote line
0029                   <LOGIN,1,M>,-           ;* Disable auto-login
0029                   <BIND,1,M>,-            ;* Terminal UCB is bound to server
0029                   >

0029   ;
0029   ; DATA_B slot data definitions
0029   ;
00000004  0029  . = SLT_T_DATA
0004   $DEF    SLT_B_DB_CONTROL        .BLKB   1       ; DATA_B slot control flag
```

```
                0005   678          _VIELD  SLT_DB,0,<-
                0005   679                  <IFENA,1,M>,-                   ; Turn on flow control of keyboard i
                0005   680                  <IFDIS,1,M>,-                   ; Turn off flow control of keyb inpu
                0005   681                  <OFENA,1,M>,-                   ; Output flow on
                0005   682                  <OFDIS,1,M>,-                   ; Output flow off
                0005   683                  <BREAK,1,M>,-                   ; Break condition detected (recv slo
                0005   684                  >
                0005   685 $DEF     SLT_B_DB_OFOF    .BLKB   1              ; Output-off flow char (from keyboar
                0006   686 $DEF     SLT_B_DB_OFON    .BLKB   1              ; Output-on flow char (from keyboard
                0007   687 $DEF     SLT_B_DB_IFOF    .BLKB   1              ; Input-off flow character (from hos
                0008   688 $DEF     SLT_B_DB_IFON    .BLKB   1              ; Input-on flow character (from host
                0009   689  $EQU    SLT_S_DB_LEN     <.-SLT_T_DATA>         ; Length of DATA_B data
                0009   690  $EQU    SLT_S_DB_SLOT    <.-SLT_T_START>        ; Length of DATA_B slot
                0009   691
                0009   692 ;
                0009   693 ; ATTention slot data definitions
                0009   694 ;
00000004        0009   695 . = SLT_T_DATA
                0004   696 $DEF     SLT_B_ATT_CONTROL        .BLKB   1     ; ATTENTION slot control flag
                0005   697          _VIELD  SLT_ATT,0,<-
                0005   698                  <,5>,-                          ; RESERVED
                0005   699                  <ABORT,1,M>,-                   ; Abort (flush pipe)
                0005   700                  >
                0005   701  $EQU    SLT_S_ATT_LEN    <.-SLT_T_DATA>         ; Length of Attention data
                0005   702  $EQU    SLT_S_ATT_SLOT   <.-SLT_T_START>        ; Length of Attention slot
                0005   703
                0005   704 ;
                0005   705 ; STOP slot data definitions
                0005   706 ;
00000004        0005   707 . = SLT_T_DATA
                0004   708 $DEF     SLT_B_STP_RLEN   .BLKB   1              ; STOP slot reason length
                0005   709  $EQU    SLT_S_STP_LEN    <.-SLT_T_DATA>         ; Length of fixed STOP slot data
                0005   710  $EQU    SLT_S_STP_SLOT   <.-SLT_T_START>        ; Length of fixed STOP slot
                0005   711 $DEF     SLT_T_STP_REAS                          ; STOP slot reason (variable)
                0005   712
                0005   713 ;
                0005   714 ; REJect slot data definitions
                0005   715 ;
00000004        0005   716 . = SLT_T_DATA
                0004   717 $DEF     SLT_B_REJ_RLEN   .BLKB   1              ; REJ slot reason length
                0005   718  $EQU    SLT_S_REJ_LEN    <.-SLT_T_DATA>         ; Length of fixed REJ slot data
                0005   719  $EQU    SLT_S_REJ_SLOT   <.-SLT_T_START>        ; Length of fixed REJ slot
                0005   720 $DEF     SLT_T_REJ_REAS                          ; REJ slot reason (variable)
                0005   721
                0005   722          $DEFEND SLT
```

```
0000   ;
0000   ;       Local macros
0000   ;

0000   .MACRO  INC_CTR,        BASE, TYPE=L, ?L
0000           INC'TYPE        BASE                        ; Increment counter
0000           BNEQ    L                                   ; Br if no overflow
0000           DEC'TYPE        BASE                        ; Else, latch at maximum
0000   L:
0000   .END    INC_CTR

0000   .MACRO  ADD_CTR,        BASE1, BASE2, TYPE=L, ?L
0000           ADD'TYPE        BASE1,BASE2                 ; Add counters
0000           BCC     L                                   ; Br if no overflow
0000           MNEG'TYPE       #1,BASE2                    ; Else, latch at maximum
0000   L:
0000   .END    ADD_CTR

0000   .MACRO  $DISPATCH,      INDX,VECTOR,TYPE=W,NMODE=S^#,?MN,?MX,?S,?SS,?ZZ
0000   SS:

0000           .MACRO  $DSP1,$DSP1_1
0000                   .IRP    $DSP1_2,$DSP1_1
0000                           $DSP2   $DSP1_2
0000                   .ENDR
0000           .ENDM   $DSP1

0000           .MACRO  $DSP2,$DSP2_1,$DSP2_2
0000                   .=<$DSP2_1-MN>*2 + S
0000                   .WORD   $DSP2_2-S
0000           .ENDM   $DSP2

0000           .MACRO  $BND1,$BND1_1,$BND1_2,$BND1_3
0000                   $BND2   $BND1_1,$BND1_2
0000           .ENDM   $BND1

0000           .MACRO  $BND2,$BND2_1,$BND2_2
0000                   .IIF    $BND2_1,$BND2_2-.,      .=$BND2_2
0000           .ENDM   $BND2

0000           .MACRO  $BND    $BND_1,$BND_2
0000                   .IRP    $BND_3,<$BND_2>
0000                           $BNDT   $BND_1,$BND_3
0000                   .ENDR
0000           .ENDM   $BND

0000           .=0
0000   ZZ:
0000           $BND    GT,<VECTOR>
0000   MX:
0000           $BND    LT,<VECTOR>
0000   MN:
0000           .=SS

0000   CASE'TYPE       INDX,#<MN-ZZ>,NMODE'<MX-MN>
0000   S:
```

```
0000          .REPT   MX-MN+1
0000          .WORD   <MX-MN>*2 + 2
0000          .ENDR
0000
0000          .=S
0000
0000          $DSP1   <<VECTOR>>
0000
0000          .=<MX-MN>*2 + S + 2
0000
0000  .ENDM   $DISPATCH
0000
0000
0000          .MACRO  SETBIT BIT, BASE, ?L    ; Set specified bit
0000          BBSS    BIT, BASE, L
0000  L:
0000          .ENDM
0000
0000          .MACRO  CLRBIT BIT, BASE, ?L    ; Clear specified bit
0000          BBCC    BIT, BASE, L
0000  L:
0000          .ENDM
0000
0000          .MACRO  BREAK
0000          JSB     G^INI$BRK
0000          .ENDM
0000
0000          .MACRO  DEBUG
0000          TSTL    @#-1
0000          .ENDM   DEBUG
0000
```

```
0000          .SBTTL  Driver Prologue Table
0000
00000000      .PSECT  $$$105_PROLOGUE
0000
0000    ;
0000    ; Driver prologue table:
0000    ;
0000
0000    LT$DPT::                                        ; Driver start
0000          DPTAB   -                                 ; Driver prologue table
0000                  END=LT$END,-                      ; Addres    ' end of module
0000                  UCBSIZE=UCB$C_LT_LENGTH,-; Size o1 each ucb of port driver
0000                  ADAPTER=NULL,-                    ; Adapter type
0000                  NAME=LTDRIVER,-                   ; Name of driver
0000                  VECTOR=LT$VECTOR                  ; Port driver vector table
0038          DPT_STORE INIT
0038          DPT_STORE UCB,UCB$B_FIPL,B,LAT$C_IPL ; Fork ipl
003C          DPT_STORE UCB,UCB$L_DEVCHAR,L,<-; Characteristics
003C                            DEV$M_REC!-     ;
003C                            DEV$M_AVL!-     ;
003C                            DEV$M_IDV!-     ;
003C                            DEV$M_ODV!-     ;
003C                            DEV$M_TRM!-     ;
003C                            DEV$M_CCL>
0043          DPT_STORE UCB,UCB$L_DEVCHAR2,L,<-
0043                            DEV$M_NNM>      ; Prefix name with <node$>
004A          DPT_STORE UCB,UCB$B_DEVCLASS,B,DC$_TERM;
004E          DPT_STORE UCB,UCB$B_TT_DETYPE,B,TT$_UNKNOWN     ; Type
0052          DPT_STORE UCB,UCB$W_TT_DESIZE,aW,TTY$GW_DEFBUF  ; Buffer size
0059          DPT_STORE UCB,UCB$L_TT_DECHAR,aL,TTY$GL_DEFCHAR ; Default characters
0060          DPT_STORE UCB,UCB$L_TT_DECHA1,aL,TTY$GL_DEFCHAR2; Default characters
0067          DPT_STORE UCB,UCB$W_TT_DESPEE,aB,TTY$GB_DEFSPEED; Default speed
006E          DPT_STORE UCB,UCB$W_TT_DESPEE+1,aB,TTY$GB_RSPEED; Default rspeed
0075          DPT_STORE UCB,UCB$B_TT_DEPARI,aB,TTY$GB_PARITY  ; Default parity
007C    ;**   DPT_STORE UCB,UCB$B_TT_PARITY,aB,TTY$GB_PARITY  ; Default parity
007C          DPT_STORE UCB,UCB$B_DEVTYPE,B,TT$_UNKNOWN       ; Type
0080          DPT_STORE UCB,UCB$W_DEVBUFSIZ,aW,TTY$GW_DEFBUF  ; Buffer size
0087          DPT_STORE UCB,UCB$L_DEVDEPEND,aL,TTY$GL_DEFCHAR ; Default characters
008E          DPT_STORE UCB,UCB$L_TT_DEVDP1,aL,TTY$GL_DEFCHAR2; Default characters
0095          DPT_STORE UCB,UCB$W_TT_SPEED,aB,TTY$GB_DEFSPEED ; Default speed
009C          DPT_STORE UCB,UCB$W_TT_SPEED+1,aB,TTY$GB_RSPEED ; Default rspeed
00A3          DPT_STORE UCB,UCB$B_DIPL,B,LAT$C_IPL            ; Device ipl
00A7          DPT_STORE UCB,UCB$L_TT_WFLINK,L,0               ; Zero write queue.
00AE          DPT_STORE UCB,UCB$L_TT_WBLINK,L,0               ; Zero write queue.
00B5          DPT_STORE UCB,UCB$L_TT_RTIMOU,L,0               ; Zero read timed out
00BC    ;**   DPT_STORE UCB,UCB$W_TT_PRTCTL,W,TTY$M_PC_NOTIME ; Disable all timers
00BC          DPT_STORE ORB,ORB$B_FLAGS,B,-                   ; Default flags
00BC                            <ORB$M_PROT_16>               ; SOGW word protection
00C0          DPT_STORE ORB,ORB$W_PROT,aW,TTY$GW_PROT         ; Default protection
00C7          DPT_STORE ORB,ORB$L_OWNER,aL,TTY$GL_OWNUIC      ; Default owner uic
00CE          DPT_STORE DDB,DDB$L_DDT,D,LT$DDT
00D3
00D3          DPT_STORE REINIT
00D3          DPT_STORE CRB,CRB$L_INTD+VEC$L_INITIAL,D,LT$CTRL_INIT ; Controller init
00D8          DPT_STORE CRB,CRB$L_INTD+VEC$L_UNITINIT,D,LT$UNIT_INIT ; Unit init
00DD          DPT_STORE END
0000
```

```
0000   870
0000   871
0000   872
0000   873
0000   874          DDTAB   DEVNAM = LT,-    ; Dummy port driver dispatch table
0000   875                  START  = 0,-
0000   876                  FUNCTB = 0
0038   877
```

```
                0038          .SBTTL  Local Storage
                0038
                0038
         00000038          .PSECT  $$$115_DRIVER,LONG
                0038
                0038 ;
                0038 ; Local storage
                0038 ;
                0038
                0038 ;+
                0038 ;       Circuit State Block table
                0038 ;
                0038 ;  This table is indexed by RCV_B_DSTIDN in received buffers. A
                0038 ;  concentrator always uses the same value when refering to a
                0038 ;  host for a second time. The RCV_B_DSTIDS byte is a check byte
                0038 ;  which helps prevent random or outdated references to CSBs.
                0038 ;
                0038 ;-
                0038
                0038          .ALIGN  LONG
                0038
                0038 GHB_L_CSBTABLE:
000000B8        0038          .BLKL   LAT$C_MAX_CSBS                  ; CSB address table
                00B8 GHB_B_LOC_CCHK:                                  ; Table of local
000000D9        00B8          .BLKB   LAT$C_MAX_CSBS+1                ; circuit checks
                00D9
                00D9
                00D9 ;
                00D9 ;       Timer queue element
                00D9 ;
                00D9 ;       Used to timeout CSB's that have had no activity.
                00D9 ;
                00D9
                00D9          .ALIGN  LONG
                00DC GHB_G_TQE:
0000010C        00DC          .BLKB   TQE$C_LENGTH
                010C
```

```
                010C
                010C  ;
                010C  ;       Host counter block
                010C  ;
                010C
                010C          .ALIGN  LONG
                010C  GHB_AREA:                         ; Communications area
0000016C        010C          .BLKB   GHB$S_COMM_AREA
                016C
                016C          .ALIGN LONG
                016C
00000170        016C  GHB_L_UCB0:     .BLKL   1         ; LT UCB zero address
00000174        0170  GHB_L_DEADLINK: .BLKL   1         ; List of dead UCBs
00000178        0174  GHB_L_FFI:      .BLKL   1         ; FFI block address
0000017C        0178  GHB_L_NULLCPU:  .BLKL   1         ; Pointer to null CPU time
00000180        017C  GHB_L_LASTCPU:  .BLKL   1         ; Value of last null CPU sample
00000184        0180  GHB_L_NULLSEC:  .BLKL   1         ; Value of latt null CPU sample on secondary
00000185        0184  GHB_B_STATUS:   .BLKB   1         ; Host status byte
                0185          _VIELD  GHB_STS,0,<-
                0185                  <MULTI,1,M>,-     ; Multicast buffer in use
                0185                  <ACTIVE,1,M>,-    ; LTDRIVER is active
                0185                  <SHUT,1,M>,-      ; Shutdown is in progress
                0185                  <TQE,1,M>,-       ; Waiting for TQE to fire
                0185                  >
00000186        0185  GHB_B_INCARN:   .BLKB   1         ; Incarnation number for multicast message
00000187        0186  GHB_B_OLD_CSBCNT: .BLKB 1         ; Count of old CSBs on queue
00000188        0187  GHB_B_RATE:     .BLKB   1         ; Initial rating computed from memory size
                0188  ;
                0188  ; Multicast data set by LATCP
                0188  ;
                0188          .ALIGN  LONG
0000018A        0188  GHB_W_MC_SIZE:  .BLKW   1         ; Size of name and ID strings
000003D4        018A  GHB_T_MC_DATA:  .BLKB   XMT_C_MC_TOTAL+16 ; Name and ID storage + fudge factor
                03D4
                03D4  ;
                03D4  ; Multicast (Configuration) Message Destination Ethernet address
                03D4  ;
                03D4          .ALIGN  LONG
    0009        03D4  GHB_Q_MULTIADD: .WORD   LAT$C_MULTIADD1 ; LAT Multicast destination address
    002B        03D6                  .WORD   LAT$C_MULTIADD2 ;       ...
    0F00        03D8                  .WORD   LAT$C_MULTIADD3 ;       ...
                03DA                                          ; Picks up next word also (garbage)
                03DA  ;
                03DA  ; Data to control multicast buffer
                03DA  ;
000003DC        03DA  GHB_W_MULTIMR:  .BLKW   1               ; Timer for multicast transmits
000003E0        03DC  GHB_L_MULTIBFR: .BLKL   1               ; Address of the multicast buffer
                03E0
                03E0  ;
                03E0  ; Host Multicast (Configuration) Message fixed portion template
                03E0  ;       (Fields are order-dependent and adjacent)
                03E0  ;
                03E0          ASSUME  XMT_B_FLAG          EQ  XMT_T_DATA
                03E0          ASSUME  XMT_B_MC_CIR_TIMER  EQ  XMT_B_FLAG+1
                03E0          ASSUME  XMT_B_MC_HI_VER     EQ  XMT_B_MC_CIR_TIMER+1
                03E0          ASSUME  XMT_B_MC_LO_VER     EQ  XMT_B_MC_HI_VER+1
                03E0          ASSUME  XMT_B_MC_CUR_VER    EQ  XMT_B_MC_LO_VER+1
```

```
          03E0          ASSUME  XMT_B_MC_CUR_ECO    EQ  XMT_B_MC_CUR_VER+1
          03E0          ASSUME  XMT_B_MC_INCARN     EQ  XMT_B_MC_CUR_ECO+1
          03E0          ASSUME  XMT_B_MC_CHG_FLAG   EQ  XMT_B_MC_INCARN+1
          03E0          ASSUME  XMT_W_MC_MSG_SIZ    EQ  XMT_B_MC_CHG_FLAG+1
          03E0          ASSUME  XMT_B_MC_SET        EQ  XMT_W_MC_MSG_SIZ+2
          03E0  GHB_T_MC_MSG:                           ; Start of multicast msg fixed data
    28    03E0                  .BYTE   MTYP_C_CONF@FLAG_V_MTYPE ; Message type
    08    03E1                  .BYTE   <LAT$C_CIR_TIMER*95/10  ; Host preferred Circuit timer
    05    03E2                  .BYTE   LAT$C_HI_VER    ; Highest LAT protocol version supported
    05    03E3                  .BYTE   LAT$C_LO_VER    ; Lowest LAT protocol version supported
    05    03E4                  .BYTE   LAT$C_CUR_VER   ; Current LAT protocol version
    00    03E5                  .BYTE   LAT$C_CUR_ECO   ; Current LAT protocol ECO version
    00    03E6                  .BYTE   0               ; Message incarnation (filled in later)
    00    03E7                  .BYTE   0               ; Change flags (filled in later)
  05DC    03E8                  .WORD   LAT$C_MAX_MSGSIZ ; Maximum message size
          03EA
          03EA  ;
          03EA  ; Standard Start message (fixed portion) template
          03EA  ;
          03EA          ASSUME  STRT_W_MSGSIZ   EQ  0
          03EA          ASSUME  STRT_B_PVER     EQ  STRT_W_MSGSIZ+2
          03EA          ASSUME  STRT_B_PECO     EQ  STRT_B_PVER+1
          03EA          ASSUME  STRT_B_MAXSLOTS EQ  STRT_B_PECO+1
          03EA          ASSUME  STRT_B_DL_BFRS  EQ  STRT_B_MAXSLOTS+1
          03EA          ASSUME  STRT_B_CIR_TIMR EQ  STRT_B_DL_BFRS+1
          03EA          ASSUME  STRT_B_KPA_TIMR EQ  STRT_B_CIR_TIMR+1
          03EA          ASSUME  STRT_W_FAC_NUM  EQ  STRT_B_KPA_TIMR+1
          03EA          ASSUME  STRT_W_PROD_TYP EQ  STRT_W_FAC_NUM+2
000003EC  03EA  GHB_W_STRT_LEN: .BLKW   1               ; Length of start message
          03EC  GHB_T_STRT_MSG:                         ; Start of start message fixed data
  05DC    03EC                  .WORD   LAT$C_MAX_MSGSIZ ; Maximum Ethernet frame size
    05    03EE                  .BYTE   LAT$C_CUR_VER   ; Current LAT protocol version
    00    03EF                  .BYTE   LAT$C_CUR_ECO   ; Current LAT ECO version
    40    03F0                  .BYTE   LAT$C_MAX_SLOTS ; Maximum number of Slots per circuit
    0C    03F1                  .BYTE   0               ; Number of additional buffers (remote)
000003F4  03F2                  .BLKB   2               ; Remote parameters only
  0000    03F4                  .WORD   0               ; Facility number (not yet implemented)
  0003    03F6                  .WORD   LAT$C_VMS       ; Product type code
          03F8
          03F8          ASSUME  STRT_B_VAR   EQ  STRT_W_PROD_TYP+2
0000049D  03F8  GHB_T_STRT_VAR: .BLKB   STRT_C_VAR_LEN  ; Variable portion of start message
          049D  ;
          049D  ; Standard Start slot template
          049D  ;
          049D  GHB_T_STRT_SLOT:                        ; Start of the start slot data
    01    049D                  .BYTE   SLT_C_SVC_INTT  ; Service class
    FF    049E                  .BYTE   LAT$C_MAX_SLOTSIZ ; Attention slot size
    FF    049F                  .BYTE   LAT$C_MAX_SLOTSIZ ; Data slot size
000004A2  04A0                  .BLKB   2               ; Null source and destination slot names
00000005  04A2          GHB_C_STRT_SLOTL = <.-GHB_T_STRT_SLOT>
```

```
                04A2          .SBTTL  Port Vector
                04A2
                04A2  ;
                04A2  ; The associated class driver uses this table to command the port driver.
                04A2  ; The address of this table is contained in the terminal ucb extension area.
                04A2  ; The offset definitons are defined by ttydefs.
                04A2  ;
                04A2
                04A2  ;
                04A2  ;       LAT specific dispatch table
                04A2  ;
                04A2  ; ORDER DEPENDENCY
                04A2  ;  The following symbol must immediately precede the vector
                04A2  ;
                04A2
                04A2          .ALIGN  LONG
                04A4  GHB_L_AREA:
000004A8        04A4          .BLKL   1                               ; Address of counter vector
                04A8  LT$VECTOR:
                04A8          $VECINI LT,LT$NULL
                04E0          $VEC    STARTIO,LT$STARTIO              ; Start new output
                04AC          $VEC    DISCONNECT,LT$DISCONNECT        ; Hangup port
                04B0          $VEC    ABORT,LT$ABORT                  ; Abort output
                04CC          $VEC    SET_LINE,LT$FLOW_CHANGE         ; Change flow control
                04B4          $VEC    XON,LT$XON                      ; Send xon sequence
                04BC          $VEC    XOFF,LT$XOFF                    ; Send xoff sequence
                04C0  ;       $VEC    DS_SET,LT$DS_SET                ; Dataset transitions
                04C0          $VECEND
                04E4
                04E4  LT$NULL:                                        ; Null port routine
          05    04E4          RSB
```

```
                04E5
                04E5  ;
                04E5  ; Stop reasons
                04E5  ;
                04E5
                04E5  .MACRO  $REASON RSNCD, RSNBIT, RSNSTR, ?L
                04E5          .ALIGN WORD
                04E5          .SIGNED_WORD L-.
                04E5          .BYTE RSNCD, RSNBIT
                04E5  L:      .ASCIC  /RSNSTR/
                04E5  .ENDM
                04E5
00000001        04E5  LAT_C_CRSN_DONE         = 1          ; No slots in VC
00000002        04E5  LAT_C_CRSN_IVMSG        = 2          ; Invalid message
00000003        04E5  LAT_C_CRSN_IVSCI        = 3          ; Invalid slave conn id
00000004        04E5  LAT_C_CRSN_IVMCI        = 4          ; invalid master conn id
00000005        04E5  LAT_C_CRSN_DISC         = 5          ; Disconnect requested
00000006        04E5  LAT_C_CRSN_PROGRESS     = 6          ; No progress being made
00000007        04E5  LAT_C_CRSN_TMO          = 7          ; Timeout of circuit
00000008        04E5  LAT_C_CRSN_RETRANS      = 8          ; Retransmit count exceeded
00000009        04E5  LAT_C_CRSN_RESOURCE     = 9          ; No resources
0000000A        04E5  LAT_C_CRSN_TIMRANG      = 10         ; Timer value out of range
                04E5
                04E5
00000000        04E5  LAT_C_RSN_xxx = 0                    ; junk
                04E5
                04E5  LT_RSN_START:
                04E5          $REASON -
                04E5                  GHB$V_START,-
                04E5                  LAT_C_RSN_XXX,-
                04E5          <other than start msg with a zero src index>
                0515  LT_RSN_CSBZERO:
                0515          $REASON -
                0515                  GHB$V_CSBZERO,-
                0515                  LAT_C_RSN_XXX,-
                0515          <circuit index zero>
                052D  LT_RSN_CSBRANGE:
                052D          $REASON -
                052D                  GHB$V_CSBRANGE,-
                052D                  LAT_C_RSN_XXX,-
                052D          <circuit index out of range>
                054D  LT_RSN_CSBINVALID:
                054D          $REASON -
                054D                  GHB$V_CSBINVALID,-
                054D                  LAT_C_RSN_XXX,-
                054D          <circuit index invalid>
                0568  LT_RSN_CSBSTALE:
                0568          $REASON -
                0568                  GHB$V_CSBSTALE,-
                0568                  LAT_C_RSN_XXX,-
                0568          <circuit stale reference>
                0584  LT_RSN_HALT:
                0584          $REASON -
                0584                  GHB$V_HALT,-
                0584                  LAT_C_RSN_XXX,-
                0584          <circuit forced to halt>
                059F  LT_RSN_INVALIDLOCID:
```

```
059F  1114          $REASON -
059F  1115                  GHB$V_INVALIDLOCID,-
059F  1116                  LAT_C_RSN_XXX,-
059F  1117          <invalid local slot id>
05BA  1118 LT_RSN_INVALIDREMID:
05BA  1119          $REASON -
05BA  1120                  GHB$V_INVALIDREMID,-
05BA  1121                  LAT_C_RSN_XXX,-
05BA  1122          <invalid remote slot id>
05D5  1123 LT_RSN_BADCREDITS:
05D5  1124          $REASON -
05D5  1125                  GHB$V_BADCREDITS,-
05D5  1126                  LAT_C_RSN_XXX,-
05D5  1127          <bad number of credits in slot>
05F8  1128 LT_RSN_REPCREATE:
05F8  1129          $REASON -
05F8  1130                  GHB$V_REPCREATE,-
05F8  1131                  LAT_C_RSN_XXX,-
05F8  1132          <repeat create of slot by server>
061C  1133 LT_RSN_REPDISC:
061C  1134          $REASON -
061C  1135                  GHB$V_REPDISC,-
061C  1136                  LAT_C_RSN_XXX,-
061C  1137          <repeat disconnect of slot by server>
0644  1138
0644  1139 .ALIGN  LONG
0644  1140
```

```
                        0644  1142          .SBTTL  LT$CTRL_INIT  - Controller Initialization
                        0644  1143
                        0644  1144          .DEFAULT DISPLACEMENT, WORD
                        0644  1145 ;++
                        0644  1146 ; LT$CTRL_INIT - Initialize port driver
                        0644  1147 ;
                        0644  1148 ; Functional description:
                        0644  1149 ;
                        0644  1150 ; This routine is entered at driver load and power recovery.
                        0644  1151 ;
                        0644  1152 ; Inputs:
                        0644  1153 ;
                        0644  1154 ;        R6 = DDB address
                        0644  1155 ;        R8 = CRB address
                        0644  1156 ;
                        0644  1157 ; Outputs:
                        0644  1158 ;
                        0644  1159 ;        R0 through r3 are modified.
                        0644  1160 ;
                        0644  1161 ; Implicit inputs:
                        0644  1162 ;
                        0644  1163 ;        IPL = IPL$_POWER
                        0644  1164 ;
                        0644  1165 ;--
                        0644  1166 LT$CTRL_INIT:                          ; Initialize controller
                        0644  1167 ;
                        0644  1168 ; Save the address of unit zero for cloning ucbs, if its here now.
                        0644  1169 ;
     51    04 A6   D0   0644  1170          MOVL    DDB$L_UCB(R6), R1        ; Its in the device data block
           05      13   0648  1171          BEQL    5$                       ; Do not store a zero
  FB1D CF    51   D0   064A  1172          MOVL    R1, GHB_L_UCB0           ; Save its address
                       064F  1173 5$:
 0000006C'GF   D0      064F  1174          MOVL    G^SCH$GL_NULLPCB+PCB$L_PHD,- ; Get address of NULL PHD
      FB20 CF          0655  1175                   GHB_L_NULLCPU
  FB1B CF    38   C0   0658  1176          ADDL    #PHD$L_CPUTIM,GHB_L_NULLCPU ; Get offset to "idle time"
                       065D  1177
                       065D  1178 ;
                       065D  1179 ; Attempt to relocate terminal class vector table in case we are the
                       065D  1180 ; first port driver to load.
                       065D  1181 ;
                       065D  1182
51   00000000'GF   D0  065D  1183          MOVL    G^TTY$GL_DPT,R1          ; Get terminal class driver DPT address
     50    1E A1   3C  0664  1184          MOVZWL  DPT$W_VECTOR(R1),R0      ; Get offset to class vector
  50    51    50   C1  0668  1185          ADDL3   R0,R1,R0                 ; Calculate virtual address
              60   D5  066C  1186 10$:     TSTL    (R0)                     ; Already relocated or done?
              05   15  066E  1187          BLEQ    20$                      ; Br if yes
        80    51   C0  0670  1188          ADDL    R1,(R0)+                 ; Else, add bias
              F7   11  0673  1189          BRB     10$                      ; Loop till done
                       0675  1190 ;
                       0675  1191 ; Relocate port vector table
                       0675  1192 ;
                       0675  1193 20$:
  51   0000'CF   DE    0675  1194          MOVAL   LT$DPT,R1                ; Get our DPT address
  50   FE2A CF   DE    067A  1195          MOVAL   LT$VECTOR,R0             ; Get address of our dispatch vector
                       067F  1196 30$:
              60   D5  067F  1197          TSTL    (R0)                     ; Already relocated or done?
              05   15  0681  1198          BLEQ    40$                      ; Br if yes, powerfail
```

```
                80   51   C0  0683  1199          ADDL    R1,(R0)+                  ; Else, add bias
                     F7   11  0686  1200          BRB     30$                       ; Loop til done
                                0688  1201 40$:
                072D'CF   9E  0688  1202          MOVAB   LT$STRENTRY,-             ; Set address of start entry point
                FA8D CF       068C  1203                  GHB$L_STRENTRY+GHB_AREA ;
                0805'CF   9E  068F  1204          MOVAB   LT$SHUTENTRY,-            ; Set address shutdown entry point
                FA82 CF       0693  1205                  GHB$L_SHUTENTRY+GHB_AREA;
                0935'CF   9E  0696  1206          MOVAB   LT$SETENTRY,-             ; Set address of multicast set entry
                FA73 CF       069A  1207                  GHB$L_SETENTRY+GHB_AREA ;
                FA6B CF   9E  069D  1208          MOVAB   GHB_AREA,-                ; Save address of the common area
                FE00 CF       06A1  1209                  GHB_L_AREA                ;   so we can find them from LATCP
            00000000'GF   90  06A4  1210          MOVB    G^EXE$GQ_SYSTIME,-        ; Get a random number
                FAD8 CF       06AA  1211                  GHB_B_INCARN              ;  and save for the message seed value
                              06AD  1212
                FAD9 CF   9E  06AD  1213          MOVAB   GHB_T_MC_DATA,-           ; Address of multicast data area
                FA74 CF       06B1  1214                  GHB$L_NODE+GHB_AREA
                              06B4  1215
                              06B4  1216          ASSUME  GHB$B_LATVERSION  EQ  GHB$W_VMSVERSION+2
                              06B4  1217          ASSUME  GHB$B_LATECO      EQ  GHB$B_LATVERSION+1
     FA4F CF    00050004 8F   D0  06B4  1218          MOVL    #<<LAT$C_CUR_ECO>@8! -    ; Store current ECO
                              06BD  1219                   LAT$C_COR_VER>@16! -     ;            and VERsion
                              06BD  1220                   LAT$C_VMS_VER, -         ;        and VMS VERsion
                              06BD  1221                  GHB$W_VMSVERSION+GHB_AREA
                         0A   90  06BD  1222          MOVB    #LAT$C_KEEP_CSB,-         ; Set number of CSBs to keep around
                FAC4 CF       06BF  1223                  GHB_B_OLD_CSBCNT
                F972 CF   9E  06C2  1224          MOVAB   GHB_L_CSBTABLE,-          ; Store address of CSB table
                FA63 CF       06C6  1225                  GHB$L_CSBLST+GHB_AREA
             50 FA63 CF   9E  06C9  1226          MOVAB   GHB$Q_OLD_CSBS+GHB_AREA,R0 ; Get address of old CSB entries
                         60   D5  06CE  1227          TSTL    (R0)                      ; Already initialized?
                         07   12  06D0  1228          BNEQ    50$                       ; Br if yes, skip next section
                60       50   D0  06D2  1229          MOVL    R0,(R0)                   ; Initialize listhead
              04 A0      50   D0  06D5  1230          MOVL    R0,4(R0)                  ;        ...
                              06D9  1231
                              06D9  1232 50$:     ;
                              06D9  1233          ;  Setup the initial rating value, based on memory size
                              06D9  1234          ;
          50    00000000'GF   D0  06D9  1235          MOVL    G^MMG$GL_MAXPFN,R0        ; Get the max PFN count
          50    00002000 8F   C6  06E0  1236          DIVL    #8192,R0                  ; Weight by 4MB chunks
             50    08      50   C3  06E7  1237          SUBL3   R0,#8,R0                  ; Compute for up to 32 MB systems
                         02   18  06EB  1238          BGEQ    70$                       ; Br if result okay
                         50   D4  06ED  1239          CLRL    R0                        ; Else, more than 32 MB - impressive!
     FA91 CF    FF 8F    50   83  06EF  1240 70$:     SUBB3   R0,#255,GHB_B_RATE        ; Store initial rating
                              05  06F6  1241          RSB
                              06F7  1242
                              06F7  1243          .IF DEFINED LT_HISTORY
```

```
                    06F7  1245          .SBTTL  LT$HISTORY  - Save history in the buffer
                    06F7  1246 ;++
                    06F7  1247 ; LT$HISTORY - Save history in the buffer
                    06F7  1248 ;
                    06F7  1249 ;   Save history on transmits and receives.
                    06F7  1250 ;   The history buffer is defined as follows:
                    06F7  1251 ;
                    06F7  1252 ;                 +-------------------------+
                    06F7  1253 ;                 !  Pointer to next slot   !
                    06F7  1254 ;                 !-------------------------!
                    06F7  1255 ;                 ! Pointer to end of buffer!
                    06F7  1256 ;                 !-------------------------!
                    06F7  1257 ;                 !spare!type! buffer size  !
                    06F7  1258 ;                 !-------------------------!
                    06F7  1259 ;                 !      ... data ...       !
                    06F7  1260 ;
                    06F7  1261 ;                 !                         !
                    06F7  1262 ;                 +-------------------------+
                    06F7  1263 ;
                    06F7  1264 ; Inputs:
                    06F7  1265 ;        R0 -> data to save
                    06F7  1266 ;
                    06F7  1267 ; Outputs:
                    06F7  1268 ;        R1 is clobbered
                    06F7  1269 ;--
                    06F7  1270
        00000020    06F7  1271          SLOT_SIZE = 32                  ; Size of each history slot
                    06F7  1272
                    06F7  1273 LT$HISTORY:
    51    FA19 CF   D0  06F7  1274          MOVL    GHB$L_HISTORY+GHB_AREA,R1 ; Point to history buffer
               2E   13  06FC  1275          BEQL    90$                     ; Br if none
                    06FE  1276                                           ;
               50   DD  06FE  1277          PUSHL   R0                      ; Save R0
               20   C1  0700  1278          ADDL3   #SLOT_SIZE,-            ; Compute end address for this slot
          50   61       0702  1279                   HBF_L_NEXT(R1),R0
       04 A1   50   D1  0704  1280          CMPL    R0,HBF_L_BUFEND(R1)     ; Are we past the end?
               04   1B  0708  1281          BLEQU   30$                     ; Br if no, continue
          0C A1   9E    070A  1282          MOVAB   HBF_Z_DATA(R1),-        ; Else, back to beginning
               61       070D  1283                   HBF_L_NEXT(R1)
          51   61   D0  070E  1284 30$:     MOVL    HBF_L_NEXT(R1),R1       ; Get address of next available slot
81  00000000'GF   7D  0711  1285          MOVQ    G^EXE$GQ_SYSTIME,(R1)+  ; Store time in first quadword
          50   6E   D0  0718  1286          MOVL    (SP),R0                 ; Get back buffer address
          81   80   7D  071B  1287          MOVQ    (R0)+, (R1)+            ; Save the data
          81   80   7D  071E  1288          MOVQ    (R0)+, (R1)+            ; Save the data
          81   80   7D  0721  1289          MOVQ    (R0)+, (R1)+            ; Save the data
                    0724  1290          ASSUME  HBF_L_NEXT EQ 0
    F9EB DF   51   D0  0724  1291          MOVL    R1,@GHB$L_HISTORY+GHB_AREA ; Save address of next slot
               50 8ED0 0729  1292          POPL    R0                      ; Restore R0
                    072C  1293                                           ;
               05   072C  1294 90$:     RSB
                    072D  1295          .ENDC   ;; DEFINED LT_HISTORY
```

```
                        072D          .SBTTL  LT$STRENTRY  - Entry to start driver
                        072D  ;++
                        072D  ; LT$STRENTRY - Start LTDRIVER entry point
                        072D  ;
                        072D  ; Functional description:
                        072D  ;
                        072D  ;       This entry is called by the startup program to initialize the
                        072D  ;       driver.  Here we save the datalink UCB address, allocate the CSB's
                        072D  ;       and start the first receive.
                        072D  ;
                        072D  ; Inputs:
                        072D  ;       R5 = UCB address of datalink
                        072D  ;
                        072D  ; Implicit inputs:
                        072D  ;
                        072D  ;       Called in user context.
                        072D  ;
                        072D  ; Outputs:
                        072D  ;       R0 = success or failure
                        072D  ;--
                        072D
                        072D
                        072D  LT$STRENTRY:
                        072D          DSBINT  #LAT$C_IPL              ; Raise IPL to prevent interruption
F9E8 CF    55    D0     0733          MOVL    R5,GHB$L_UCB+GHB_AREA   ; Save the datalink UCB address
                        0738
59   F8FC CF    9E     0738          MOVAB   GHB_L_CSBTABLE,R9       ; Get CSB table base address
     5A   20    9A     073D          MOVZBL  #LAT$C_MAX_CSBS,R10     ; Get table length in longwords
          89    D4     0740  20$:    CLRL    (R9)+                   ; Zero the whole table
       FB 5A    F5     0742          SOBGTR  R10,20$                 ; Do whole table
                        0745          ;
                        0745          ;  Save the last CPU idle value
                        0745          ;
          FA2F DF    D0  0745          MOVL    @GHB_L_NULLCPU,-        ; Save last idle value
          FA30 CF        0749                   GHB_L_LASTCPU
50   00000000'GF    D0  074C          MOVL    G^EXE$GL_MP,R0          ; Get ptr to MP code
               07    13  0753          BEQL    40$                     ; Br if not there
          0000'C0    D0  0755          MOVL    MPS$GL_NULLCPU(R0),-    ; Else, get Secondary null time
          FA24 CF        0759                   GHB_L_NULLSEC
                        075C  40$:    ;
                        075C          ;  See if the datalink supports the fast interface and if so
                        075C          ;  then allocate an FFI block and initialize that interface.
                        075C          ;
               50    D4  075C          CLRL    R0                      ; Assume error
       51   24 A5    D0  075E          MOVL    UCB$L_CRB(R5), R1       ; Get address of datalink's CRB
       51   40 A1    D0  0762          MOVL    CRB$L_INTD+VEC$L_START(R1), R1 ; Get address of FFI INIT routine
               36    13  0766          BEQL    70$                     ; Br if not present
             0045    30  0768          BSBW    LT$GETFFI               ; Else, create and initialize FFI
            30 50    E9  076B          BLBC    R0,70$                  ; Br if error
FA01 CF     54    D0     076E          MOVL    R4,GHB_L_FFI            ; Else, save FFI block address
                        0773
                        0773          CLRBIT  #GHB_STS_V_MULTI,-      ; Clear the multicast busy flag
                        0773                   GHB_B_STATUS
          FC5F CF    D4  0779          CLRL    GHB_L_MULTIBFR          ; No multicast buffer is possible
             01B5    30  077D          BSBW    LT$SETENTRY             ; Setup multicast message
                        0780
50   F9E8 CF    D0     0780          MOVL    GHB_L_UCBO, R0          ; Set unit seed in the ucb
```

```
                17   13  0785  1354          BEQL    70$                         ; No ucb0, bad news
                60   B4  0787  1355          CLRW    UCB$W_MB_SEED(R0)           ; Start with unit 1
              1053   30  0789  1356          BSBW    LT$SET_TIMER                ; Start timer, so we a'ways have a tick
                         078C  1357
        55   F990 CF   D0  078C  1358          MOVL    GHB$L_UCB+GHB_AREA, R5      ; Get datalink UCB address
             5C A5     B6  0791  1359          INCW    UCB$W_REFC(R5)              ; One more ref to keep it around for us
    68 A5    8000 8F   A8  0794  1360          BISW    #^x8000,UCB$W_DEVSTS(R5)    ; Allow unit to be re-started
                         079A  1361                                              ;   automatically by datalink
        50     00'8F   9A  079A  1362          MOVZBL  #SS$_NORMAL, R0             ; Return success
             0B 50     E8  079E  1363 70$:     BLBS    R0, 80$                     ; Br if okay
            F97B CF    D4  07A1  1364          CLRL    GHB$L_UCB+GHB_AREA          ; Else, forget about datalink UCB
                5E     10  07A5  1365          BSBB    LT$SHOTENTRY                ; Clean it all up
        50   0000'8F   3C  07A7  1366          MOVZWL  #SS$_INSFMEM, R0            ; Return not enough memory
                         07AC  1367 80$:     ENBINT                              ; Restore IPL
                     05  07AF  1368          RSB
                         07B0  1369
```

```
                    07B0  1371          .SBTTL  LT$GETFFI - Get the FFI block and initialize FFI interface
                    07B0  1372 ;++
                    07B0  1373 ; LT$GETFFI - Get FFI block
                    07B0  1374 ;
                    07B0  1375 ; Functional description:
                    07B0  1376 ;
                    07B0  1377 ;        Initialize the FFI interface.
                    07B0  1378 ;
                    07B0  1379 ; Inputs:
                    07B0  1380 ;
                    07B0  1381 ;        R1 = Address of FFI INIT routine in datalink driver
                    07B0  1382 ;        R5 = UCB address of datalink
                    07B0  1383 ;
                    07B0  1384 ; Outputs:
                    07B0  1385 ;
                    07B0  1386 ;        R0 = Status
                    07B0  1387 ;        R1-R4 are destroyed.
                    07B0  1388 ;
                    07B0  1389 ;--
                    07B0  1390
                    07B0  1391 LT$GETFFI:
          51   DD   07B0  1392          PUSHL   R1                      ; Save init routine address
     51   3E   9A   07B2  1393          MOVZBL  #FFI$C_LENGTH,R1        ; Set length of buffer to allocate
00000000'GF   16   07B5  1394          JSB     G^EXE$ALONONPAGED       ; Try to allocate an FFI block
       31 50   E9   07BB  1395          BLBC    R0,30$                  ; Br if error
     54   52   D0   07BE  1396          MOV     R2,R4                   ; Copy FFI block address
                    07C1  1397          ASSUME  FFI$L_FL EQ 0
                    07C1  1398          ASSUME  FFI$L_BL EQ FFI$L_FL+4
                    07C1  1399          ASSUME  FFI$W_SIZE EQ FFI$L_BL+4
                    07C1  1400          ASSUME  FFI$B_TYPE EQ FFI$W_SIZE+2
                    07C1  1401          ASSUME  FFI$B_SPARE EQ FFI$B_TYPE+1
          82   7C   07C1  1402          CLRQ    (R2)+                   ; Clear link pointers
82   0013003E 8F   D0   07C3  1403          MOVL    #<DYN$C_BUFIO@16>!FFI$C_LENGTH,(R2)+ ; Store block size & type
                    07CA  1404          ASSUME  FFI$L_CTX_DL EQ FFI$B_SPARE+1
                    07CA  1405          ASSUME  FFI$L_XMIT EQ FFI$L_CTX_DL+4
                    07CA  1406          ASSUME  FFI$L_XMIT_DONE EQ FFI$L_XMIT+4
                    07CA  1407          ASSUME  FFI$L_RECV_DONE EQ FFI$L_XMIT_DONE+4
                    07CA  1408          ASSUME  FFI$L_ERROR EQ FFI$L_RECV_DONE+4
                    07CA  1409          ASSUME  FFI$L_SHUT_DONE EQ FFI$L_ERROR+4
          82   7C   07CA  1410          CLRQ    (R2)+                   ; Zero CTX & XMIT routine address
82   1B1B'CF   9E   07CC  1411          MOVAB   W^LT$XMT_FFIDONE,(R2)+  ; Store XMIT Complete routine address
82   0CEC'CF   9E   07D1  1412          MOVAB   W^LT$FFI_RCV_MSG,(R2)+  ; Store RECV Complete routine address
82   0934'CF   9E   07D6  1413          MOVAB   W^LT$ASYNCERR,(R2)+     ; Store address of ERROR handler
82   0923'CF   9E   07DB  1414          MOVAB   W^LT$SHUT_DONE,(R2)+    ; Store address of SHUTDOWN handler
                    07E0  1415          ASSUME  FFI$L_SPARE0 EQ FFI$L_SHUT_DONE+4
                    07E0  1416          ASSUME  FFI$L_SPARE1 EQ FFI$L_SPARE0+4
                    07E0  1417          ASSUME  FFI$L_SPARE2 EQ FFI$L_SPARE1+4
                    07E0  1418          ASSUME  FFI$L_SPARE3 EQ FFI$L_SPARE2+4
          82   7C   07E0  1419          CLRQ    (R2)+                   ; Not used
          82   7C   07E2  1420          CLRQ    (R2)+
                    07E4  1421          ASSUME  FFI$L_DL_UCB EQ FFI$L_SPARE3+4
                    07E4  1422          ASSUME  FFI$L_PID EQ FFI$L_DL_UCB+4
                    07E4  1423          ASSUME  FFI$W_CHAN EQ FFI$L_PID+4
     82   55   D0   07E4  1424          MOVL    R5,(R2)+                ; Store datalink UCB address
          82   D4   07E7  1425          CLRL    (R2)+                   ; Zero PID
          82   B4   07E9  1426          CLRW    (R2)+                   ; Zero CHAN
  50   00'8F   9A   07EB  1427          MOVZBL  #SS$_NORMAL,R0          ; Return success
```

```
      51 8ED0  07EF  1428 30$:    POPL    R1                      ; Restore R1
   0F 50   E9  07F2  1429         BLBC    R0,90$                  ; Br if allocation error
      53   D4  07F5  1430         CLRL    R3                      ; No parameters
      61   16  07F7  1431         JSB     (R1)                    ; Else, call datalink init routine
   08 50   E8  07F9  1432         BLBS    R0,90$                  ; Br if success
50    54   D0  07FC  1433         MOVL    R4,R0                   ; Else, copy FFI block address
    139E   30  07FF  1434         BSBW    LT$DEALPOOL             ; Deallocate the FFI block
      50   D4  0802  1435         CLRL    R0                      ; Return failure
           05  0804  1436 90$:    RSB
```

```
                        0805  1438          .SBTTL  LT$SHUTENTRY  - Shutdown entry point
                        0805  1439 ;++
                        0805  1440 ; LT$SHUTENTRY - Shutdown LTDRIVER entry point
                        0805  1441 ;
                        0805  1442 ; Functional description:
                        0805  1443 ;
                        0805  1444 ;       Shutdown things when we are done.
                        0805  1445 ;
                        0805  1446 ; Inputs:
                        0805  1447 ;       none
                        0805  1448 ;
                        0805  1449 ; Implicit inputs:
                        0805  1450 ;
                        0805  1451 ;       Called in user context.
                        0805  1452 ;
                        0805  1453 ; Outputs:
                        0805  1454 ;       none
                        0805  1455 ;--
                        0805  1456
                        0805  1457 LT$SHUTENTRY:
                        0805  1458          DSBINT  #LAT$C_IPL              ; Sync with rest of things
                        080B  1459          SETBIT  #GHB_STS_V_SHUT,-       ; Indicate that shutdown has been
                        080B  1460                   GHB_B_STATUS           ;    requested
                        0811  1461
                        0811  1462 ;
                        0811  1463 ;       Deallocate and shut down all the circuits
                        0811  1464 ;
59      F823 CF    DE   0811  1465          MOVAL   GHB_L_CSBTABLE, R9      ; Table of CSB addresses
        5A   20    9A   0816  1466          MOVZBL  #LAT$C_MAX_CSBS, R10    ; and its length
        58   89    D0   0819  1467 40$:     MOVL    (R9)+, R8               ; Get a CSB address
             03    13   081C  1468          BEQL    60$                     ; No CSB address here
           0ED0    30   081E  1469          BSBW    LT$CIRCDEAD             ; Kill all processes on this one
          F5 5A    F5   0821  1470 60$:     SOBGTR  R10, 40$                ; Til end of table
                        0824  1471
                        0824  1472 ;
                        0824  1473 ; We can reduce our interest in the datalink UCB here early because
                        0824  1474 ; the LAT control program still has a channel to the UCB and is still
                        0824  1475 ; active.   The UCB will disappear when the LATCP deassigns its last
                        0824  1476 ; channel to it.
                        0824  1477 ;
                        0824  1478
59      F8F8 CF    D0   0824  1479          MOVL    GHB$L_UCB+GHB_AREA, R9  ; Get datalink UCB and reduce our interest
             19    13   0829  1480          BEQL    80$                     ; No ucb here
68 A9   8000 8F    AA   082B  1481          BICW    #^x8000,UCB$W_DEVSTS(R9); Don't want datalink to re-start
                        0831  1482                                          ;    any more!
        5C A9      B7   0831  1483          DECW    UCB$W_REFC(R9)          ; Let datalink UCB go away.
        F8E8 CF    D4   0834  1484          CLRL    GHB$L_UCB+GHB_AREA      ; Forget we ever had it
                        0838  1485                                          ; This prevents more circuits from
                        0838  1486                                          ; being accepted.
                        0838  1487          ;
                        0838  1488          ;  Wait for any datalink re-start timer to expire. Spin on
                        0838  1489          ;  the INTERLOCK bit.
                        0838  1490          ;
                        0838  1491          SETIPL  #IPL$_ASTDEL            ; Prevent us from being deleted,
                        083B  1492                                          ;  but allow scheduling activity
04 68 A9        0E   E1 083B  1493 70$:     BBC     #14,UCB$W_DEVSTS(R9),80$ ; Proceed if UCB can evaporate
             70    10   0840  1494          BSBB    BUILD_TQE               ; Wait around for a while
```

```
                F7   11  0842  1495          BRB     70$                     ; Try again
                         0844  1496
                         0844  1497 80$:
                         0844  1498 ;
                         0844  1499 ;        Now we must wait for all the CSBs to disappear.  We must not leave
                         0844  1500 ;        here until all the processes have run down and the CSBs are gone
                         0844  1501 ;        since latstart can be used again very quickly and that could cause
                         0844  1502 ;        CSBs to be allocated again.  If processes don't run down we will
                         0844  1503 ;        be stuck here until they do.
                         0844  1504 ;
                         0844  1505          SETIPL  #IPL$_ASTDEL            ; Prevent us from being deleted,
                         0847  1506                                          ;  but allow scheduling activity
                02   11  0847  1507          BRB     110$                    ; See if CSB's are all gone
                         0849  1508          ;
                         0849  1509          ;  Now delay while we wait for CSB's to run down
                         0849  1510          ;
                67   10  0849  1511 100$:    BSBB    BUILD_TQE               ; Build TQE to wait on and wait
                         084B  1512
59    F7E9 CF    DE  084B  1513 110$:    MOVAL   GHB_L_CSBTABLE,R9       ; Table of CSB addresses
      5A   20    9A  0850  1514          MOVZBL  #LAT$C_MAX_CSBS, R10    ;   and its length
      58   89    D0  0853  1515 120$:    MOVL    (R9)+, R8               ; Get a CSB address
           F1    12  0856  1516          BNEQ    100$                    ; Just spin here looking
        F8 5A    F5  0858  1517          SOBGTR  R10, 120$               ; Til end of table
                         085B  1518          ;
                         085B  1519          ;  Now wait for all the UCBs to dissappear except unit 0
                         085B  1520          ;
                02   11  085B  1521          BRB     160$                    ; Check if we're all done
                         085D  1522
                53   10  085D  1523 140$:    BSBB    BUILD_TQE               ; Build TQE to wait on and wait
                         085F  1524          ;
                         085F  1525          ;  Now delay while we wait for terminal UCB's to run down
                         085F  1526          ;
50    F909 CF    D0  085F  1527 160$:    MOVL    GHB_L_UCB0,R0           ; Get the LT UCB address
           0A    13  0864  1528          BEQL    180$                    ; ???
  51    24 A0    D0  0866  1529          MOVL    UCB$L_CRB(R0),R1        ; Get the CRB address
  01    0C A1    B1  086A  1530          CMPW    CRB$W_REFC(R1),#1       ; All done ?
           ED    12  086E  1531          BNEQ    140$                    ; Br if no
                         0870  1532 180$:    ;
                         0870  1533          ;  Reset completion routines for datalink's FFI interface
                         0870  1534          ;
                         0870  1535          SETIPL  #LAT$C_IPL              ; Sync access to rest of driver
55    F8FD CF    D0  0873  1536          MOVL    GHB_L_FFI, R5           ; Get FFI block address
           0C    13  0878  1537          BEQL    200$                    ; Br if none
      1B9F'CF    9E  087A  1538          MOVAB   W^LT$DROPCXB,-          ; Drop all RECEIVE buffers
        18 A5         087E  1539                  FFI$L_RECV_DONE(R5)
      1B96'CF    9E  0880  1540          MOVAB   W^LT$FFIPOSTDONE,-      ; Deallocate all TRANSMIT CXBs
        14 A5         0884  1541                  FFI$L_XMIT_DONE(R5)
                         0886  1542 200$:    ;
                         0886  1543          ;  Deallocate the multicast buffer
                         0886  1544          ;
50    FB52 CF    D0  0886  1545          MOVL    GHB_L_MULTIBFR,R0       ; Get the multicast buffer
           08    13  088B  1546          BEQL    220$                    ; Br if none
                         088D  1547          ASSUME  GHB_STS_V_MULTI EQ 0
  03 F8F3 CF    E8  088D  1548          BLBS    GHB_B_STATUS,220$       ; Br if multicast buffer is active
        130B    30  0892  1549          BSBW    LT$DEALPOOL             ; Else, deallocate the multicast buffer
      FB43 CF    D4  0895  1550 220$:    CLRL    GHB_L_MULTIBFR          ; Forget we had a multicast buffer
                         0899  1551          ;
```

```
                    0899  1552          ;  Flush all CSBs on the old CSB display queue
                    0899  1553          ;
50   F893 DF   0F   0899  1554 240$:    REMQUE  @GHB$Q_OLD_CSBS+GHB_AREA,R0 ; Get CSB address
          05   1D   089E  1555          BVS     300$                        ; Br if none
        12FD   30   08A0  1556          BSBW    LT$DEALPOOL                 ; Else, deallocate the CSB
          F4   11   08A3  1557          BRB     240$                        ; Loop til done
          0A   90   08A5  1558 300$:    MOVB    #LAT$C_KEEP_CSB,-           ; Reset number of CSBs to keep around
     F8DC CF        08A7  1559                  GHB_B_OLD_CSBCNT
                    08AA  1560          ;
                    08AA  1561          ;  Mark LTDRIVER as inactive
                    08AA  1562          ;
     F8D6 CF   94   08AA  1563          CLRB    GHB_B_STATUS                ; Clear ACTIVE flag to stop timer
                    08AE  1564                                              ;   clear SHUTdown flag
                    08AE  1565                                              ;   clear MULTIcast buffer flag
                    08AE  1566          ENBINT
              05    08B1  1567          RSB
                    08B2  1568
```

```
                        08B2  1570          .SBTTL  TQE TIMER BUILDING AND WAITING ROUTINES
                        08B2  1571 ;++
                        08B2  1572 ; BUILD_TQE - Build a TQE
                        08B2  1573 ;
                        08B2  1574 ; Inputs:
                        08B2  1575 ;        IPL = ASTDEL
                        08B2  1576 ;
                        08B2  1577 ; Implicit inputs:
                        08B2  1578 ;
                        08B2  1579 ;        Called in user context
                        08B2  1580 ;
                        08B2  1581 ; Outputs:
                        08B2  1582 ;        R0-R5 are destroyed
                        08B2  1583 ;--
                        08B2  1584
                        08B2  1585 BUILD_TQE:
     00000000'GF    16  08B2  1586          JSB     G^EXE$ALLOCTQE          ; Allocate a TQE (returns IPL = ASTDEL)
           4A 50    E9  08B8  1587          BLBC    R0,90$                  ; Br if allocation failure
        0B A2    01    90  08BB  1588          MOVB    #TQE$C_SSSNGL,TQE$B_RQTYPE(R2) ; Indicate single shot request
     0C A2    06'AF    9E  08BF  1589          MOVAB   B^SHUT_TIMER,TQE$L_FPC(R2) ; Set wake up routine address
              7E    DC  08C4  1590          MOVPSL  -(SP)                   ; Push current PSL to resume process
                        08C6  1591          ASSUME  IPL$_SYNCH EQ IPL$_TIMER
                        08C6  1592          SETIPL  #IPL$_SYNCH             ; Sync access to system data base
50   00000000'GF    7D  08C9  1593          MOVQ    G^EXE$GQ_SYSTIME,R0     ; Get current time
50   004C4B40 8F    C0  08D0  1594          ADDL    #LAT$C_SHUTDELAY,R0     ; Calculate expiration time
           51    00    D8  08D7  1595          ADWC    #0,R1                   ; Add carry to top half
           55    52    D0  08DA  1596          MOVL    R2,R5                   ; Copy TQE address
     00000000'GF    16  08DD  1597          JSB     G^EXE$INSTIMQ           ; Insert TQE on timer queue
                        08E3  1598          CLRBIT  #GHB_STS_V_TQE,-        ; Indicate TQE not complete - yet
                        08E3  1599                   GHB_B_STATUS
              08    11  08E9  1600          BRB     60$                     ; Wait for TQE
              03    E0  08EB  1601 40$:     BBS     #GHB_STS_V_TQE,-        ; Br if TQE is complete
     14 F894 CF         08ED  1602                   GHB_B_STATUS,90$
              7E    DC  08F1  1603          MOVPSL  -(SP)                   ; Push current PSL to resume process
54   00000000'GF    D0  08F3  1604 60$:     MOVL    G^SCH$GL_CURPCB,R4      ; Get current PCB address
           50    01    9A  08FA  1605          MOVZBL  #RSN$_ASTWAIT,R0        ; Set waiting resource
     00000000'GF    16  08FD  1606          JSB     G^SCH$RWAIT             ; Make process wait for resource
              E6    11  0903  1607          BRB     40$                     ; Check if TQE expired
                    05  0905  1608 90$:     RSB                             ; Return to caller
                        0906  1609
                        0906  1610 ;++
                        0906  1611 ; SHUT_TIMER
                        0906  1612 ;
                        0906  1613 ;        This routine wakes up the waiting process
                        0906  1614 ;
                        0906  1615 ; Inputs:
                        0906  1616 ;        R4 = PCB address
                        0906  1617 ;        R5 = TQE address
                        0906  1618 ;
                        0906  1619 ;        IPL = IPL$_TIMER
                        0906  1620 ;
                        0906  1621 ; Outputs:
                        0906  1622 ;        R0 is destroyed
                        0906  1623 ;        R5 = TQE address of system non-repeating TQE
                        0906  1624 ;--
                        0906  1625
                        0906  1626 SHUT_TIMER:
```

```
                    0906  1627        SETBIT  #GHB_STS_V_TQE,-        ; Indicate TQE has completed
                    0906  1628                 GHB_B_STATUS
        50    01    9A    090C  1629  MOVZBL  #RSN$_ASTWAIT,R0        ; Set waiting resource
 00000000'GF        16    090F  1630  JSB     G^SCH$RAVAIL            ; Mark resource available
        50    55    D0    0915  1631  MOVL    R5,R0                   ; Copy TQE address
            1285    30    0918  1632  BSBW    LT$DEALPOOL             ; Deallocate the TQE
55   00000000'GF    DE    091B  1633  MOVAL   G^EXE$AL_TQENOREPT,R5   ; Setup SYSTEM no repeat TQE address
                    05    0922  1634  RSB                             ; Return to caller
```

```
                    0923  1636          .SBTTL  LT$SHUT_DONE - Datalink shutdown complete entry
                    0923  1637 ;++
                    0923  1638 ; LT$SHUT_DONE - Datalink shutdown complete entry point
                    0923  1639 ;
                    0923  1640 ; Functional description:
                    0923  1641 ;
                    0923  1642 ;        Deallocate the FFI block when datalink is all done.
                    0923  1643 ;
                    0923  1644 ; inputs:
                    0923  1645 ;        R4 = FFI block address
                    0923  1646 ;
                    0923  1647 ; outputs:
                    0923  1648 ;        R0,R1,R4 are destroyed
                    0923  1649 ;        All other registers must be preserved
                    0923  1650 ;--
                    0923  1651
                    0923  1652 LT$SHUT_DONE::
          2C   BB   0923  1653          PUSHR   #^M<R2,R3,R5>           ; Save registers
50   F84B CF   D0   0925  1654          MOVL    GHB_L_FFI,R0            ; Get FFI block address
     F846 CF   D4   092A  1655          CLRL    GHB_L_FFI               ; Zero pointer to FFI block
        126F   30   092E  1656          BSBW    LT$DEALPOOL             ; Deallocate the FFI block
          2C   BA   0931  1657          POPR    #^M<R2,R3,R5>           ; Restore registers
               05   0933  1658          RSB
```

```
     0934  1660          .SBTTL  LT$ASYNCERR - Process async errors from datalink
     0934  1661 ;++
     0934  1662 ; LT$ASYNCERR - Process async errors from datalink
     0934  1663 ;
     0934  1664 ; Functional description:
     0934  1665 ;
     0934  1666 ;        Do Nothing.
     0934  1667 ;
     0934  1668 ;--
     0934  1669 LT$ASYNCERR:
 05  0934  1670          RSB
```

```
                        0935  1672          .SBTTL  LT$SETENTRY  - Set a new multicast message
                        0935  1673 ;++
                        0935  1674 ; LT$SETENTRY - Set new multicast message
                        0935  1675 ;
                        0935  1676 ; Functional description:
                        0935  1677 ;
                        0935  1678 ; Take the data from the multicast data area and store it in a new message
                        0935  1679 ; or bump the incarnation so we know to do it next time.
                        0935  1680 ;
                        0935  1681 ; Inputs:
                        0935  1682 ;        None.
                        0935  1683 ;
                        0935  1684 ; Outputs:
                        0935  1685 ;        R0 = success or failure
                        0935  1686 ;        All other registers are preserved.
                        0935  1687 ;
                        0935  1688 ; Implicit outputs:
                        0935  1689 ;        Multicast timer set to 1 second interval.
                        0935  1690 ;
                        0935  1691 ;--
                        0935  1692
                        0935  1693 LT$SETENTRY:
                        0935  1694          DSBINT  #LAT$C_IPL
     007E 8F    BB      093B  1695          PUSHR   #^M<R1,R2,R3,R4,R5,R6>  ; Save registers
          56    D4      093F  1696          CLRL    R6                      ; Assume no old CXB change flags
52   FA97 CF    D0      0941  1697          MOVL    GHB_L_MULTIBFR, R2      ; Get the Multicast CXB
          1A    13      0946  1698          BEQL    10$                     ; Br if none, go make one
     F839 CF    96      0948  1699          INCB    GHB_B_INCARN            ; Bump the incarnation count
                        094C  1700          ASSUME  GHB_STS_V_MULTI EQ 0
  03 F834 CF    E9      094C  1701          BLBC    GHB_B_STATUS, 5$        ; Br if multicast buffer is free
        0099    31      0951  1702          BRW     70$                     ; Else, do this operation later
                        0954  1703          ;
                        0954  1704          ;  Multicast buffer is available, deallocate it!
                        0954  1705          ;
     56   4F A2    90   0954  1706 5$:      MOVB    XMT_B_MC_CHG_FLAG(R2),R6, Save previous change flags
          50   52    D0  0958  1707          MOVL    R2, R0                  ; Deallocate the old CXB buffer
          FA7D CF    D4  095B  1708          CLRL    GHB_L_MULTIBFR          ; NO more mutlicast CXB buffer
             123E    30  095F  1709          BSBW    LT$DEALPOOL             ; Dump the buffer
                        0962  1710
     51   F822 CF    3C  0962  1711 10$:     MOVZWL  GHB_W_MC_SIZE, R1       ; Get size of data set by LATCP
51   00000052 8F      C0  0967  1712          ADDL    #XMT_B_MC_SET, R1       ;    plus message overhead
     00000000'GF      16  096E  1713          JSB     G^EXE$ALONONPAGED       ; Now get the buffer
          7B   50     E9  0974  1714          BLBC    R0, 80$                 ; No pool for buffer
     08 A2     51     B0  0977  1715          MOVW    R1, CXB$W_SIZE(R2)      ; Size of buffer
                        097B  1716          ASSUME  CXB$B_CODE EQ CXB$B_TYPE+1
          011B 8F     B0  097B  1717          MOVW    #<1@8>!DYN$C_CXB,-      ; Set type and indicate multicast
             0A A2        097F  1718                  CXB$B_TYPE(R2)          ;  buffer
 18 A2    0048 8F     B0  0981  1719          MOVW    #XMT_T_DATA,CXB$W_BOFF(R2) ; Set offset to start of data
     FA50 CF     52     D0  0987  1720          MOVL    R2, GHB_L_MULTIBFR      ; Save address of the buffer
                        098C  1721          ;
                        098C  1722          ; Copy fixed portion of multicast message
                        098C  1723          ;
     51   FA50 CF     9E  098C  1724          MOVAB   GHB_T_MC_MSG,R1         ; Point to msg template
          61   0A       28  0991  1725          MOVC3   #XMT_C_MC_LENGTH,(R1),- ; Copy the fixed portion of message
             48 A2        0994  1726                   XMT_T_DATA(R2)
                          0996  1727          ;
                          0996  1728          ; Copy the variable data set by LATCP
```

```
                        0996  1729          ;
     51   F7F0 CF   9E  0996  1730          MOVAB   GHB_T_MC_DATA,R1          ; Get address of variable data
 63  61   F7E9 CF   28  099B  1731          MOVC3   GHB_W_MC_SIZE,(R1),(R3)   ; Copy the variable data
                        09A1  1732          ;
                        09A1  1733          ; Set change flags, and incarnation number
                        09A1  1734          ;
     52   FA37 CF   D0  09A1  1735          MOVL    GHB_L_MULTIBFR,R2         ; Get back the multicast buffer
4F A2  56   FF 8F   8D  09A6  1736          XORB3   #XMT_CHFLG_M_ALL,R6,-     ; Set change flags
                        09AC  1737                   XMT_B_MC_CHG_FLAG(R2)
          F7D5 CF   90  09AC  1738          MOVB    GHB_B_INCARN,-            ; Set incarnation number
             4E A2      09B0  1739                   XMT_B_MC_INCARN(R2)
        50   48 A2   9E  09B2  1740          MOVAB   XMT_T_DATA(R2), R0        ; Address of start of data
          53    50   C2  09B6  1741          SUBL    R0, R3                    ; Calculate the length
       1A A2    53   B0  09B9  1742          MOVW    R3, CXB$W_BCNT(R2)        ; Set length of message
                        09BD  1743          ;
                        09BD  1744          ; Copy the node name and descriptor for the START message
                        09BD  1745          ;
     51   F7CB CF   9E  09BD  1746          MOVAB   GHB_T_MC_DATA+2,R1        ; Get address of multicast variable data
     53   FA32 CF   9E  09C2  1747          MOVAB   GHB_T_STRT_VAR,R3         ; Get address of start msg variable data
          56    53   D0  09C7  1748          MOVL    R3, R6                    ; Save address of start of variable data
          50    81   9A  09CA  1749          MOVZBL  (R1)+, R0                 ; Get size of group codes
          51    50   C0  09CD  1750          ADDL    R0, R1                    ; Calculate address of node name
          50    61   9B  09D0  1751          MOVZBW  (R1), R0                  ; Get size of node name
                50   B6  09D3  1752          INCW    R0                        ; Include the byte count
     63   61    50   28  09D5  1753          MOVC3   R0,(R1),(R3)              ; Copy the node name
          50    61   9B  09D9  1754          MOVZBW  (R1), R0                  ; Get size of node descriptor
                50   B6  09DC  1755          INCW    R0                        ; Include the byte count
     63   61    50   28  09DE  1756          MOVC3   R0,(R1),(R3)              ; Copy the node descriptor
                83   B4  09E2  1757          CLRW    (R3)+                     ; No location text, no parameters
          53    56   C2  09E4  1758          SUBL    R6, R3                    ; Calculate length of start message
F9FD CF   53    0C   A1  09E7  1759          ADDW3   #STRT_C_LENGTH, R3, -     ; Store length in holding cell
                         09ED  1760                   GHB_Q_STRT_LEN
     50   0000'8F    3C  09ED  1761 70$:      MOVZWL  #SS$_NORMAL, R0           ; Return success
F9E3 CF   01         B0  09F2  1762 80$:      MOVW    #1, GHB_W_MULTIMR         ; Send multicast msg on next tick
          007E 8F    BA  09F7  1763 90$:      POPR    #^M<R1,R2,R3,R4,R5,R6>    ; Restore registers
                         09FB  1764          ENBINT
                     05  09FE  1765          RSB
                         09FF  1766
```

```
                        09FF          .SBTTL  LT$UNIT_INIT  - Unit Initialization
                        09FF  ;++
                        09FF  ; LT$UNIT_INIT - Unit Initialization
                        09FF  ;
                        09FF  ; Functional description:
                        09FF  ;
                        09FF  ; This routine performs a simple unit initialization.
                        09FF  ;
                        09FF  ; Inputs:
                        09FF  ;
                        09FF  ;       R5 = UCB address
                        09FF  ;
                        09FF  ; Outputs:
                        09FF  ;
                        09FF  ;       R0-R3 are modified
                        09FF  ;--
                        09FF
                        09FF  LT$UNIT_INIT:
                 05   E0  09FF          BBS     #UCB$V_POWER,-          ; Skip if power fail recovery
            5A 64 A5      0A01                  UCB$L_STS(R5), 70$      ;
                          0A04
51   00000000'GF   D0     0A04          MOVL    G^TTY$GL_DPT,R1         ; Address of class dpt
      50    1E A1   3C    0A0B          MOVZWL  DPT$W_VECTOR(R1),R0     ; Locate class driver vector table
         51    50   C0    0A0F          ADDL    R0,R1                   ; Relocate base address
    0114 C5    51   D0    0A12          MOVL    R1,UCB$L_TT_CLASS(R5)   ; Set terminal class driver vector
                          0A17
    010C C5    61   D0    0A17          MOVL    CLASS_GETNXT(R1),UCB$L_TT_GETNXT(R5)
 0110 C5    04 A1   D0    0A1C          MOVL    CLASS_PUTNXT(R1),UCB$L_TT_PUTNXT(R5)
      50    28 A5   D0    0A22          MOVL    UCB$L_DDB(R5),R0        ; Get DDB address
   0C A0    10 A1   D0    0A26          MOVL    CLASS_DDT(R1),DDB$L_DDT(R0)
 0088 C5    10 A1   D0    0A2B          MOVL    CLASS_DDT(R1),UCB$L_DDT(R5) ; Set DDT address in UCB
                          0A31
            54 A5   B5    0A31          TSTW    UCB$W_UNIT(R5)          ; Special action for ucb 0
               07   13    0A34          BEQL    20$                     ; Its is unit zero
                          0A36          CLRBIT  #UCB$V_TEMPLATE,-       ; All others lose the template
                          0A36                  UCB$L_STS(R5)           ;  bit so they won't clone on assign
               1D   11    0A3B          BRB     40$                     ; Continue
  F72A CF      55   D0    0A3D  20$:    MOVL    R5, GHB_L_UCB0          ; Save address of ucb zero
                          0A42  ;
                          0A42  ; We are setting or clearing some bits in the LTA0 device here so that all
                          0A42  ; later lta devices will have them set/clear the same way.  These bits are
                          0A42  ; different from the defaults that are set in the sysgen parameters for
                          0A42  ; the other types of terminals.
                          0A42  ;
                          0A42  ;       SETBIT  #TT$V_MODEM, -          ; Set modem so we do modem
                          0A42  ;               UCB$L_TT_DECHAR(R5)     ;  processing for this terminal
                          0A42          SETBIT  #TT$V_REMOTE,-          ; Set remote so we know its a remote
                          0A42                  UCB$L_TT_DECHAR(R5)     ;  terminal forever.
                          0A48                                          ; See setup_ucb routine for reasons
                          0A48          CLRBIT  #TT2$V_AUTOBAUD,-       ; We do not autobaud here.
                          0A48                  UCB$L_TT_DECHA1(R5)     ;
                          0A4E          SETBIT  #TT2$V_HANGUP,-         ; Set default for terminal to hangup
                          0A4E                  UCB$L_TT_DECHA1(R5)     ; on logout.  this can be cleared by
                          0A54                                          ; the user from his terminal.
                          0A54          SETBIT  #TTY$V_PC_NOTIME,-      ; no timer service please
                          0A54                   UCB$W_TT_PRTCTL(R5)    ; ...
                          0A5A  40$:    ;
```

```
                    0A5A  1825          ; Finish initializing UCB.
                    0A5A  1826          ;
          0F   10   0A5A  1827          BSBB    LT$UCB_INIT             ; Initialize the terminal UCB
          0C   11   0A5C  1828          BRB     90$                     ; And leave
                    0A5E  1829
50   0134 C5   D0   0A5E  1830 70$:     MOVL    UCB$L_LT_CSB(R5), R0    ; On power fail, reset the timeout
          05   13   0A63  1831          BEQL    90$                     ; No CSB, just forget it
       66 A0   B0   0A65  1832          MOVW    CSB_W_TIMRESET(R0),-    ; And just wait on some thing
       50 A0        0A68  1833                  CSB_W_TIMEOUT(R0)       ; To happen.
          05        0A6A  1834 90$:     RSB
```

```
                        0A6B          .SBTTL  LT$UCB_INIT  - Initialize a terminal UCB
                        0A6B  ;++
                        0A6B  ; LT$UCB_INIT - Initialize a terminal UCB
                        0A6B  ;
                        0A6B  ; Functional description:
                        0A6B  ;
                        0A6B  ;       Init the terminal UCB before starting a process to use it.
                        0A6B  ;
                        0A6B  ; Inputs:
                        0A6B  ;       R5 = UCB address
                        0A6B  ;
                        0A6B  ; Outouts:
                        0A6B  ;       R5 = UCB address
                        0A6B  ;       R0, R1 clobbered
                        0A6B  ;--
                        0A6B
                        0A6B  LT$UCB_INIT:
          54 A5    B5   0A6B          TSTW    UCB$W_UNIT(R5)          ; Is this unit #0?
             3D    13   0A6E          BEQL    90$                     ; Br if yes, skip it
          5C A5    B5   0A70          TSTW    UCB$W_REFC(R5)          ; Any channels attached?
             21    12   0A73          BNEQ    40$                     ; Br if yes, don't reinit UCB
51   01   54 A5    78   0A75          ASHL    UCB$W_UNIT(R5),#1,R1    ; Build unit's bit mask
     0080 C5       51   0A7A          MOVW    R1,UCB$P_[illegible]NT(R5)  ; Save it
     0141 C5       94   0A7F          CLRB    UCB$B_[illegible]LOCID(R5)  ; Flag as available (no local slot indx)
     FA21 CF       DE   0A83          MOVAL   LT$VECTOR,-             ; Set the port address in ucb
     0118 C5            0A87                  UCB$L_TT_PORT(R5)       ;  in case port driver reloaded
     00CC C5       D4   0A8A          CLRL    UCB$L_TT_WFLINK(R5)     ; Zero to force reinit of queue head
51   0114 C5       D0   0A8E          MOVL    UCB$L_TT_CLASS(R5),R1   ; Address class vector table
          08 B1    16   0A93          JSB     @CLASS_SETUP_UCB(R1)    ; Init UCB fields
                        0A96  40$:
             01    90   0A96          MOVB    #UCB$M_LT_DATA,-        ; Look for data first time
     0142 C5            0A98                   UCB$B_LT_DATAW(R5)
          50 8F    90   0A9B          MOVB    #LAT$C_UCB_BUFSIZ,-     ; Init buffer
     014A C5            0A9E                  UCB$B_LT_MAXC(R5)       ;       ...
     014B C5       94   0AA1          CLRB    UCB$B_LT_CURC(R5)       ;       ...
     0146 C5       94   0AA5          CLRB    UCB$B_LT_TCRED(R5)      ;       ...
     0120 C5       B4   0AA9          CLRW    UCB$W_TT_OUTLEN(R5)     ; No data waiting
                   05   0AAD  90$:    RSB
```

```
                        0AAE  1875          .SBTTL  LT$STARTIO  - Start I/O routine
                        0AAE  1876 ;++
                        0AAE  1877 ; LT$STARTIO - Start I/O Operation
                        0AAE  1878 ;
                        0AAE  1879 ; Functional description:
                        0AAE  1880 ;
                        0AAE  1881 ; This routine is entered from the device independent terminal startio
                        0AAE  1882 ; Routine to enable output interrupts on an idle unit.
                        0AAE  1883 ;
                        0AAE  1884 ; Inputs:
                        0AAE  1885 ;
                        0AAE  1886 ; R3 =  character       and     cc = plus
                        0AAE  1887 ; R3 =  address         and     cc = minus      and     R2 = count
                        0AAE  1888 ; R5 = ucb address
                        0AAE  1889 ;
                        0AAE  1890 ; Outputs:
                        0AAE  1891 ;
                        0AAE  1892 ;        R5 = UCB address
                        0AAE  1893 ;        R0,R1,R2,R3 and R4 are modified
                        0AAE  1894 ;--
                        0AAE  1895
                        0AAE  1896 LT$STARTIO:                                  ; Start I/O on unit
        0760 8F     BB  0AAE  1897          PUSHR   #^M<R5,R6,R8,R9,R10>
   58   0134 C5     D0  0AB2  1898          MOVL    UCB$L_LT_CSB(R5), R8        ; Get Circuit State Block
             59     13  0AB7  1899          BEQL    85$                         ; Br if UCB is hanging up
                        0AB9  1900          ;
                        0AB9  1901          ;  Dispatch on our terminal UCB state
                        0AB9  1902          ;
                        0AB9  1903          $DISPATCH   UCB$B_LT_STATE(R5),TYPE=B,-
                        0AB9  1904          <-      ; state         ; action
                        0AB9  1905            <UCB$C_LT_STATE_KILL  168$>,- ; KILL state, flush data
                        0AB9  1906          >
                        0AC1  1907
                        0AC1  1908          ;  All other states -- okay to send data
                        0AC1  1909
             01     88  0AC1  1910          BISB    #UCB$M_LT_DATA,-            ; Assume output data available
        0142 C5         0AC3  1911                   UCB$B_LT_DATAW(R5)
                        0AC6  1912          ;
                        0AC6  1913          ;   Buffer data locally
                        0AC6  1914          ;
                        0AC6  1915 5$:      $DISPATCH   UCB$B_TT_OUTYPE(R5),TYPE=B,-
                        0AC6  1916          <-      ; type          ; action
                        0AC6  1917                  < 0     170$>,-             ; No data, allow more calls
                        0AC6  1918                  < 1     10$>,-              ; Single character
                        0AC6  1919          >
                        0AD0  1920          ;
                        0AD0  1921          ;  Default to burst data
                        0AD0  1922          ;
   53   011C C5     D0  0AD0  1923          MOVL    UCB$L_TT_OUTADR(R5),R3      ; Get output buffer address
   52   0120 C5     3C  0AD5  1924          MOVZWL  UCB$W_TT_OUTLEN(R5),R2      ;  and charcter count
             21     11  0ADA  1925          BRB     30$                         ; Continue in common path
                        0ADC  1926
   50   014A C5     9E  0ADC  1927 10$:     MOVAB   UCB$B_LT_MAXC(R5),R0        ; Get address of maxc
        60   80     91  0AE1  1928          CMPB    (R0)+,(R0)                  ; Any room left ?
             02     14  0AE4  1929          BGTR    20$                         ; Br if yes
             60     97  0AE6  1930          DECB    (R0)                        ; Make room for it
             60     96  0AE8  1931 20$:     INCB    (R0)                        ; Update count
```

```
        51   60   9A  0AEA          MOVZBL  (R0),R1                  ; Get count
        50   51   C0  0AED          ADDL    R1,R0                    ; Form destination address
        60   53   90  0AF0          MOVB    R3,(R0)                  ; Load character
                      0AF3  ;
                      0AF3  ;       We are setting up the address and count of buffered characters in
                      0AF3  ;       case we are balanced and we should send these characters in a msg
                      0AF3  ;       to come out of balanced mode
                      0AF3  ;
52   014B C5   9A  0AF3          MOVZBL  UCB$B_LT_CURC(R5), R2    ; Get count of buffered characters
53   014C C5   9E  0AF8          MOVAB   UCB$B_LT_CBUF(R5), R3    ; Get address of buffer
           00   E1  0AFD  30$:    BBC     #FLAG_V_RRF,-            ; Br if we can we send an
     03 5C A8       0AFF                  CSB_B_FLAG(R8),70$       ;  unsolicited message
        00E0   31  0B02  60$:    BRW     190$
                      0B05  ;
                      0B05  ; Transmit data to concentrator
                      0B05  ;
        0146 C5   97  0B05  70$:    DECB    UCB$B_LT_TCRED(R5)       ; Use a credit
           0A   18  0B09          BGEQ    90$                      ; Br if we have one to use
        0146 C5   96  0B0B  80$:    INCB    UCB$B_LT_TCRED(R5)       ; Else, no credits
                      0B0F  ;
                      0B0F  ;       Woops.  We find we have data and we can't send it anywhere since
                      0B0F  ;       we have no credit.  Never mind we can just leave here.  We will
                      0B0F  ;       receive a credit message eventually.  We are set non-balanced here
                      0B0F  ;       and if we leave so that all the int bits are turned off in the
                      0B0F  ;       other ucbs, and this one, then we won't get anymore embarrassing
                      0B0F  ;       startio calls.  When we get the credit message we will start things
                      0B0F  ;       up again.
                      0B0F  ;
        00CE   31  0B0F          BRW     180$                     ; Leave quietly
                      0B12
                      0B12  ;
                      0B12  ;       We have nowhere to put the data and the circuit is down, so just
                      0B12  ;       clear interrupt expected, discard the data and return
                      0B12  ;
        00C4   31  0B12  85$:    BRW     168$                     ; Leave, flush data & clear INT expected
                      0B15
   56   34 A8   7E  0B15  90$:    MOVAQ   CSB_Q_XBUFQ(R8),R6       ; Get transmit buffer queue address
        56   66   D1  0B19          CMPL    (R6), R6                 ; Is there a transmit buffer?
           ED   13  0B1C          BEQL    80$                      ; Br if no transmit buffer, ignore it
   5D A8   01   90  0B1E          MOVB    #1, CSB_B_NUM_SLOTS(R8)  ; We have data for the server now
                      0B22          SETBIT  #FLAG_V_RRF,-            ; No more unsolicited messages
                      0B22                  CSB_B_FLAG(R8)           ;
   52 A8   02   B0  0B27          MOVW    #LAT$C_HOST_TIMER, -     ; Set retransmit timer
                      0B2B                  CSB_W_XMTTMO(R8)         ;
        56   66   D0  0B2B          MOVL    (R6), R6                 ; Get the buffer address
50      014C C5   9E  0B2E          MOVAB   UCB$B_LT_CBUF(R5), R0    ; Address the UCB extension
        50   53   D1  0B33          CMPL    R3, R0                   ; Is that our buffer?
           04   12  0B36          BNEQ    100$                     ; Br if no
                      0B38  ; ***
                      0B38  ; Reading the code I noticed the following line(s) would not work if
                      0B38  ; the startio entry were to discover that it were balanced after
                      0B38  ; previous previous calls stored data in the UCB extensions because
                      0B38  ; we were not balanced. I guess that never happens, because UCB$B_LT_CURC
                      0B38  ; would not be equal to one, as assumed below, but some higher value. BEM
                      0B38  ; ***
        014B C5   97  0B38          DECB    UCB$B_LT_CURC(R5)        ; Yes. Don't save the character there
                      0B3C  100$:
```

```
   59    50 A6    9E  0B3C  1989          MOVAB   XMT_T_MDATA(R6), R9      ; Address of the slot to use (first)
5A    0149 C5    9A  0B40  1990          MOVZBL  UCB$B_LT_SLOTSZ(R5), R10; Get the maximum slot size
                      0B45  1991
                      0B45  1992          ASSUME SLT_B_DSTID EQ 0
                      0B45  1993          ASSUME SLT_B_SRCID EQ <SLT_B_DSTID+1>
                      0B45  1994          ASSUME SLT_B_COUNT EQ <SLT_B_SRCID+1>
                      0B45  1995          ASSUME SLT_B_CRED EQ <SLT_B_COUNT+1>
                      0B45  1996
                      0B45  1997          ASSUME  UCB$B_LT_LOCID EQ UCB$B_LT_REMID+1
                      0B45  1998
89    0140 C5    B0  0B45  1999          MOVW    UCB$B_LT_REMID(R5), (R9)+ ; Load remote and local ids
      89    5A   90  0B4A  2000          MOVB    R10, (R9)+                ; Save the character count
                      0B4D  2001          ;
                      0B4D  2002          ; Extend credits to concentrator if needed
                      0B4D  2003          ;
      0147 C5    81  0B4D  2004          ADDB3   UCB$B_LT_XCRED(R5),-      ; See if we need to send
      50    01         0B51  2005                  #LAT$C_MAX_RCRED,R0       ;   a credit to the remote
0148 C5    50    80  0B53  2006          ADDB    R0,UCB$B_LT_RCRED(R5)     ; Add them to total credits to send
0147 C5    50    82  0B58  2007          SUBB    R0,UCB$B_LT_XCRED(R5)     ; Credits have been extended
89    0148 C5    90  0B5D  2008          MOVB    UCB$B_LT_RCRED(R5), (R9)+ ; Load credits
      0148 C5    94  0B62  2009          CLRB    UCB$B_LT_RCRED(R5)        ; Don't send them again!
                      0B66  2010
      5A    52   D1  0B66  2011 110$:    CMPL    R2, R10                   ; Do we have more than we can handle?
            03   1B  0B69  2012          BLEQU   120$                      ; Br if no, its just fine
      52    5A   D0  0B6B  2013          MOVL    R10, R2                   ; Use just what we can handle
      5A    52   C2  0B6E  2014 120$:    SUBL    R2, R10                   ; Adjust what we have left
                      0B71  2015 ; ***
                      0B71  2016 ; Reading the code I noticed the following line(s) would not work if
                      0B71  2017 ; the startio entry were to discover that it were balanced after
                      0B71  2018 ; previous calls stored data in the UCB extensions because
                      0B71  2019 ; we were not balanced. I guess that never happens, because R2 would
                      0B71  2020 ; not reflect the UCB extension data area. BEM
                      0B71  2021 ; ***
011C C5    52   C0  0B71  2022          ADDL2   R2,UCB$L_TT_OUTADR(R5)    ; Update UCB state
0120 C5    52   A2  0B76  2023          SUBW2   R2,UCB$W_TT_OUTLEN(R5)    ; Update UCB state
            55   DD  0B7B  2024          PUSHL   R5                        ; Save the UCB address
69    63    52   28  0B7D  2025          MOVC3   R2,(R3),(R9)              ; Copy the string
            55 8ED0  0B81  2026          POPL    R5                        ; Restore the UCB address
      59    53   D0  0B84  2027          MOVL    R3,R9                     ; Set address of next available byte
            5A   D5  0B87  2028          TSTL    R10                       ; Do we have any left?
            28   13  0B89  2029          BEQL    155$                      ; Br if no, all done.
      010C  D5   16  0B8B  2030          JSB     @UCB$L_TT_GETNXT(R5)      ; Else, call back class driver for more
            19   13  0B8F  2031          BEQL    150$                      ; Br if no more to be had
            0C   14  0B91  2032          BGTR    140$                      ; Br is single character
                      0B93  2033          ;
                      0B93  2034          ;  Found a burst of data
                      0B93  2035          ;
53    011C C5    D0  0B93  2036          MOVL    UCB$L_TT_OUTADR(R5),R3    ; Get output buffer address
52    0120 C5    3C  0B98  2037          MOVZWL  UCB$W_TT_OUTLEN(R5),R2    ;  and charcter count
            C7   11  0B9D  2038          BRB     110$                      ; Else buffer address
                      0B9F  2039
                      0B9F  2040 140$:    ;
                      0B9F  2041          ;  Found a single character
                      0B9F  2042          ;
      52    01   D0  0B9F  2043          MOVL    #1, R2                    ; Else, just one character returned
      69    53   90  0BA2  2044          MOVB    R3, (R9)                  ;  and fake many chars returned
      53    59   D0  0BA5  2045          MOVL    R9, R3                    ; Copy buffer address
```

```
                BC   11  0BA8  2046          BRB     110$                        ;  and continue
                         0BAA  2047
        64 A5   02   AA  0BAA  2048 150$:    BICW    #UCB$M_INT,UCB$L_STS(R5);Allow more start io calls (checked)
                01   8A  0BAE  2049          BICB    #UCB$M_LT_DATA,-            ;  no output data available
           0142 C5       0BB0  2050                   UCB$B_LT_DATAW(R5)
     52    59   56   C3  0BB3  2051 155$:    SUBL3   R6, R9, R2                  ; Make the length of the data to
52   00000048 8F   C2  0BB7  2052          SUBL    #XMT_T_DATA, R2             ;  transmit
                59   D6  0BBE  2053          INCL    R9                          ; Round end address of data
           59   C1   CA  0BC0  2054          BICL    #1,R9                       ;       ...
        1C A6   59   D0  0BC3  2055          MOVL    R9,CXB$L_T_ENDADR(R6)       ; Store it in the CXB
           52   0C   C2  0BC7  2056          SUBL    #<<XMT_T_MDATA- -           ; Make total characters in slot
                         0BCA  2057                  XMT_T_DATA> + -             ; minus those already accounted for
                         0BCA  2058                  SLT_T_DATA>, R2             ; beyond the circuit and slot headers
        52 A6   52   90  0BCA  2059          MOVB    R2, <SLT_B_COUNT+-          ; Store that count away.
                         0BCE  2060                  XMT_T_MDATA>(R6)            ; In the first slot maxc
                         0BCE  2061
                         0BCE  2062          .IF DEFINED LT_XMT_CHECK
                         0BCE  2063          ;
                         0BCE  2064          ;  debug code
                         0BCE  2065          ;
                         0BCE  2066          MOVQ    CSB_T_CIRCHDR(R8),-             ; Move the circ header
                         0BCE  2067                  XMT_T_DATA(R6)                  ; to look at real msg
                         0BCE  2068          MOVZBL  XMT_B_NUM_SLOTS(R6), R1         ; number of valid slots
                         0BCE  2069          BNEQ    160$                            ; We have some, skip check
                         0BCE  2070          ; No slots - we must only have a circuit header!
                         0BCE  2071          MOVAB   XMT_T_MDATA(R6), R0             ; Past circuit header
                         0BCE  2072          CMPL    R0, R9                          ; Only have a circuit header?
                         0BCE  2073          BEQL    160$                            ; Br if yes, okay
                         0BCE  2074          DEBUG                                   ; Not correct
                         0BCE  2075
                         0BCE  2076          .ENDC   ;; DEFINED LT_XMT_CHECK
                         0BCE  2077
        54   58   D0     0BCE  2078 160$:    MOVL    R8, R4                          ; CSB address for call
           0E93   30     0BD1  2079          BSBW    LT$XMITONE                      ; Transmit the buffer
        0760 8F   BA     0BD4  2080          POPR    #^M<R5,R6,R8,R9,R10>
                  05     0BD8  2081          RSB
                         0BD9  2082
                         0BD9  2083 168$:    ;
                         0BD9  2084          ;  Discard output data, no session active !
                         0BD9  2085          ;
           004D   30     0BD9  2086          BSBW    LT$FLUSH_DATA                   ; Flush the output data
                         0BDC  2087
                         0BDC  2088 170$:    ;
                         0BDC  2089          ;  Allow more calls to startio
                         0BDC  2090          ;
        64 A5   02   AA  0BDC  2091          BICW    #UCB$M_INT, -                   ; Allow more calls here
                         0BE0  2092                  UCB$L_STS(R5)                   ;
        0760 8F   BA     0BE0  2093 180$:    POPR    #^M<R5,R6,R8,R9,R10>
                  05     0BE4  2094          RSB
                         0BE5  2095
                         0BE5  2096 ;
                         0BE5  2097 ;        Not balanced, so buffer data if we can.
                         0BE5  2098 ;        R2 = count
                         0BE5  2099 ;        R3 -> data
                         0BE5  2100 ;
                         0BE5  2101
54      014C C5   9E     0BE5  2102 190$:    MOVAB   UCB$B_LT_CBUF(R5), R4           ; Address of character buffer
```

```
                54   53   D1  0BEA  2103          CMPL    R3, R4                    ; Is this the character buffer?
                     28   13  0BED  2104          BEQL    210$                      ; Br if yes, check if its full
        50      014B C5   9A  0BEF  2105          MOVZBL  UCB$B_LT_CURC(R5), R0     ; Calculate amount of room
                     51   D4  0BF4  2106          CLRL    R1                        ;  left in the output buffer
  51    014A C5   50   83  0BF6  2107          SUBB3   R0, -                     ;       ...
                          0BFC  2108                  UCB$B_LT_MAXC(R5), R1     ;       ...
                52   51   D1  0BFC  2109          CMPL    R1, R2                    ; Do we have room?
                     DF   15  0BFF  2110          BLEQ    180$                      ; Br if no, so shut output off
                0120 C5   B4  0C01  2111          CLRW    UCB$W_TT_OUTLEN(R5)       ; Else, forget about the data here.
        014B C5   52   80  0C05  2112          ADDB2   R2, UCB$B_LT_CURC(R5)     ; Calculate the number of chars
           51   6440   9E  0C0A  2113          MOVAB   (R4)[R0], R1              ;  and get address to put them
                          0C0E  2114 200$:
                     55   DD  0C0E  2115          PUSHL   R5                        ; Save the UCB address
        61   63      52   28  0C10  2116          MOVC3   R2,(R3),(R1)              ; Copy the string
                     55 8ED0  0C14  2117          POPL    R5                        ; Restore the UCB address
                          0C17  2118
  014A C5   014B C5   91  0C17  2119 210$:  CMPB    UCB$B_LT_CURC(R5), -              ; Is buffer full?
                          0C1E  2120                  UCB$B_LT_MAXC(R5)
                     C0   18  0C1E  2121          BGEQ    180$                      ; Br if yes, shut off output
                010C D5   16  0C20  2122          JSB     @UCB$L_TT_GETNXT(R5)      ; Else, get more data
                     B6   13  0C24  2123          BEQL    170$                      ; No more data, so just return
                   FE9D   31  0C26  2124          BRW     5$                        ; Else, process data
                          0C29  2125
```

```
                0C29  2127          .SBTTL  LT$FLUSH_DATA  - Flush buffered output data and drain TT buffer
                0C29  2128 ;++
                0C29  2129 ; LT$FLUSH_DATA - Flush buffered output data and drain class driver buffer
                0C29  2130 ;
                0C29  2131 ; Functional description:
                0C29  2132 ;
                0C29  2133 ; Flush buffered output data and drain class driver output buffer.
                0C29  2134 ;
                0C29  2135 ; Inputs:
                0C29  2136 ;        R5 = UCB address
                0C29  2137 ;
                0C29  2138 ; Outputs:
                0C29  2139 ;        R3 is destroyed.
                0C29  2140 ;--
                0C29  2141
                0C29  2142 LT$FLUSH_DATA:
010C D5   16    0C29  2143 20$:     JSB     @UCB$L_TT_GETNXT(R5)     ; Call back for more to class driver
     02   13    0C2D  2144          BEQL    90$                      ; Br if no more to be had
     F8   11    0C2F  2145          BRB     20$                      ; Else, drain all output data
0120 C5   B4    0C31  2146 90$:     CLRW    UCB$W_TT_OUTLEN(R5)      ; Clear CLASS driver output data
014B C5   94    0C35  2147          CLRB    UCB$B_LT_CURC(R5)        ; Dump UCB output buffering
          05    0C39  2148          RSB
```

```
                 0C3A  .SBTTL  LT$ABORT  - Abort (flush) buffered output data
                 0C3A ;++
                 0C3A ; LT$ABORT - Abort buffered output data
                 0C3A ;
                 0C3A ; Functional description:
                 0C3A ;
                 0C3A ; Abort (flush) buffered output data
                 0C3A ;
                 0C3A ; Inputs:
                 0C3A ;        R5 = UCB address
                 0C3A ;
                 0C3A ; Outputs:
                 0C3A ;
                 0C3A ;--
                 0C3A
                 0C3A LT$ABORT:
0148 C5    94    0C3A         CLRB    UCB$B_LT_CURC(R5)        ; Dump UCB buffering
0120 C5    B4    0C3E         CLRW    UCB$W_TT_OUTLEN(R5)      ; Dump pending class driver data,
                 0C42                                          ;  (might have dissapp+ared)
     04    88    0C42         BISB    #UCB$M_LT_ABORT,-        ; Abort requested
0142 C5          0C44                  UCB$B_LT_DATAW(R5)
           05    0C47         RSB
                 0C48
                 0C48
                 0C48
                 0C48
                 0C48         .SBTTL  LT$FLOW_CHANGE  - Change flow control
                 0C48
                 0C48 ;+
                 0C48 ;        LT$FLOW_CHANGE
                 0C48 ;
                 0C48 ; Function: Inform terminal server of change in host flow control.
                 0C48 ;                The terminal server will mimic host policy.
                 0C48 ;
                 0C48 ; Inputs:
                 0C48 ;        R5 = UCB address
                 0C48 ;
                 0C48 ; Outputs:
                 0C48 ;
                 0C48 ;-
                 0C48
                 0C48 LT$FLOW_CHANGE:
     02    88    0C48         BISB    #UCB$M_LT_FLOW,-         ; Flow change
0142 C5          0C4A                  UCB$B_LT_DATAW(R5)
           05    0C4D         RSB
                 0C4E
```

```
                        0C4E  2197          .SBTTL  LT$XON  - Resume input stream into class driver
                        0C4E  2198 ;++
                        0C4E  2199 ; LT$XON - Resume input stream into class driver
                        0C4E  2200 ;
                        0C4E  2201 ; Inputs:
                        0C4E  2202 ;        R3 = XON character
                        0C4E  2203 ;        R5 = UCB address
                        0C4E  2204 ;
                        0C4E  2205 ; Outputs:
                        0C4E  2206 ;
                        0C4E  2207 ;        R3 is destroyed
                        0C4E  2208 ;--
                        0C4E  2209 LT$XON:
     1803 8F    BB      0C4E  2210          PUSHR   #^M<R0,R1,R11,AP>       ; Save registers
                        0C52  2211          CLRBIT  #UCB$V_LT_OVFLOW,-      ; Clear overflow indicator
                        0C52  2212                   UCB$B_LT_LATSTS(R5)
50   013C C5    D0      0C58  2213          MOVL    UCB$L_LT_INPBUF(R5),R0  ; Get address of input buffer
          3A    13      0C5D  2214          BEQL    70$                     ; Br if none, nothing to do
     0147 C5    96      0C5F  2215          INCB    UCB$B_LT_XCRED(R5)      ; Okay to return credit now
                        0C63  2216
                        0C63  2217          .IF DEFINED LT_CRED_CHECK
                        0C63  2218          BLEQ    10$                     ; Br if okay
                        0C63  2219          DEBUG                           ; Else, we went too far
                        0C63  2220 10$:
                        0C63  2221          .ENDC   ;; DEFINED LT_CRED_CHECK
                        0C63  2222
                        0C63  2223          ASSUME  JCB$B_LT_LATSTS EQ UCB$B_LT_DATAW+1
                        0C63  2224
     1001 8F    A8      0C63  2225          BISW    #<UCB$M_LT_INPUT@8>!UCB$M_LT_DATA,- ; Data potentially available
     0142 C5            0C67  2226                   UCB$B_LT_DATAW(R5)     ;   and INPUT stream in enabled
  64 A5   02    88      0C6A  2227          BISB    #UCB$M_INT,UCB$L_STS(R5); Force no data to be returned
  5C   02 A0    3C      0C6E  2228          MOVZWL  INB_W_BCNT(R0),AP       ; Get count of bytes remaining
     5B   60    3C      0C72  2229          MOVZWL  INB_W_BOFF(R0),R11      ; Get offset to start of input data
     5B   50    C0      0C75  2230          ADDL    R0,R11                  ; Calculate address of input data
                        0C78  2231          ;
                        0C78  2232          ;  Feed remaining data into class driver
                        0C78  2233          ;
     53   8B    9A      0C78  2234 30$:     MOVZBL  (R11)+,R3               ; Get next character
     0110 D5    16      0C7B  2235          JSB     @UCB$L_TT_PUTNXT(R5)    ; Pass character to class driver
          02    E0      0C7F  2236          BBS     #UCB$V_LT_OVFLOW,-      ; Br if overflow, wait for next XON
  0F 0143 C5            0C81  2237                   UCB$B_LT_LATSTS(R5),50$
     F0 5C    F5      0C85  2238          SOBGTR  AP,30$                  ; Loop if more
50   013C C5    D0      0C88  2239          MOVL    UCB$L_LT_INPBUF(R5),R0  ; Get input buffer address
        0F 10   30      0C8D  2240          BSBW    LT$DEALPOOL             ; Deallocate the input buffer
     013C C5    D4      0C90  2241          CLRL    UCB$L_LT_INPBUF(R5)     ; Zero input buffer pointer
          10    8A      0C94  2242 50$:     BICB    #UCB$M_LT_INPUT,-       ; Input stream is now disabled
     0143 C5            0C96  2243                   UCB$B_LT_LATSTS(R5)
     1803 8F    BA      0C99  2244 70$:     POPR    #^M<R0,R1,R11,AP>       ; Restore registers
                05      0C9D  2245          RSB
                        0C9E  2246
                        0C9E  2247          .SBTTL  LT$XOFF  - Stop input stream into class driver
                        0C9E  2248
                        0C9E  2249 ;++
                        0C9E  2250 ; LT$XOFF - Stop input stream into class driver
                        0C9E  2251 ;
                        0C9E  2252 ; Inputs:
                        0C9E  2253 ;        R3 = XOFF character (^S or ^G)
```

```
                        0C9E  ;       R5 = UCB address
                        0C9E  ;       AP = count of bytes remaining
                        0C9E  ;       R11 = address of input data
                        0C9E  ;
                        0C9E  ; Implicit inputs:
                        0C9E  ;
                        0C9E  ;       UCB$V_LT_INPUT bit must be on UCB$B_LT_LATSTS status byte.
                        0C9E  ;
                        0C9E  ; Outputs:
                        0C9E  ;       R3 is c stroyed
                        0C9E  ;
                        0C9E  ;--
                        0C9E  LT$XOFF:
            37   BB     0C9E          PUSHR   #^M<R0,R1,R2,R4,R5>     ; Save registers
        07  53   91     0CA0          CMPB    R3,#TTY_C_BELL          ; Is this a control_G?
            44   13     0CA3          BEQL    90$                     ; Br if yes, do nothing
            04   E1     0CA5          BBC     #UCB$V_LT_INPUT,-       ; Br if input stream not enabled
     3E 0143 C5         0CA7                   UCB$B_LT_LATSTS(R5),90$
        01  5C   D1     0CAB          CMPL    AP,#1                   ; Any data to buffer? (counting current)
            39   15     0CAE          BLEQ    90$                     ; Br if no, skip it
            5C   D7     0CB0          DECL    AP                      ; Else, account for character just passed
                        0CB2          SETBIT  #UCB$V_LT_OVFLOW,-      ; Set overflow indicator
                        0CB2                   UCB$B_LT_LATSTS(R5)
52      013C C5   D0    0CB8          MOVL    UCB$L_LT_INPBUF(R5),R2  ; Get input buffer
            16   12     0CBD          BNEQ    30$                     ; Br if one there
51      5C   0C   C1    0CBF          ADDL3   #INB_C_LENGTH,AP,R1     ; Else, calculate length of buffer
 00000000'GF   16       0CC3          JSB     G^EXE$ALONONPAGED       ; Allocate a buffer
        1D 50   E9      0CC9          BLBC    R0,90$                  ; Br if no buffer
013C C5     52   D0     0CCC          MOVL    R2,UCB$L_LT_INPBUF(R5)  ; Save address of input buffer
    08 A2   51   B0     0CD1          MOVW    R1,INB_W_SIZE(R2)       ; Set size of input buffer
                        0CD5  30$:    ASSUME  INB_B_SPARE EQ INB_B_TYPE+1
    0A A2   13   9B     0CD5          MOVZBW  #DYN$C_BUFIO,INB_B_TYPE(R2) ; Set structure type
    02 A2   5C   B0     0CD9          MOVW    AP,INB_W_BCNT(R2)       ; Set size of remaining string
        62   0C   B0    0CDD          MOVW    #INB_C_LENGTH,INB_W_BOFF(R2) ; Set offset to start of data
                        0CE0          ;
                        0CE0          ;  Don't return credit... wait for XON
                        0CE0          ;
        0147 C5   97    0CE0          DECB    UCB$B_LT_XCRED(R5)      ; Don't return credit
                        0CE4
                        0CE4          .IF DEFINED LT_CRED_CHECK
                        0CE4          CMPB    UCB$B_LT_XCRED(R5),-    ; Did we goof?
                        0CE4                  #-LAT$C_MAX_RCRED
                        0CE4          BGEQU   10$                     ; Br if okay
                        0CE4          DEBUG                           ; Else, we went too far
                        0CE4  10$:
                        0CE4          .ENDC   ;; DEFINED LT_CRED_CHECK
                        0CE4
0C A2       6B   5C   28 0CE4          MOVC3   AP,(R11),INB_C_LENGTH(R2) ; Copy the data
            37   BA     0CE9  90$:    POPR    #^M<R0,R1,R2,R4,R5>     ; Restore registers
            05          0CEB          RSB                             ; Return to caller
                        0CEC
```

```
                 0CEC  2306          .SBTTL  LT$FFI_RCV_MSG - FAST Interface Receive Complete routine
                 0CEC  2307 ;++
                 0CEC  2308 ; LT$FFI_RCV_MSG - FAST Interface Receive Complete routine
                 0CEC  2309 ;
                 0CEC  2310 ; Functional description:
                 0CEC  2311 ;
                 0CEC  2312 ; Each time a frame is received, the data is passed to the class driver
                 0CEC  2313 ; for each terminal represented within the frame. Echoed data is copied
                 0CEC  2314 ; into a frame to be transmitted back to the concentrator, and finally
                 0CEC  2315 ; the transmit frame is queued for transmission after appropriate
                 0CEC  2316 ; virtual circuit maintenance functions are performed.
                 0CEC  2317 ;
                 0CEC  2318 ; Inputs:
                 0CEC  2319 ;
                 0CEC  2320 ;        R3 = CXB address
                 0CEC  2321 ;        R4 = FFI address
                 0CEC  2322 ;
                 0CEC  2323 ;        IPL = SYNCH
                 0CEC  2324 ;
                 0CEC  2325 ; Outputs:
                 0CEC  2326 ;
                 0CEC  2327 ;        R1-R5 are preserved.
                 0CEC  2328 ;
                 0CEC  2329 ;        The interrupt is dismissed when the received frame
                 0CEC  2330 ;        has been compeletely processed. The CXB is always
                 0CEC  2331 ;        returned to the datalink driver.
                 0CEC  2332 ;
                 0CEC  2333 ;--
                 0CEC  2334
                 0CEC  2335          ASSUME  LAT$C_IPL EQ IPL$_SYNCH
                 0CEC  2336 LT$FFI_RCV_MSG:
   3FFE 8F    BB  0CEC  2337          PUSHR   #^M<R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,AP,FP> ; Save registers
                 0CF0  2338
56   18 A3    3C  0CF0  2339          MOVZWL  CXB$W_BOFF(R3),R6       ; Get offset to start of data
   56   53    C0  0CF4  2340          ADDL    R3,R6                   ; Get pointer to start of data
                 0CF7  2341
                 0CF7  2342          .IF DEFINED LT_HISTORY
                 0CF7  2343 ;
                 0CF7  2344 ;        Save history of receives
                 0CF7  2345 ;
   50   66    9E  0CF7  2346          MOVAB   RCV_B_FLAG(R6), R0      ; Address is circuit header
      F9FA    30  0CFA  2347          BSBW    LT$HISTORY              ; Store the data
                 0CFD  2348          .ENDC   ;; DEFINED LT_HISTORY
                 0CFD  2349
                 0CFD  2350          INC_CTR GHB$L_RCOUNT+GHB_AREA   ; Count frame
                 0D07  2351 ;
                 0D07  2352 ;  Validate received frame
                 0D07  2353 ;
50   02 A6    3C  0D07  2354          MOVZWL  RCV_W_DSTID(R6),R0      ; Get remote vci
        76    12  0D0B  2355          BNEQ    120$                    ; Got index, okay
              91  0D0D  2356          CMPB    #<MTYP_C_START@FLAG_V_MTYPE>!- ; Is this a start message?
                 0D0E  2357                  FLAG_M_MASTER,-
   66   06        0D0E  2358                  RCV_B_FLAG(R6)
        29    12  0D10  2359          BNEQ    91$                     ; Br if no, signal protocol error
                 0D12  2360 ;
                 0D12  2361 ;  START message received on an inactive circuit, try to start up a
                 0D12  2362 ;  new circuit.
```

```
                 0D12  ;
F40A CF    D5    0D12          TSTL    GHB$L_UCB+GHB_AREA       ; Data link UCB still here?
     4F    13    0D16          BEQL    99$                      ; No, then don't accept circuit
  04 A6    B5    0D18          TSTW    RCV_W_SRCID(R6)          ; Is the source ID zero?
     32    13    0D1B          BEQL    96$                      ; Br if yes, protocol error
  06 A6    95    0D1D          TSTB    RCV_B_SEQ(R6)            ; Is this the correct sequenced msg?
     32    12    0D20          BNEQ    97$                      ; Br if not, protocol error
   0577    30    0D22          BSBW    LT$NEW_CIRCUIT           ; Initialize and/or allocate CSB
  10 50    E8    0D25          BLBS    R0, 19$                  ; Go ahead, CSB address in R8
                 0D28                                           ;  startup any transmits waiting
     50    D5    0D28          TSTL    R0                       ; Are we ignoring a duplicate start msg?
     40    12    0D2A          BNEQ    100$                     ; Br if yes, just retransmit start msg
                 0D2C          INC_CTR GHB$L_RESOURCE+GHB_AREA  ; Else, no CSB to use
     2F    11    0D36          BRB     99$                      ; So just ignore it
                 0D38
   00B5    31    0D38  19$:    BRW     190$                     ; Long branch
                 0D3B
5D   00    9A    0D3B  91$:    MOVZBL  #GHB$V_START,FP          ; Invalid START message received
     17    11    0D3E          BRB     98$                      ; continue
5D   02    9A    0D40  93$:    MOVZBL  #GHB$V_CSBRANGE,FP       ; Bad range on CSB index
     12    11    0D43          BRB     98$                      ; continue
5D   03    9A    0D45  94$:    MOVZBL  #GHB$V_CSBINVALID,FP     ; Invalid CSB index
     0D    11    0D48          BRB     98$                      ; continue
5D   04    9A    0D4A  95$:    MOVZBL  #GHB$V_CSBSTALE,FP       ; Stale CSB index
     08    11    0D4D          BRB     98$                      ; continue
5D   06    9A    0D4F  96$:    MOVZBL  #GHB.V_INVALIDREMID,FP   ; Invalid remote circuit id
     03    11    0D52          BRB     98$                      ; continue
5D   0A    9A    0D54  97$:    MOVZBL  #GHB$V_INVALIDSEQ,FP     ; Invalid sequence number
                 0D57  ...     BRB     98$                      ; continue
                 0D57  98$:    SETBIT  FP,GHB$L_PROTOMASK+GHB_AREA ; Set bit mask
                 0D5D          INC_CTR GHB$L_PROTOCOL+GHB_AREA  ; Increment counter
3FFE 8F    BA    0D67  99$:    POPR    #^M<R7,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,AP,FP> ; Restore registers
           05    0D6B          RSB                              ; Return to caller
                 0D6C
                 0D6C          ;
                 0D6C          ;  A duplicate frame was received
                 0D6C          ;
                 0D6C  100$:   INC_ TR GHB$L_DUPLMSG+GHB_AREA   ; We received a duplicate message
                 0D76          INC_CTR GHB$L_RETRANS+GHB_AREA   ; We are retransmitting a message
     04AE    31  0D80          BRW     LT$REPLY_REXMIT          ; This data is stale, resend buffers
                 0D83  ;
                 0D83  ;
                 0D83  ;  Non-zero circuit ID !
                 0D83  ;        - circuit must be active already
                 0D83  ;
                 0D83  ;  Inputs:      R0 = CSB index
                 0D83  ;               R3 = CXB address
                 0D83  ;               R6 = Pointer to start of data
                 0D83  ;
                 0D83  ;
     51    50    9A    0D83  120$:   MOVZBL  R0,R1                    ; Get CSB index
     20    51    D1    0D86          CMPL    R1, #LAT$C_MAX_CSBS      ; Range check
           B5    14    0D89          BGTR    93$                      ; Br if bad range, protocol error
58   F2A4 CF41   D0    0D8B          MOVL    GHB_L_CSBTABLE-4[R1],R8  ; Get CSB address
           B2    13    0D91          BEQL    94$                      ; Br if invalid CSB, protocol error
   60 A8   50    B1    0D93          CMPW    R0,CSB_W_LOCID(R8)       ; Valid reference ?
           B1    12    0D97          BNEQ    95$                      ; Br if no, Stale reference to CSB
```

```
                        0D99  2420          INC_CTR CSB_Z_LCB+LCB_L_MSG_RCV(R8) ; Count one more rcvd msg
                        0DA1  2421          ;
                        0DA1  2422          ;  Check ACK number and complete any transmits acknowledged
                        0DA1  2423          ;
53   07 A6   9A         0DA1  2424 130$:    MOVZBL  RCV_B_ACK(R6), R3        ; Get the ack from the message
65 A8   53   90         0DA5  2425          MOVB    R3, CSB_B_RACK(R8)       ;
53   62 A8   82         0DA9  2426          SUBB    CSB_B_XSEQ(R8), R3       ; Set up the normallized ack
52   3C A8   7E         0DAD  2427          MOVAQ   CSB_Q_XWAITQ(R8), R2     ; Look through the wait queue for it
     51   52   D0        0DB1  2428          MOVL    R2, R7                   ;
                        0DB4  2429
     51   61   D0        0DB4  2430 140$:    MOVL    (R1), R1                 ; Next item in queue
     52   51   D1        0DB7  2431 150$:    CMPL    R1, R2                   ; End of queue
          34   13        0DBA  2432          BEQL    190$                     ; Done with that search
50   4E A1   62 A8   83 0DBC  2433          SUBB3   CSB_B_XSEQ(R8), -        ; Normallize the seq of message
                        0DC2  2434                  XMT_B_SEQ(R1), -         ;
                        0DC2  2435                  R0                       ;
     53   50   91        0DC2  2436          CMPB    R0, R3                   ; If the seq is greater than ack,
          ED   14        0DC5  2437          BGTR    140$                     ; wait on another ack
     54   61   D0        0DC7  2438          MOVL    (R1), R4                 ; Save link to next item
     51   61   0F        0DCA  2439          REMQUE  (R1), R1                 ; Remove acked message from wait queue
38 B8   61   0E         0DCD  2440          INSQUE  (R1), -                  ; And insert on tail of transmit queue
                        0DD1  2441                  @CSB_Q_XBUFQ+4(R8)       ;
   59 A8   97           0DD1  2442          DECB    CSB_B_XMTCNT(R8)         ; One less to transmit
     51   54   D0        0DD4  2443          MOVL    R4, R7                   ; Restore link
          DE   11        0DD7  2444          BRB     150$                     ;
                        0DD9  2445 170$:    ;
                        0DD9  2446          ;  Invalid sequence number received
                        0DD9  2447          ;
                        0DD9  2448          INC_CTR CSB_Z_LCB+LCB_W_SEQ_ERR(R8),W ; Bad sequence number
          89   11        0DE1  2449          BRB     100$                     ; Also count as duplicate
                        0DE3  2450 180$:    ;
                        0DE3  2451          ;  No transmit buffer available
                        0DE3  2452          ;
                        0DE3  2453          INC_CTR GHB$L_NOXBFR+GHB_AREA    ; Not transmit buffer available
   FF77   31            0DED  2454          BRW     99$                      ; And exit
                        0DF0  2455          ;
                        0DF0  2456          ;  Make sure that we have a transmit buffer to respond with
                        0DF0  2457          ;
52   34 A8   7E         0DF0  2458 190$:    MOVAQ   CSB_Q_XBUFQ(R8), R2      ; Look to see if we have at least one
   62   52   D1         0DF4  2459          CMPL    R2, (R2)                 ; transmit buffer
          EA   13        0DF7  2460          BEQL    180$                     ; Br if None, just retransmit and wait
                        0DF9  2461
   66 A8   B0           0DF9  2462          MOVW    CSB_W_TIMRESET(R8),-     ; Reset timer for timeout of circuit
   50 A8                0DFC  2463                  CSB_W_TIMEOUT(R8)        ; Since we got a valid message.
64 A8   06 A6   91      0DFE  2464          CMPB    RCV_B_SEQ(R6),CSB_B_RSEQ(R8) ; Right sequence number ?
          D4   12        0E03  2465          BNEQ    170$                     ; Br if no, retransmit all messages
                        0E05  2466
                        0E05  2467
                        0E05  2468 ;
                        0E05  2469 ;  Dispatch on message type
                        0E05  2470 ;
                        0E05  2471 ;   Inputs:
                        0E05  2472 ;
                        0E05  2473 ;       R8 = CSB address
                        0E05  2474 ;       R6 = CXB address
                        0E05  2475 ;
   64 A8   96           0E05  2476          INCB    CSB_B_RSEQ(R8)           ; Next time it's the next number
```

```
63 A8   06 A6   90  0E08  2477          MOVB    RCV_B_SEQ(R6),CSB_B_XACK(R8) ; Tell other guy we got this one
           02   EF  0E0D  2478          EXTZV   #FLAG_V_MTYPE,-          ; Obtain the message type
           06       0E0F  2479                  #FLAG_S_MTYPE,-
      50   66       0E10  2480                  RCV_B_FLAG(R6), R0
                    0E12  2481
                    0E12  2482          $DISPATCH       R0,TYPE=B,-      ; Dispatch on message type
                    0E12  2483          <-      ; msg type       ; action
                    0E12  2484                  <MTYP_C_RUN     LT$RCV_RUN_MSG>,- ; RUN message received
                    0E12  2485                  <MTYP_C_START   LT$RCV_START_MSG>,- ; START message received
                    0E12  2486                  <MTYP_C_HALT    LT$RCV_HALT_MSG>,- ; HALT message received
                    0E12  2487          >
                    0E1C  2488
                    0E1C  2489 ;;       BRB     LT$FORCE_HALT            ; All others, force halt
```

```
                0E1C          .SBTTL  PROCESS RECEIVED MESSAGE
                0E1C  ;++
                0E1C  ;
                0E1C  ;  Other guy is halted or all screwed up
                0E1C  ;
                0E1C  ;--
                0E1C
                0E1C  LT$FORCE_HALT:            ; receive halt msg
                0E1C                            ; invalid circuit state in msg
                0E1C                            ; invalid circuit state in CSB
                0E1C                            ; invalid local vci
                0E1C          SETBIT  #GHB$V_HALT,-
                0E1C                  GHB$L_PROTOMASK+GHB_AREA ; Circuit forced to halted
                0E22          INC_CTR GHB$L_PROTOCOL+GHB_AREA ; Increment protocol error counter
                0E2C
                0E2C  LT$STATE_HALT:
08C2    30      0E2C          BSBW    LT$CIRCDEAD             ; Circuit is dead now
03FF    31      0E2F          BRW     LT$REPLY_REXMIT         ; Transmit halt message
                0E32
                0E32  ;++
                0E32  ;
                0E32  ;  Other system is halted
                0E32  ;
                0E32  ;--
                0E32
                0E32  LT$RCV_HALT_MSG:
08BC    30      0E32          BSBW    LT$CIRCDEAD             ; Circuit is dead now
03F4    31      0E35          BRW     LT$REPLY_DONE           ; All done
                0E38
                0E38  ;++
                0E38  ;
                0E38  ;  Other guy is starting up
                0E38  ;
                0E38  ;--
                0E38
                0E38  LT$RCV_START_MSG:
                0E38
                0E38          $DISPATCH CSB_B_STATE(R8),TYPE=B,- ; Dispatch on our CSB state
                0E38          <- ; state                ; action
                0E38          <CSB_C_STATE_HALT         LT$STATE_START>,- ; Enter the START state
                0E38          <CSB_C_STATE_START        LT$REPLY_MSG>,-  ; Re-send START message
                0E38          <CSB_C_STATE_RUN          LT$STATE_HALT>,- ; Enter the HALTED state
                0E38          >
                0E43
D7      11      0E43          BRB     LT$FORCE_HALT           ; What kind of kinky state is this ?
                0E45
                0E45  ;++
                0E45  ;
                0E45  ;  Other guy is running
                0E45  ;
                0E45  ;--
                0E45
                0E45  LT$RCV_RUN_MSG:
                0E45
                0E45          $DISPATCH       CSB_B_STATE(R8),TYPE=B,- ; Dispatch on our CSB state
                0E45          <-      ; state                 ; action
                0E45                  <CSB_C_STATE_HALT       LT$REPLY_MSG>,- ; Send STOP message
```

```
                0E45  2548                  <CSB_C_STATE_START      LT$STATE_ENTER_RUN>,- ; Enter RUN state
                0E45  2549                  <CSB_C_STATE_RUN        LT$STATE_RUN>,- ; Stay in RUN state
                0E45  2550          >
                0E50  2551
CA    11        0E50  2552          BRB     LT$FORCE_HALT           ; What kind of kinky state is this ?
```

```
                        0E52  .SBTTL  CIRCUIT STATE TRANSITIONS
                        0E52 ;++
                        0E52 ;
                        0E52 ;  We can go only from halted to starting
                        0E52 ;
                        0E52 ;--
                        0E52
                        0E52 LT$STATE_START:
          01    90      0E52          MOVB    #CSB_C_STATE_START,-      ;
       0B A8            0E54                  CSB_B_STATE(R8)           ; Now we are starting
          04    90      0E56          MOVB    #MTYP_C_START@FLAG_V_MTYPE,- ; Set the start message type
       5C A8            0E58                  CSB_B_FLAG(R8)            ;
       04 A6    B0      0E5A          MOVW    RCV_W_SRCID(R6),-         ;
       5E A8            0E5D                  CSB_W_REMID(R8)           ; Save his circuit id
  59   34 A8    D0      0E5F          MOVL    CSB_Q_XBUFQ(R8),R9        ; Format a start message
50   F585 CF    9E      0E63          MOVAB   GHB_T_STRT_MSG,R0         ; Get start address of template data
60   F57E CF    28      0E68          MOVC3   GHB_W_STRT_LEN,(R0),-     ; Copy the start message
       50 A9            0E6D                   XMT_T_MDATA(R9)
  1C A9   53    D0      0E6F          MOVL    R3, CXB$L_T_ENDADR(R9)    ; Store end address of data
     03B0       31      0E73          BRW     LT$REPLY_ALT              ; Go send the reply message
                        0E76
                        0E76 ;++
                        0E76 ;
                        0E76 ;  We transit to the running state only from the starting state
                        0E76 ;
                        0E76 ;--
                        0E76
                        0E76 LT$STATE_ENTER_RUN:                        ; Enter the RUNNING state
          02    90      0E76          MOVB    #CSB_C_STATE_RUN,-        ; Now we are running
       0B A8            0E78                  CSB_B_STATE(R8)
          00    90      0E7A          MOVB    #MTYP_C_RUN@FLAG_V_MTYPE,- ; Set the run message type
       5C A8            0E7C                  CSB_B_FLAG(R8)
                        0E7E
                        0E7E ;++
                        0E7E ;
                        0E7E ;  Update state from validated frame
                        0E7E ;
                        0E7E ;--
                        0E7E
                        0E7E LT$STATE_RUN:
       04 A6    B1      0E7E          CMPW    RCV_W_SRCID(R6),-         ; Referencing a valid circuit?
       5E A8            0E81                  CSB_W_REMID(R8)
          97    12      0E83          BNEQ    LT$FORCE_HALT             ; Br if not, force a halt
59     34 A8    D0      0E85          MOVL    CSB_Q_XBUFQ(R8),R9        ; Get transmit buffer address
                        0E89          ;
                        0E89          ;  Set up for the receive loop
                        0E89          ;
                        0E89          CLRBIT  #FLAG_V_RRF,-             ; No response requested until
                        0E89                  CSB_B_FLAG(R8)            ;  a credit is used
       5D A8    94      0E8E          CLRB    CSB_B_NUM_SLOTS(R8)       ; Assume no output data (balanced)
       52 A8    B4      0E91          CLRW    CSB_W_XMTTMO(R8)          ; Circuit state is no longer timed
57     08 A6    9E      0E94          MOVAB   RCV_T_MDATA(R6),R7        ; Receive buffer first slot address
59     50 A9    9E      0E98          MOVAB   XMT_T_MDATA(R9),R9        ; Transmit buffer first slot address
5A     6A A8    3C      0E9C          MOVZWL  CSB_W_MAX_MSGSIZ(R8),R10 ; Total chars available to slots
5D     01 A6    9A      0EA0          MOVZBL  RCV_B_NUM_SLOTS(R6),FP    ; Get slot loop count
          03    12      0EA4          BNEQ    SLOT_LOOP_IN              ; Got some input data
      016E      31      0EA6          BRW     SLOT_LOOP_OUT             ; Go process output buffer
```

0EA9 2611

```
                    0EA9  2613          .SBTTL  PROCESS RECEIVED SLOT DATA
                    0EA9  2614 ;++
                    0EA9  2615 ; PROCESS RECEIVED SLOT DATA
                    0EA9  2616 ;
                    0EA9  2617 ; Functional description:
                    0EA9  2618 ;
                    0EA9  2619 ; Now we are set up to pass data to and from the class driver:
                    0EA9  2620 ;
                    0EA9  2621 ; The following loop copies data from the receive buffer into the class
                    0EA9  2622 ; driver.
                    0EA9  2623 ;
                    0EA9  2624 ; Inputs:
                    0EA9  2625 ;        R0-R4 available
                    0EA9  2626 ;        R5 =  UCB address
                    0EA9  2627 ;        R6 =  CXB address
                    0EA9  2628 ;        R7 =  current receive buffer slot address
                    0EA9  2629 ;        R8 =  CSB address
                    0EA9  2630 ;        R9 =  current transmit buffer slot address
                    0EA9  2631 ;        R10 = transmit buffer bytes left
                    0EA9  2632 ;        R11 = scratch (used as current input slot character copy base register)
                    0EA9  2633 ;        AP =  scratch (used as loop count register)
                    0EA9  2634 ;        FP =  number of slots to process in received message
                    0EA9  2635 ;
                    0EA9  2636 ; Outputs:
                    0EA9  2637 ;        none.
                    0EA9  2638 ;
                    0EA9  2639 ;   We continue by processing the output slot loop.
                    0EA9  2640 ;
                    0EA9  2641 ;--
                    0EA9  2642          .ENABLE LOCAL_BLOCK
                    0EA9  2643 SLOT_LOOP_IN:                                ; Input data slot loop
          51   67   9A  0EA9  2644          MOVZBL  SLT_B_DSTID(R7),R1      ; Is this slot alive ?
               40   12  0EAC  2645          BNEQ    20$                     ; Br if possible
            01 A7   95  0EAE  2646          TSTB    SLT_B_SRCID(R7)         ; Is the remote index valid?
               38   13  0EB1  2647          BEQL    10$                     ; Br if not, ignore slot
                    0EB3  2648          ;
                    0EB3  2649          ;  Zero local slot index with non-zero remote (someone's attempting
                    0EB3  2650          ;    to login). Must be a start slot!
                    0EB3  2651          ;
50  03 A7   04   04  EF  0EB3  2652          EXTZV   #4,#4,SLT_B_CRED(R7),R0 ; Get slot type
          50   09   91  0EB9  2653          CMPB    #SLT_C_STR_SLOT,R0      ; Is this a start slot ?
               5A   12  0EBC  2654          BNEQ    LOOP_ABORT              ; Br if not, ignore slot
             0651   30  0EBE  2655          BSBW    LT$ACC_UCB              ; Try to allocate an LT UCB
            27 50   E8  0EC1  2656          BLBS    R0, 10$                 ; Br if success
                    0EC4  2657          INC_CTR GHB$L_RESOURCE+GHB_AREA ; Else, no UCB for us to use
                    0ECE  2658          ;
                    0ECE  2659          ;  Send back a reject slot in the case of resource errors
                    0ECE  2660          ;
                    0ECE  2661          ASSUME  SLT_S_REJ_SLOT&1 EQ 1   ; Assume we need rounding
          06   5A   D1  0ECE  2662          CMPL    R10,-                   ; Enough room for REJECT slot header?
                    0ED1  2663                  #SLT_S_REJ_SLOT+1
               18   19  0ED1  2664          BLSS    10$                     ; Nope, just ignore the data
            5D A8   96  0ED3  2665          INCB    CSB_B_NUM_SLOTS(R8)     ; Increment slot count
       89   01 A7   90  0ED6  2666          MOVB    SLT_B_SRCID(R7),(R9)+   ; Copy local id from receive message
               89   94  0EDA  2667          CLRB    (R9)+                   ; Tell him we are not here !
          89   01   90  0EDC  2668          MOVB    #SLT_S_REJ_LEN,(R9)+    ; Load slot byte count
       69   C0 8F   90  0EDF  2669          MOVB    #SLT_C_REJ_SLOT@4,(R9)  ; Load REJECT slot format
```

```
                89   05   88  0EE3  2670          BISB    #SLT_C_RESOURCE,(R9)+    ; Load reason code (resource error)
                                0EE6  2671          ASSUME  SLT_S_REJ_SLOT&1 EQ 1    ; Assume we need rounding
                     89   94    0EE6  2672          CLRB    (R9)+                    ; Round up to even boundary
                5A   06   C2    0EE8  2673          SUBL2   #SLT_S_REJ_SLOT+1,R10    ; Adjust bytes left for other slots
                   00EC   31    0EEB  2674 10$:     BRW     NEXT_SLOT                ; Just process the next slot.
                                0EEE  2675 20$:     ;
                                0EEE  2676          ;  Validate UCB reference - non-zero local index
                                0EEE  2677          ;
             69 A8   51   91    0EEE  2678          CMPB    R1, CSB_B_MAX_SLOTS(R8) ; Do a range check on the local index
                     58   1A    0EF2  2679          BGTRU   90$                      ; Br if not in range
          55    68 A841   D0    0EF4  2680          MOVL    CSB_L_UCBLST-4(R8)[R1],R5 ; Get UCB address
                     2D   18    0EF9  2681          BGEQ    70$                      ; That slot doesn't live here anymore
      0140 C5    01 A7   91    0EFB  2682          CMPB    SLT_B_SRCID(R7),UCB$B_LT_REMID(R5) ; Valid remote reference?
                     63   13    0F01  2683          BEQL    LOAD_CREDITS             ; Br if yes, process slot
                01 A7   95    0F03  2684          TSTB    SLT_B_SRCID(R7)          ; Else, are we hanging up from remote?
                     10   12    0F06  2685          BNEQ    LOOP_ABORT               ; Br if no, invalid slot
                                0F08  2686 ;
                                0F08  2687 ;  Received a zero in the slot source-id - is this a stop slot?
                                0F08  2688 ;
   50   03 A7   04   04   EF    0F08  2689 40$:     EXTZV   #4,#4,SLT_B_CRED(R7),R0 ; Get slot type
                50   0D   91    0F0E  2690          CMPB    #SLT_C_STP_SLOT,R0       ; Is this a STOP slot?
                     05   12    0F11  2691          BNEQ    LOOP_ABORT               ; Br if not, do nothing
                   0738   30    0F13  2692          BSBW    LT$HANGUP_UCB_NOW        ; Hangup the UCB to logout process
                     4B   11    0F16  2693          BRB     NEXT_SLOT0               ; Process next slot
                                0F18  2694 ;
                                0F18  2695 ;  Flag protocol error and ignore slot
                                0F18  2696 ;
                                0F18  2697 LOOP_ABORT:
                                0F18  2698          SETBIT  #GHB$V_INVALIDLOCID,-
                                0F18  2699                  GHB$L_PROTOMASK+GHB_AREA ; Invalid remote ID or slot format
                                0F1E  2700          INC_CTR GHB$L_PROTOCOL+GHB_AREA ;
                                0F28  2701
                                0F28  2702 ;
                                0F28  2703 ;  Conditionally send stop slot
                                0F28  2704 ;
             01 A7   95    0F28  2705 70$:     TSTB    SLT_B_SRCID(R7)          ; Check remote slot index
                     36   13    0F2B  2706          BEQL    NEXT_SLOT0               ; Br if empty slot
                                0F2D  2707
                                0F2D  2708          ASSUME  SLT_S_STP_SLOT&1 EQ 1    ; Assume we need rounding
          06         5A   D1    0F2D  2709          CMPL    R10,-                    ; Enough room for STOP slot header?
                                0F30  2710                  #SLT_S_STP_SLOT+1
                     31   19    0F30  2711          BLSS    NEXT_SLOT0               ; Nope, just ignore the data
             5D A8   96    0F32  2712          INCB    CSB_B_NUM_SLOTS(R8)      ; Increment slot count
      89     01 A7   90    0F35  2713          MOVB    SLT_B_SRCID(R7),(R9)+    ; Copy local id from receive message
                     89   94    0F39  2714          CLRB    (R9)+                    ; Tell him we are not here !
          89         01   90    0F3B  2715          MOVB    #SLT_S_STP_LEN,(R9)+     ; Load slot byte count
      89     D0 8F   90    0F3E  2716          MOVB    #SLT_C_STP_SLOT@4,(R9)+ ; Load stop slot format
          89         03   90    0F42  2717          MOVB    #SLT_C_INVSLOT,(R9)+     ; load reason code
                                0F45  2718          ASSUME  SLT_S_STP_SLOT&1 EQ 1    ; Assume we need rounding
                     89   94    0F45  2719          CLRB    (R9)+                    ; round up to even boundary
          5A         06   C2    0F47  2720          SUBL2   #SLT_S_STP_SLOT+1,R10    ; Adjust bytes left for other slots
                     17   11    0F4A  2721          BRB     NEXT_SLOT0               ; Go do next input slot
                                0F4C  2722 ;
                                0F4C  2723 ;  Range check error, is this really a disconnect slot ?
                                0F4C  2724 ;
             01 A7   95    0F4C  2725 90$:     TSTB    SLT_B_SRCID(R7)          ; Empty remote slot index? (stop slot?)
                     12   13    0F4F  2726          BEQL    NEXT_SLOT0               ; Br if yes, ignore the slot.
```

```
                0F51  2727          SETBIT  #GHB$V_INVALIDREMID,-
                0F51  2728                  GHB$L_PROTOMASK+GHB_AF_A; Invalid LOCID with REMID non_zero
                0F57  2729          INC_CTR GHB$L_PROTOCOL+GHB_AREA ; Count as protocol error
        C5   11 0F61  2730          BRB     70$                     ; Send a stop slot
                0F63  2731
                0F63  2732 NEXT_SLOT0:
      0074   31 0F63  2733          BRW     NEXT_SLOT               ; Branch aid
                0F66  2734
                0F66  2735          .DISABLE LOCAL_BLOCK
```

```
                                0F66          .SBTTL  PROCESS SLOT CREDITS RECEIVED
                                0F66  ;++
                                0F66  ;
                                0F66  ;  Update UCB credit account from received message slots
                                0F66  ;
                                0F66  ;--
                                0F66  LOAD_CREDITS:
          50   03 A7   90       0F66          MOVB    SLT_B_CRED(R7),R0       ; Get credits and slot type
                                0F6A          ASSUME  SLT_C_DA_SLOT EQ 0      ; assumption
                     2C   18    0F6A          BGEQ    50$                     ; Got a DATA_A slot
50   03 A7   04      00   EF    0F6C          EXTZV   #0, #4, SLT_B_CRED(R7), R0 ; Get the credit counts
51   03 A7   04      04   EF    0F72          EXTZV   #4, #4, SLT_B_CRED(R7), R1 ; Get the slot type
                                0F78          $DISPATCH       R1,TYPE=B,-     ; Dispatch on slot type
                                0F78          <-      ; slot type     ; action
                                0F78                  <SLT_C_DB_SLOT  20$>,-  ; DATA_B slot????
                                0F78                  <SLT_C_ATT_SLOT 40$>,-  ; Attention slot
                                0F78          >
                                0F80
                   FF95   31    0F80          BRW     LOOP_ABORT              ; All others, including Reject slot
                                0F83
                                0F83  ;  DATA_B slot
                                0F83
                     01   88    0F83  20$:    BISB    #UCB$M_LT_DATA,-        ; Remember to at least send back the
                0142 C5         0F85                   UCB$B_LT_DATAW(R5)     ;    credits!
                0147 C5   96    0F88          INCB    UCB$B_LT_XCRED(R5)      ; One less credit on remote end (negative)
                     03   15    0F8C          BLEQ    40$                     ; Br if credits not exceeded, proceed
                   06CA   30    0F8E          BSBW    LT$HANGUP_UCB           ; Else, Hangup the UCB to logout process
                                0F91
                                0F91  ;  ATTention slot, IGNORE DATA
                                0F91
            0146 C5   50   80   0F91  40$:    ADDB    R0,UCB$B_LT_TCRED(R5)   ; Update credits
                     42   11    0F96          BRB     NEXT_SLOT               ; Skip to next slot
                                0F98
                                0F98  ;  RUN slot
                                0F98
            0146 C5   50   80   0F98  50$:    ADDB    R0,UCB$B_LT_TCRED(R5)   ; Update credits
                                0F9D
                                0F9D
                                0F9D          .IF DEFINED LT_CRED_CHECK
                                0F9D          CMPB    UCB$B_LT_TCRED(R5),#2
                                0F9D          BLEQ    60$                     ; Br if ok
                                0F9D          SETBIT  #GHB$V_BADCREDITS,-
                                0F9D                  GHB$L_PROTOMASK+GHB_AREA; Too many credits in host
                                0F9D          INC_CTR GHB$L_PROTOCOL+GHB_AREA ; Too many credits in host
                                0F9D          BSBW    LT$HANGUP_UCB           ; Hangup the UCB to logout process
                                0F9D  60$:
                                0F9D          .ENDC   ;; DEFINED LT_CRED_CHECK
                                0F9D
                                0F9D  ;
                                0F9D  ;  Move terminal data through class driver
                                0F9D  ;
                                0F9D  ;       R3 - contains the character
                                0F9D  ;       R5 - UCB address
                                0F9D  ;
   5C   02 A7   9A              0F9D          MOVZBL  SLT_B_COUNT(R7),AP      ; Character count for this slot
                     37   13    0FA1          BEQL    NEXT_SLOT               ; No input here
   5B   04 A7   9E              0FA3          MOVAB   SLT_T_DATA(R7),R11      ; Source of characters
```

```
2D 0143 C5   00   E0  0FA7  2794          BBS     #UCB$V_LT_HANGUP, -      ; If device is hanging up, do not con-
                      0FAD  2795                  UCB$B_LT_LATSTS(R5),NEXT_SLOT ; fuse class driver with more data
        0147 C5   96  0FAD  2796          INCB    UCB$B_LT_XCRED(R5)       ; One less credit on remote end (negative)
             05   15  0FB1  2797          BLEQ    70$                      ; Br if credits not exceeded, proceed
           06A5   30  0FB3  2798          BSBW    LT$HANGUP_UCB            ; Else, Hangup the UCB to logout process
             22   11  0FB6  2799          BRB     NEXT_SLOT                ; Continue
                      0FB8  2800 70$:     ;
                      0FB8  2801          ;  Indicate that input stream is running again, and data is potentially
                      0FB8  2802          ;  available.
                      0FB8  2803          ;
                      0FB8  2804          ASSUME  UCB$B_LT_LATSTS EQ UCB$B_LT_DATAW+1
                      0FB8  2805
        1001 8F   A8  0FB8  2806          BISW    #<UCB$M_LT_INPUT@8>!UCB$M_LT_DATA,- ; Data potentially available
        0142 C5       0FBC  2807                   UCB$B_LT_DATAW(R5)      ;   and INPUT stream in enabled
     64 A5   02   88  0FBF  2808          BISB    #UCB$M_INT,UCB$L_STS(R5); Force no data to be returned
                      0FC3  2809          ;
                      0FC3  2810          ;  Dump characters into class driver
                      0FC3  2811          ;
        53   8B   9A  0FC3  2812 80$:     MOVZBL  (R11)+,R3                ; Get next character
             02   E0  0FC6  2813          BBS     #UCB$V_LT_OVFLOW,-       ; Br if overflow, wait for XON
     09 0143 C5       0FC8  2814                   UCB$B_LT_LATSTS(R5),90$
        0110 D5   16  0FCC  2815          JSB     @UCB$L_TT_PUTNXT(R5)     ; Pass character to class driver
             1B   12  0FD0  2816          BNEQ    ECHO_ERROR               ; Br if character being echoed
           EE 5C   F5  0FD2  2817          SOBGTR  AP,80$                   ; Do all characters
                      0FD5  2818
             10   8A  0FD5  2819 90$:     BICB    #UCB$M_LT_INPUT,-        ; Input stream is now disabled
        0143 C5       0FD7  2820                   UCB$B_LT_LATSTS(R5)
                      0FDA  2821 ;
                      0FDA  2822 ;  Bump pointers to the next slot
                      0FDA  2823 ;
                      0FDA  2824 NEXT_SLOT:
     50   02 A7   9A  0FDA  2825          MOVZBL  SLT_B_COUNT(R7), R0      ; Max bytes in the slot
        05 A740   9E  0FDE  2826          MOVAB   <SLT_T_DATA+1>(R7)-      ; Compute address of next slot +1
             57       0FE2  2827                  [R0], R7                  ; to help round to word
        57   01   CA  0FE3  2828          BICL    #1, R7                   ; make it a word boundary.
             5D   D7  0FE6  2829          DECL    FP                       ; Loop on all slots
             2D   13  0FE8  2830          BEQL    SLOT_LOOP_OUT            ; Done all input slots, now process
                      0FEA  2831                                           ;  any output data
           FEBC   31  0FEA  2832          BRW     SLOT_LOOP_IN             ; Process all received slots
                      0FED  2833
                      0FED  2834 ECHO_ERROR:                               ; Character being echoed???
                      0FED  2835          DEBUG                            ;;&& Should not get to here
```

```
                    0FF3  2837          .SBTTL  BUILD A STOP SLOT
                    0FF3  2838 ;++
                    0FF3  2839 ; BUILD A STOP SLOT
                    0FF3  2840 ;
                    0FF3  2841 ; Functional description:
                    0FF3  2842 ;
                    0FF3  2843 ; Build a stop slot in the out message buffer, then delete the UCB.
                    0FF3  2844
                    0FF3  2845 ; Inputs:
                    0FF3  2846 ;        R5  = UCB address
                    0FF3  2847 ;        R8  = CSB address
                    0FF3  2848 ;        R9  = current transmit buffer slot address
                    0FF3  2849 ;        R10 = transmit buffer bytes left
                    0FF3  2850 ;
                    0FF3  2851 ; Outputs:
                    0FF3  2852 ;        R5  = UCB address
                    0FF3  2853 ;        R8  = CSB address
                    0FF3  2854 ;        R9  = current transmit buffer slot address
                    0FF3  2855 ;        R10 = transmit buffer bytes left
                    0FF3  2856 ;
                    0FF3  2857 ;--
                    0FF3  2858
                    0FF3  2859 SEND_STOP_SLOT:
        014B C5  95 0FF3  2860          TSTB    UCB$B_LT_CURC(R5)       ; Is UCB data waiting?
           03    13 0FF7  2861          BEQL    30$                     ; Br if no, send the STOP slot
         00B2    31 0FF9  2862          BRW     CREDIT_CHECK            ; Send data if our credit is good
                    0FFC  2863
        04   5A  D1 0FFC  2864 30$:     CMPL    R10, #SLT_T_DATA        ; Enough room for STOP slot?
           1E    15 0FFF  2865          BLEQ    ONE_LESS                ; Br if no, send it next time
        5D A8   96 1001  2866          INCB    CSB_B_NUM_SLOTS(R8)     ; Update slot count
                   1004  2867
                   1004  2868          ASSUME  SLT_B_DSTID EQ 0
    89  0140 C5  90 1004  2869          MOVB    UCB$B_LT_REMID(R5), (R9)+ ; Send the remote slot index
                   1009  2870
                   1009  2871          ASSUME  SLT_B_SRCID EQ SLT_B_DSTID+1
                   1009  2872          ASSUME  SLT_B_COUNT EQ SLT_B_SRCID+1
           89  B4 1009  2873          CLRW    (R9)+                   ; Zero SRCID for a stop slot
                   100B  2874                                          ;   and no data
                   100B  2875          ASSUME  SLT_B_CRED EQ SLT_B_COUNT+1
    89   D0 8F  90 100B  2876          MOVB    #SLT_C_STP_SLOT@4,(R9)+ ; Load stop slot format & no credits
        5A  04  C2 100F  2877          SUBL    #SLT_T_DATA,R10         ; Adjust bytes left for other slots
                   1012  2878                                          ;;&&;;** Skip call if HOST BIND
         0688  30 1012  2879          BSBW    LT$KILLUCB              ; Wipe out the UCB now.
           08  11 1015  2880          BRB     ONE_LESS                ; Get next slot
```

```
                        1017  2882          .SBTTL  PROCESS OUTPUT SLOT DATA
                        1017  2883 ;++
                        1017  2884 ; PROCESS OUTPUT SLOT DATA
                        1017  2885 ;
                        1017  2886 ; Functional description:
                        1017  2887 ;
                        1017  2888 ; Now output any waiting data from each UCB on this circuit and any
                        1017  2889 ; data returned from the class driver is transfered to the appropriate
                        1017  2890 ; output slot in the transmit buffer.
                        1017  2891 ;
                        1017  2892 ; Inputs:
                        1017  2893 ;        R0-R4 available
                        1017  2894 ;        R5  = UCB address
                        1017  2895 ;        R6,R7 =  scratch
                        1017  2896 ;        R8  = CSB address
                        1017  2897 ;        R9  = current transmit buffer slot address
                        1017  2898 ;        R10 = transmit buffer bytes left
                        1017  2899 ;        R11-FP = scratch
                        1017  2900 ;
                        1017  2901 ; Outputs:
                        1017  2902 ;        none.
                        1017  2903 ;
                        1017  2904 ; We continue processing in the send reply message section.
                        1017  2905 ;
                        1017  2906 ;--
                        1017  2907 SLOT_LOOP_OUT:
5D    2D A8    9A       1017  2908          MOVZBL  CSB_B_REFC(R8),FP       ; Set output UCB loop count
57    6C A8    DE       101B  2909          MOVAL   CSB_L_UCBLST(R8),R7     ; Set base address of UCB vector
                        101F  2910 ONE_LESS:                                ; One less unit left
         5D    D7       101F  2911          DECL    FP                      ; Any UCBs left ?
         03    18       1021  2912          BGEQ    10$                     ; Br if yes
       01E4    31       1023  2913          BRW     LOOP_EXIT               ; Else, exit loop
                        1026  2914 10$:
   55    87    D0       1026  2915          MOVL    (R7)+,R5                ; Get next UCB address
         FB    13       1029  2916          BEQL    10$                     ; ignore holes in non-dense vector
                        102B  2917          ;
                        102B  2918          ;  Dispatch on our terminal UCB state
                        102B  2919          ;
                        102B  2920          $DISPATCH   UCB$B_LT_STATE(R5),TYPE=B,-
                        102B  2921          <-      ; state         ; action
                        102B  2922          <UCB$C_LT_STATE_STOP    SEND_STOP_SLOT>,- ; STOP state, send stop slot
                        102B  2923          <UCB$C_LT_STATE_KILL    ONE_LESS>,-     ; KILL state, ignore this
                        102B  2924          >
                        1035  2925                                          ; RUN and START, just fall through
                        1035  2926
   0142 C5    95        1035  2927          TSTB    UCB$B_LT_DATAW(R5)      ; Is there any data waiting?
         E4    13       1039  2928          BEQL    ONE_LESS                ; Br if no, skip this UCB
                        103B  2929 ;
                        103B  2930 ;  Check for ABORT message first
                        103B  2931 ;
         02    E5       103B  2932          BBCC    #UCB$V_LT_ABORT,-       ; Br if no abort requested
1A 0142 C5              103D  2933                   UCB$B_LT_DATAW(R5),30$ ;   and clear flag
                        1041  2934          ;
                        1041  2935          ;  Send an ATTention slot
                        1041  2936          ;
                        1041  2937          ASSUME  SLT_S_ATT_SLOT&1 EQ 1   ; Assume we need rounding
   06    5A    D1       1041  2938          CMPL    R10, #SLT_S_ATT_SLOT+1  ; Enough room for ATTENTION slot ?
```

```
                30   15  1044  2939          BLEQ    50$                       ; Br if no, skip it (use br helper)
             5D A8   96  1046  2940          INCB    CSB_B_NUM_SLOTS(R8)       ; Update slot count
                         1049  2941
                         1049  2942          ASSUME  UCB$B_LT_LOCID EQ UCB$B_LT_REMID+1
                         1049  2943          ASSUME  SLT_B_DSTID EQ 0
                         1049  2944          ASSUME  SLT_B_SRCID EQ SLT_B_DSTID+1
                         1049  2945          ASSUME  SLT_B_COUNT EQ SLT_B_SRCID+1
                         1049  2946          ASSUME  SLT_B_TYPE EQ SLT_B_COUNT+1
                         1049  2947          ASSUME  SLT_B_ATT_CONTROL EQ SLT_B_TYPE+1
                         1049  2948
89     0140 C5   B0      1049  2949          MOVW    UCB$B_LT_REMID(R5), (R9)+ ; Send the remote slot id
       89   01   90      104E  2950          MOVB    #SLT_S_ATT_LEN,(R9)+      ; Set ATTENTION slot data size
  89     B0 8F   90      1051  2951          MOVB    #SLT_C_ATT_SLOT@4,(R9)+   ; Load ATTn type & no credits
                         1055  2952
                         1055  2953          ASSUME  SLT_S_ATT_SLOT&1 EQ 1     ; Assume we need rounding
       89   20   B0      1055  2954          MOVW    #SLT_ATT_M_ABORT,(R9)+    ; Send abort flag & zero byte
       5A   06   C2      1058  2955          SUBL    #SLT_S_ATT_SLOT+1,R10     ; Account from room taken
                         105B  2956 ;
                         105B  2957 ;  Check for flow control message
                         105B  2958 ;
                         105B  2959 30$:
                01   E5  105B  2960          BBLC    #UCB$V_LT_FLOW,-          ; Br if no flow requested
  4D 0142 C5             105D  2961                   UCB$B_LT_DATAW(R5),90$ ;   and clear flag
                         1061  2962          ;
                         1061  2963          ;  Send DATA_B slot
                         1061  2964          ;
       0146 C5   97      1061  2965          DECB    UCB$B_LT_TCRED(R5)        ; Use up a credit
                0C   18  1065  2966          BGEQ    40$                       ; Br if okay, continue
       0146 C5   96      1067  2967          INCB    UCB$B_LT_TCRED(R5)        ; Else, restore credits
                         106B  2968          SETBIT  #UCB$V_LT_FLOW,-          ; Remember to send flow control
                         106B  2969                   UCB$B_LT_DATAW(R5)       ;  when we have good credit
                3B   11  1071  2970          BRB     90$                       ; And continue
                         1073  2971
                         1073  2972 40$:
                         1073  2973          ASSUME  SLT_S_DB_SLOT&1 EQ 1      ; Assume we need rounding
       0A       5A   D1  1073  2974          CMPL    R10,-                     ; Enough room for DATA_B slot ?
                         1076  2975                   #<SLT_S_DB_SLOT+1>       ;     rounded up
                36   15  1076  2976 50$:     BLEQ    90$                       ; Br if no, skip it (branch helper)
             5D A8   96  1078  2977          INCB    CSB_B_NUM_SLOTS(R8)       ; Update slot count
                         107B  2978
                         107B  2979          ASSUME  UCB$B_LT_LOCID EQ UCB$B_LT_REMID+1
                         107B  2980          ASSUME  SLT_B_DSTID EQ 0
                         107B  2981          ASSUME  SLT_B_SRCID EQ SLT_B_DSTID+1
                         107B  2982          ASSUME  SLT_B_COUNT EQ SLT_B_SRCID+1
                         107B  2983          ASSUME  SLT_B_TYPE EQ SLT_B_COUNT+1
                         107B  2984          ASSUME  SLT_B_DB_CONTROL EQ SLT_B_TYPE+1
                         107B  2985          ASSUME  SLT_B_DB_OFOF EQ SLT_B_DB_CONTROL+1
                         107B  2986          ASSUME  SLT_B_DB_OFON EQ SLT_B_DB_OFOF+1
                         107B  2987          ASSUME  SLT_B_DB_IFOF EQ SLT_B_DB_OFON+1
                         107B  2988          ASSUME  SLT_B_DB_IFON EQ SLT_B_DB_IFOF+1
                         107B  2989
89     0140 C5   B0      107B  2990          MOVW    UCB$B_LT_REMID(R5), (R9)+ ; Send the remote slot id
       89   05   90      1080  2991          MOVB    #SLT_S_DB_LEN,(R9)+       ; Set DATA_B data size
  89     A0 8F   90      1083  2992          MOVB    #SLT_C_DB_SLOT@4,(R9)+    ; Load DATA_B type & no credits
       89   05   90      1087  2993          MOVB    #SLT_DB_M_IFENA!SLT_DB_M_OFENA,(R9)+ ; Assume default values
89   11131113 8F   D0    108A  2994          MOVL    #<17@245!<19@16>!<17@85!<19@0>,(R9)+ ; Assume default values
                         1091  2995
```

```
44 A5   10   B3  1091  2996          BITW    #TT$M_HOSTSYNC,UCB$L_DEVDEPEND(R5) ; Assumed ^S and ^Q
        04   12  1095  2997          BNEQ    60$                                ; Br if ok
FE A9   07   B0  1097  2998          MOVW    #<7@0>,-2(R9)                      ; Else, we were wrong
                 109B  2999
                 109B  3000 60$:
44 A5   20   B3  109B  3001          BITW    #TT$M_TTSYNC,UCB$L_DEVDEPEND(R5) ; Assumed ^S and ^Q
        08   12  109F  3002          BNEQ    70$                                ; Br if assumption was correct
FB A9   04   8A  10A1  3003          BICB    #SLT_DB_M_OFENA,-5(R9)             ; Else, clear output flow enable
F9 A9   08   88  10A5  3004          BISB    #SLT_DB_M_OFDIS,-5(R9)             ; and set output flow disable
                 10A9  3005
                 10A9  3006 70$:
                 10A9  3007          ASSUME  SLT_S_DB_SLOT&1 EQ 1               ; Assume we need rounding
        89   94  10A9  3008          CLRB    (R9)+                              ; Round up
   5A   0A   C2  10AB  3009          SUBL    #SLT_S_DB_SLOT+1,R10               ; Adjust bytes left (rounded up)
                 10AE  3010
                 10AE  3011 ;
                 10AE  3012 ;  Check if there is any output data waiting
                 10AE  3013 ;
                 10AE  3014 90$:
                 10AE  3015
```

```
                    10AE  3017 ;++
                    10AE  3018 ;
                    10AE  3019 ;  Any credits and enough room in output buffer for this slot ?
                    10AE  3020 ;
                    10AE  3021 ;--
                    10AE  3022 CREDIT_CHECK:                          ; UNUSED LABEL (breaks up LSB)
                    10AE  3023 ;
                    10AE  3024 ;  Flush all characters to transmit buffer
                    10AE  3025 ;
                    10AE  3026 ;        R8 = CSB address
                    10AE  3027 ;        R5 = UCB address
                    10AE  3028 ;
56   0149 C5   9A   10AE  3029          MOVZBL  UCB$B_LT_SLOTSZ(R5),R6  ; Get maximum slot size
     56   5A   D1   10B3  3030          CMPL    R10,R6                  ; Is space left greater?
          03   14   10B6  3031          BGTR    10$                     ; Br if yes, then we can use it
     56   5A   D0   10B8  3032          MOVL    R10,R6                  ; Else, use lesser size
     56   04   C2   10BB  3033 10$:     SUBL    #SLT_T_DATA, R6         ; Subtract slot header size
          08   14   10BE  3034          BGTR    30$                     ; Br if enough room
                    10C0  3035 20$:     SETBIT  #FLAG_V_RRF,-           ; Else, indicate response requested
                    10C0  3036                   CSB_B_FLAG(R8)         ;  since not all data fit in buffer
        FF57   31   10C5  3037          BRW     ONE_LESS                ; No room left in output slot
     5C   56   D0   10C8  3038 30$:     MOVL    R6,AP                   ; Get room left in output slot
  5B   04 A9   9E   10CB  3039          MOVAB   SLT_T_DATA(R9),R11      ; Sink of characters
                    10CF  3040          ;
                    10CF  3041          ;  Check if we have credits, or if we need to return credits
                    10CF  3042          ;  before building the message.
                    10CF  3043          ;
     0146 C5   95   10CF  3044          TSTB    UCB$B_LT_TCRED(R5)      ; Any credits we can use to send msg?
          09   14   10D3  3045          BGTR    40$                     ; Br if yes, okay to build data message
                    10D5  3046          ASSUME  LAT$C_MAX_RCRED EQ 1
     0147 C5   95   10D5  3047          TSTB    UCB$B_LT_XCRED(R5)      ; Remote have any credits?
          E5   12   10D9  3048          BNEQ    20$                     ; Br if yes, just leave now
        00C4   31   10DB  3049          BRW     USE_CREDIT              ; Else, send back credits
                    10DE  3050 40$:     ;
                    10DE  3051          ;  Any data from startio ?
                    10DE  3052          ;
52   014B C5   9A   10DE  3053          MOVZBL  UCB$B_LT_CURC(R5),R2    ; Get characters in startio ?
          4A   13   10E3  3054          BEQL    70$                     ; Br if none
53   014C C5   9E   10E5  3055          MOVAB   UCB$B_LT_CBUF(R5),R3    ; Set address of buffer
     52   5C   D1   10EA  3056          CMPL    AP,R2                   ; Got enough room in xmit buffer?
          29   18   10ED  3057          BGEQ    60$                     ; Br if yes
                    10EF  3058 ;
                    10EF  3059 ;        R2 = COUNT
                    10EF  3060 ;        R3 -> DATA
                    10EF  3061 ;
014B C5   5C   82   10EF  3062          SUBB2   AP, UCB$B_LT_CURC(R5)   ; Save residual count
          55   DD   10F4  3063          PUSHL   R5                      ; Save the UCB address
6B   63   5C   28   10F6  3064          MOVC3   AP,(R3),(R11)           ; Copy the string from UCB to output bfr
          55 8ED0   10FA  3065          POPL    R5                      ; Restore the UCB address
     5B   53   D0   10FD  3066          MOVL    R3,R11                  ; Set address of next available byte
52   014C C5   9E   1100  3067          MOVAB   UCB$B_LT_CBUF(R5), R2   ; Copy residual chars to beginning
53   014B C5   9A   1105  3068          MOVZBL  UCB$B_LT_CURC(R5), R3   ;  of UCB buffer area
          55   DD   110A  3069          PUSHL   R5                      ; Save the UCB address
62   61   53   28   110C  3070          MOVC3   R3,(R1),(R2)            ; Move the string up in the UCB
          55 8ED0   1110  3071          POPL    R5                      ; Restore the UCB address
          5C   D4   1113  3072          CLRL    AP                      ; No more room left
        008A   31   1115  3073          BRW     USE_CREDIT              ; Use a credit and send them
```

```
                        1118  3074
     014B C5    94      1118  3075 60$:    CLRB    UCB$B_LT_CURC(R5)        ; No data now
     5C   52    C2      111C  3076          SUBL2   R2,AP                    ; Adjust room left in slot
          55    DD      111F  3077          PUSHL   R5                       ; Save the UCB address
6B   63   52    28      1121  3078          MOVC3   R2,(R3),(R11)            ; Copy the string
          55 8ED0         1125  3079          POPL    R5                       ; Restore the UCB address
     5B   53    D0      1128  3080          MOVL    R3,R11                   ; Set address of next available byte
          5C    D5      112B  3081          TSTL    AP                       ; Any room left in slot ?
          73    15      112D  3082          BLEQ    USE_CREDIT               ; No, just transmit data
                        112F  3083 ;
                        112F  3084 ;  Check if there is any data from last time
                        112F  3085 ;
52   0120 C5    32      112F  3086 70$:     CVTWL   UCB$W_TT_OUTLEN(R5),R2   ; Any data left from last time ?
          2C    15      1134  3087          BLEQ    GET_MORE                 ; Br if no, try for new data
53   011C C5    D0      1136  3088          MOVL    UCB$L_TT_OUTADR(R5),R3   ; Get address of data
     52   5C    D1      113B  3089          CMPL    AP,R2                    ; Enough room in buffer ?
          03    18      113E  3090          BGEQ    80$                      ; Br if yes
     52   5C    D0      1140  3091          MOVL    AP, R2                   ; Use all the space we have left
0120 C5   52    A2      1143  3092 80$:     SUBW2   R2,UCB$W_TT_OUTLEN(R5)   ; Adjust descriptor
011C C5   52    C0      1148  3093          ADDL2   R2,UCB$L_TT_OUTADR(R5)   ; ...
     5C   52    C2      114D  3094          SUBL    R2, AP                   ; Adjust count of chars left
          55    DD      1150  3095          PUSHL   R5                       ; Save the UCB address
6B   63   52    28      1152  3096          MOVC3   R2,(R3),(R11)            ; Copy the string
          55 8ED0         1156  3097          POPL    R5                       ; Restore the UCB address
     5B   53    D0      1159  3098          MOVL    R3,R11                   ; Set address of next available byte
          5C    D5      115C  3099          TSTL    AP                       ; Any room left now?
          02    14      115E  3100          BGTR    GET_MORE                 ; Br if yes, get some more data
          40    11      1160  3101          BRB     USE_CREDIT               ; Else, try next slot
                        1162  3102
                        1162  3103 ;
                        1162  3104 ;  Loop draining class driver until no data is left or buffer is full
                        1162  3105 ;
                        1162  3106 GET_MORE:
     010C D5    16      1162  3107          JSB     @UCB$L_TT_GETNXT(R5)     ; Get characters from class driver
                        1166  3108 ;
                        1166  3109 ; R1-R4          modified
                        1166  3110 ; R3 = ??        and cc = 0               ! No more data
                        1166  3111 ; R3 = char      and cc = +               ! Cne character
                        1166  3112 ; R3 = ??        and cc = -               ! Burst data (TT_OUTADR, TT_OUTLEN)
                        1166  3113 ; R5 - ucb address
                        1166  3114 ;
          40    13      1166  3115          BEQL    USE_CREDIT1              ; No characters
          09    19      1168  3116          BLSS    GOT_A_LOT                ; Got more than one character
                        116A  3117
     8B   53    90      116A  3118          MOVB    R3,(R11)+                ; Copy character to output
          5C    D7      116D  3119          DECL    AP                       ; Adjust room in slot
          F1    14      116F  3120          BGTR    GET_MORE                 ; Go get more
          2F    11      1171  3121          BRB     USE_CREDIT               ; No room left in slot
                        1173  3122 GOT_A_LOT:
53   011C C5    D0      1173  3123          MOVL    UCB$L_TT_OUTADR(R5),R3   ; Get output buffer address
52   0120 C5    3C      1178  3124          MOVZWL  UCB$W_TT_OUTLEN(R5),R2   ;  and charcter count
     52   5C    D1      117D  3125          CMPL    AP,R2                    ; Got enough room in output buffer?
          03    18      1180  3126          BGEQ    10$                      ; Br if yes
     52   5C    D0      1182  3127          MOVL    AP,R2                    ; Output maximum possible
     5C   52    C2      1185  3128 10$:     SUBL2   R2,AP                    ; Adjust room left in slot
011C C5   52    C0      1188  3129          ADDL    R2,UCB$L_TT_OUTADR(R5)   ; Update output address
0120 C5   52    A2      118D  3130          SUBW2   R2,UCB$W_TT_OUTLEN(R5)   ;  and charcter count
```

```
                55   DD  1192  3131        PUSHL   R5                      ; Save the UCB address
        6B   63 52   28  1194  3132        MOVC3   R2,(R3),(R11)           ; Copy the string
                55 8ED0  1198  3133        POPL    R5                      ; restore the UCB address
             5B 53   D0  119B  3134        MOVL    R3,R11                  ; set address of next available byte
                5C   D5  119E  3135        TSTL    AP                      ; Any room left in slot ?
                C0   14  11A0  3136        BGTR    GET_MORE                ; Br if yes, look for more
```

```
                        11A2  3138          .SBTTL  CONSUME CREDIT WHEN FILLING IN SLOT FIELD
                        11A2  3139 ;
                        11A2  3140 ;  If any data was added to output slot, use credit locally, pass credit
                        11A2  3141 ;  to other end if needed, and set circuit mode to unbalanced.
                        11A2  3142 ;
                        11A2  3143          .ENABL  LSB
                        11A2  3144 USE_CREDIT:
   64 A5   02   A8      11A2  3145          BISW    #UCB$M_INT,UCB$L_STS(R5); Set interrupt expected, locks
                        11A6  3146                                          ;  out transmits from STARTIO
           0F   11      11A6  3147          BRB     10$                     ; Continue
                        11A8  3148
                        11A8  3149 USE_CREDIT1:
   64 A5   02   AA      11A8  3150          BICW    #UCB$M_INT,UCB$L_STS(R5); Clear interrupt expected
           02   E0      11AC  3151          BBS     #UCB$V_LT_OVFLOW,-      ; Don't clear data waiting flag
   05 0143 C5           11AE  3152                   UCB$B_LT_LATSTS(R5),10$; ;* we're waiting for OVFLOW to clear
           01   8A      11B2  3153          BICB    #UCB$M_LT_DATA,-        ; No more output data waiting
      0142 C5           11B4  3154                   UCB$B_LT_DATAW(R5)
                        11B7  3155 10$:
                        11B7  3156 ;;&&     SUBL3   AP,R6,AP                ; form character count in AP
                        11B7  3157 ;;&&     DECB    UCB$B_LT_TCRED(R5)      ; Use up a credit
                        11B7  3158 ;;&&     BISB    #FLAG_M_RRF,-           ; Set response requested flag
                        11B7  3159 ;;&&              CSB_B_FLAG(R8)         ;
           01   88      11B7  3160          BISB    #FLAG_M_RRF,-           ; Set response requested flag
        5C A8           11B9  3161                   CSB_B_FLAG(R8)         ;
5C      56   5C   C3    11BB  3162          SUBL3   AP,R6,AP                ; Calculate character count in AP
           0C   13      11BF  3163          BEQL    20$                     ; Br if no data, DON'T use credit
      0146 C5   97      11C1  3164          DECB    UCB$B_LT_TCRED(R5)      ; Else, use up a credit
                        11C5  3165          ASSUME  SLT_C_DA_SLOT EQ 0
      0147 C5   81      11C5  3166          ADDB3   UCB$B_LT_XCRED(R5),-    ; Calculate credits to send to remote
      50   01           11C9  3167                   #LAT$C_MAX_RCRED,R0     ;   system as DATA_A slot
           08   11      11CB  3168          BRB     40$                     ; Send credits and data
                        11CD  3169          ASSUME  SLT_C_DA_SLOT EQ 0
      0147 C5   81      11CD  3170 20$:     ADDB3   UCB$B_LT_XCRED(R5),-    ; See if we need to send
      50   01           11D1  3171                   #LAT$C_MAX_RCRED,R0     ;   a credit to the remote
           32   13      11D3  3172          BEQL    60$                     ; Br if no, send nothing
0148 C5    50   80      11D5  3173 40$:     ADDB    R0,UCB$B_LT_RCRED(R5)   ; Else, add in credits to send
0147 C5    50   82      11DA  3174          SUBB    R0,UCB$B_LT_XCRED(R5)   ; Credits have been extended
                        11DF  3175 ;
                        11DF  3176 ;        Setup slot header and advance pointer to next transmit slot header
                        11DF  3177 ;
                        11DF  3178          ASSUME  UCB$B_LT_LOCID EQ UCB$B_LT_REMID+1
                        11DF  3179          ASSUME  SLT_B_SRCID EQ SLT_B_DSTID+1
                        11DF  3180
      0140 C5   B0      11DF  3181          MOVW    UCB$B_LT_REMID(R5),-    ; Set up slot connection ids
           69           11E3  3182                   SLT_B_DSTID(R9)
        5D A8   96      11E4  3183          INCB    CSB_B_NUM_SLOTS(R8)     ; Update slot count
   02 A9   5C   90      11E7  3184          MOVB    AP,SLT_B_COUNT(R9)      ; Load character count
      0148 C5   90      11EB  3185          MOVB    UCB$B_LT_RCRED(R5),-    ; Load slot type and credits
        03 A9           11EF  3186                   SLT_B_CRED(R9)
                        11F1  3187          ASSUME  SLT_C_DA_SLOT EQ 0
      0148 C5   94      11F1  3188          CLRB    UCB$B_LT_RCRED(R5)      ; Set RUN state, no more credits
      50   59   D0      11F5  3189          MOVL    R9, R0                  ; Copy slot starting address
      59   5B   D0      11F8  3190          MOVL    R11, R9                 ; Address of start of next slot
           59   D6      11FB  3191          INCL    R9                      ; Make word address
      59   01   CA      11FD  3192          BICL    #1, R9                  ; which is even
50    59   50   C3      1200  3193          SUBL3   R0, R9, R0              ; Compute bytes used by slot
      5A   50   C2      1204  3194          SUBL2   R0, R10                 ; Adjust bytes left for other slots
```

```
                FE15   31  1207  3195 60$:    BRW     ONE_LESS                ; go do next UCB
                           120A  3196
                           120A  3197 LOOP_EXIT:
                           120A  3198 ;
                           120A  3199 ;        Compute and save the number of bytes in the transmit buffer.
                           120A  3200 ;
                           120A  3201 ;        R9 -> end of data in the transmit buffer
                           120A  3202 ;
                           120A  3203          ;
                           120A  3204          ;  There is a special case where the XMIT buffer can be taken,
                           120A  3205          ;  and that is when the circuit is halted. So if the circuit is
                           120A  3206          ;  in the HALT state, then we can assume we no longer have the
                           120A  3207          ;  XMIT buffer that we copied data into.
                           120A  3208          ;
                           120A  3209          $DISPATCH       CSB_B_STATE(R8),TYPE=B,- ; Dispatch on our CSB state
                           120A  3210          <-      ; state                 ; action
                           120A  3211                  <CSB_C_STATE_HALT       LT$REPLY_DONE>,- ; All done, exit
                           120A  3212          >
                           1211  3213
            5D A8      95  1211  3214          TSTB    CSB_B_NUM_SLOTS(R8)     ; Are there any slots?
               04      12  1214  3215          BNEQ    80$                     ; Br if yes, continue
               01      8A  1216  3216          BICB    #FLAG_M_RRF,-           ; Else, clear RRF flag
            5C A8          1218  3217                   CSB_B_FLAG(R8)         ;
        51  34 A8      D0  121A  3218 80$:     MOVL    CSB_Q_XBUFQ(R8), R1     ; Address of buffer
        1C A1          59  D0  121E  3219      MOVL    R9, CXB$L_T_ENDADR(R1)  ; Save end address of data
                           1222  3220          .DSABL  LSB
```

```
                1222  3222          .SBTTL  SEND REPLY RESPONSE TO RECEIVED MESSAGE
                1222  3223
                1222  3224 LT$REPLY_MSG:
                1222  3225 ;
                1222  3226 ;  We come here to transmit a new frame in response to a received frame
                1222  3227 ;
                1222  3228 ;        R8   =  CSB address
                1222  3229 ;
59   34 A8   D0 1222  3230          MOVL    CSB_Q_XBUFQ(R8),R9      ; Transmit buffer address
                1226  3231
                1226  3232 LT$REPLY_ALT:
                1226  3233          .IF DEFINED LT_XMT_CHECK
                1226  3234 ;
                1226  3235 ;        Debug code
                1226  3236 ;
                1226  3237 ;        check the format of the slots and make sure that they are
                1226  3238 ;        well formatted.
                1226  3239 ;
                1226  3240 ;        R8 = CSB address
                1226  3241 ;        R9 = transmit buffer address
                1226  3242 ;        R0-R3 clobbered
                1226  3243 ;
                1226  3244          MOVAQ   CSB_Q_XBUFQ(R8), R0             ; Address of transmit head
                1226  3245          CMPL    R9, R0                          ; Is there a transmit buffer
                1226  3246          BNEQ    5$                              ; Nope, its the queue head
                1226  3247          DEBUG                                   ; Can't be the queue head
                1226  3248
                1226  3249 5$:      MOVQ    CSB_T_CIRCHDR(R8),-             ; Move the circ header
                1226  3250                  XMT_T_DATA(R9)                  ; to look at real msg
                1226  3251          CMPB    XMT_B_FLAG(R9),-                ; Only analyse running msgs
                1226  3252                  #MTYP_C_RUN@FLAG_V_MTYPE        ;  with slots
                1226  3253          BNEQ    50$                             ; Not a running message
                1226  3254          MOVL    CXB$L_T_ENDADR(R9), R2
                1226  3255          MOVAB   XMT_T_MDATA(R9), R0             ; First slot address
                1226  3256          MOVZBL  XMT_B_NUM_SLOTS(R9), R1         ; number of valid slots
                1226  3257          BNEQ    10$                             ; We have some
                1226  3258          ;  No slot - message must only have a circuit header!
                1226  3259          CMPL    R2, R0                          ; Only circuit header?
                1226  3260          BEQL    50$                             ; Br if yes, okay
                1226  3261          DEBUG                                   ; Not correct
                1226  3262          BRB     50$
                1226  3263
                1226  3264 10$:     MOVZBL  SLT_B_COUNT(R0), R3             ; Look to next slot
                1226  3265          BGEQ    30$                             ; Must be greater or zero
                1226  3266          DEBUG                                   ; Maxc not valid
                1226  3267 30$:     ADDL    #SLT_T_DATA, R3                 ; Slot header length
                1226  3268          ADDL    R3, R0                          ;
                1226  3269          INCL    R0                              ; Round to word boundary
                1226  3270          BICL    #1, R0                          ;
                1226  3271          CMPL    R0, R2                          ; Still inside buffer?
                1226  3272          BLEQU   40$                             ; Yep
                1226  3273          DEBUG                                   ; Slot beyond buffer
                1226  3274 40$:     SOBGTR  R1, 10$                         ; Check all slots
                1226  3275 50$:
                1226  3276          .ENDC   ;; DEFINED LT_XMT_CHECK
                1226  3277
54   58   D0    1226  3278          MOVL    R8, R4                          ; CSB address to r4
```

```
   080A   30  1229  3279          BSBW    LT$XMIT                          ; Transmit the buffer
              122C  3280
              122C  3281 LT$REPLY_DONE:                                      ; All done
3FFE 8F   BA  122C  3282          POPR    #^M<R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,AP,FP> ; Restore registers
          05  1230  3283          RSB
              1231  3284
              1231  3285
              1231  3286 LT$REPLY_REXMIT:
              1231  3287 ;
              1231  3288 ;  Retransmit last message
              1231  3289 ;
              1231  3290 ;        R8 = CSB address
              1231  3291 ;
54    5C  D0  1231  3292          MOVL    R8, R4                   ; CSB address to R4
   0864   30  1234  3293          BSBW    LT$REXMIT                ; Transmit all waiting buffers
     F3   11  1237  3294          BRB     LT$REPLY_DONE            ; Finish up
              1239  3295
```

```
                                1239  3297          .SBTTL  LT$CREATECSB  - Create a CSB
                                1239  3298 ;++
                                1239  3299 ; LT$CREATECSB - Create a CSB
                                1239  3300 ;
                                1239  3301 ;       Allocate a CSB for use and initialize it partially.
                                1239  3302 ;
                                1239  3303 ; Inputs:
                                1239  3304 ;       R4 -> next entry in CSB table beyond the one that is free
                                1239  3305 ;       R1 = length of UCB list at end of CSB
                                1239  3306 ;
                                1239  3307 ; Outputs:
                                1239  3308 ;       R8 = new CSB address
                                1239  3309 ;       R0 = success or failure
                                1239  3310 ;
                                1239  3311 ;       CSB table entry set to its address
                                1239  3312 ;
                                1239  3313 ;       R1-R2 are destroyed.
                                1239  3314 ;--
                                1239  3315
                                1239  3316 LT$CREATECSB:
        51   00000070 8F   C0   1239  3317          ADDL    #CSB_C_FIXLENGTH+4,R1   ; Add in size of fixed block +
                                1240  3318                                          ;  one "no access" UCB slot
                     53   DD   1240  3319          PUSHL   R3                      ; Save R3
             00000000'GF   16   1242  3320          JSB     G^EXE$ALONONPAGED       ; Get block
                     53 8ED0   1248  3321          POPL    R3                      ; Restore R3
                  4D 50   E9   124B  3322          BLBC    R0, 10$                 ; Got block
                     3E   BB   124E  3323          PUSHR   #^M<R1,R2,R3,R4,R5>     ; Save registers
 62  51   00   6E   00   2C   1250  3324          MOVC5   #0,(SP),#0,R1,(R2)      ; Zero block
                     3E   BA   1256  3325          POPR    #^M<R1,R2,R3,R4,R5>     ; Restore registers
              08 A2  51   B0   1258  3326          MOVW    R1,CSB_W_SIZE(R2)       ; Save size of CSB in CSB
              0A A2  33   90   125C  3327          MOVB    #DYN$C_CDB, -           ; Store an unlikely type in the
                                1260  3328                  CSB_B_TYPE(R2)          ; the block
           50   EDD4 CF   DE   1260  3329          MOVAL   GHB_L_CSBTABLE, R0      ; Create the slave index from the
           50   54   50   C3   1265  3330          SUBL3   R0, R4, R0              ; address in the CSB table
                50   04   C6   1269  3331          DIVL2   #4, R0                  ; from 1 to n
              60 A2  50   90   126C  3332          MOVB    R0, CSB_B_LOCIDN(R2)    ; and save it in the CSB
              68 A2  50   90   1270  3333          MOVB    R0, CSB_B_INX(R2)       ; Save it here also
 61 A2   EE3F CF40   90   1274  3334          MOVB    GHB_B_LOC_CCHK[R0],-    ; Obtain the check field for this
                                127B  3335                  CSB_B_LOCIDS(R2)        ;     CSB slot
                                127B  3336          ;
                                127B  3337          ;  Initialize queue headers
                                127B  3338          ;
              50   34 A2   9E   127B  3339          MOVAB   CSB_Q_XBUFQ(R2), R0     ; Initialize the queue headers for the
                   60   50   D0   127F  3340          MOVL    R0, (R0)                ; transmit buffers
              04 A0   50   D0   1282  3341          MOVL    R0, 4(R0)               ;
              50   3C A2   9E   1286  3342          MOVAB   CSB_Q_XWAITQ(R2), R0    ;
                   60   50   D0   128A  3343          MOVL    R0, (R0)                ;
              04 A0   50   D0   128D  3344          MOVL    R0, 4(R0)               ;
                   58   52   D0   1291  3345          MOVL    R2, R8                  ; Return address of CSB here
                   74   52   D0   1294  3346          MOVL    R2, -(R4)               ; Store address of CSB in table
              50   00'8F   9A   1297  3347          MOVZBL  #SS$_NORMAL, R0
                        05   129B  3348 10$:    RSB
```

```
                        129C  3350          .SBTTL   LT$NEW_CIRCUIT - Get Concentrator State Block
                        129C  3351  ;++
                        129C  3352  ; LT$NEW_CIRCUIT - Get concentrator state block and initialize it
                        129C  3353  ;
                        129C  3354  ; Functional description:
                        129C  3355  ;
                        129C  3356  ; Whenever a frame with a zero svci is seen and the circuit state is
                        129C  3357  ; starting, we attempt to allocate a CSB and start a circuit.  First
                        129C  3358  ; we check for a CSB with the same source address.  If any are found
                        129C  3359  ; and the mvci matches and the state is starting, then we retransmit
                        129C  3360  ; our starting message for that circuit.  If the other things are not
                        129C  3361  ; true, then the circuit is clobbered to that source since the mvci
                        129C  3362  ; does not match or the circuit is already running and the concentrator
                        129C  3363  ; is trying to restart it perhaps because it rebooted.
                        129C  3364  ;
                        129C  3365  ; Inputs:
                        129C  3366  ;       R3 =  CXB address
                        129C  3367  ;       R6 =  receive buffer address
                        129C  3368  ;       R10 = IDB address
                        129C  3369  ;
                        129C  3370  ; Outputs:
                        129C  3371  ;       R6 =  receive buffer address
                        129C  3372  ;       R8 =  CSB address
                        129C  3373  ;       R0,R1,R2,R3,R4,R7 and R9 are modified
                        129C  3374  ;--
                        129C  3375
                        129C  3376 LT$NEW_CIRCUIT:
54  ED98 CF    9E       129C  3377          MOVAB   GHB_L_CSBTABLE,R4       ; Find free entry in CSB_TABLE
    52   20    9A       12A1  3378          MOVZBL  #LAT$C_MAX_CSBS,R2      ; Length of table  in longwords
    58   84    D0       12A4  3379 10$:     MOVL    (R4)+,R8                ; Get CSB address
         2A    13       12A7  3380          BEQL    30$                     ; Ignore unallocated ones for pass
                        12A9  3381  ;
                        12A9  3382  ;       If this is the same source address, then we are looking for a duplicate
                        12A9  3383  ;       starting message.  It will have the same local index and a state of
                        12A9  3384  ;       starting.  If this is not the case, then just clobber the circuit.
                        12A9  3385  ;
    28 A3    D1         12A9  3386          CMPL    CXB$Q_STATION(R3),-     ; NI source address match?
    2E A8               12AC  3387                  CSB_Z_DST(R8)           ;
       23    12         12AE  3388          BNEQ    30$                     ; Br if not, look for more
    2C A3    B1         12B0  3389          CMPW    CXB$Q_STATION+4(R3),-   ; Rest match too?
    32 A8               12B3  3390                  CSB_Z_DST+4(R8)         ;
       1C    12         12B5  3391          BNEQ    30$                     ; Br if not, look for more
    02 A6    B1         12B7  3392          CMPW    RCV_W_DSTID(R6),-       ; Is this a duplicate starting msg
    60 A8               12BA  3393                  CSB_W_LOCID(R8)         ;   with the same local vc indx/seq?
       0A    12         12BC  3394          BNEQ    20$                     ; Br if not, kill circuit
    0B A8    91         12BE  3395          CMPB    CSB_B_STATE(R8),-       ; Is the state still starting?
       01               12C1  3396                  #CSB_C_STATE_START      ;
       04    12         12C2  3397          BNEQ    20$                     ; Br if not, kill circuit
50     02    D0         12C4  3398          MOVL    #2, R0                  ; Else, tell higher code to retrans
             05         12C7  3399          RSB                             ;   our starting message
                        12C8  3400
     0426    30         12C8  3401 20$:     BSBW    LT$CIRCDEAD             ; Get rid of UCBs
2E A8  01    CE         12CB  3402          MNEGL   #1, CSB_Z_DST(R8)       ; Form illegal address
32 A8  01    AE         12CF  3403          MNEGW   #1, CSB_Z_DST+4(R8)     ;  to stop next compare
                        12D3  3404
    CE 52    F5         12D3  3405 30$:     SOBGTR  R2,10$                  ; Try whole table
                        12D6  3406          ;
```

```
                        12D6  3407          ;  Search failed, try to find a free CSB
                        12D6  3408          ;
            02    E0    12D6  3409          BBS     #GHB_STS_V_SHUT,-       ; Br if we are shutting down,
   0F EEA9 CF           12D8  3410                   GHB_B_STATUS,50$       ;   don't let connection succeed
54 ED58 CF    DE        12DC  3411          MOVAL   GHB_L_CSBTABLE, R4      ; Look again for a free CSB
      52    20    9A    12E1  3412          MOVZBL  #LAT$C_MAX_CSBS, R2     ; Set max CSBs
            84    D5    12E4  3413 40$:     TSTL    (R4)+                   ; Is this slot free?
            06    13    12E6  3414          BEQL    60$                     ; Br if yes, great
         F9 52    F5    12E8  3415          SOBGTR  R2, 40$                 ; Loop til done
                        12EB  3416          ;
                        12EB  3417          ;  Return resouce error
                        12EB  3418          ;
            50    D4    12EB  3419 50$:     CLRL    R0                      ; Return failure indicating resource
            05          12ED  3420          RSB                             ; failure
                        12EE  3421
                        12EE  3422 60$:     ;
                        12EE  3423          ;  Check the protocol version of the incoming message and reject all but
                        12EE  3424          ;  our own.
                        12EE  3425          ;
   50    08 A6    9E    12EE  3426          MOVAB   RCV_T_MDATA(R6), R0     ; Address of first slot
   05    02 A0    91    12F2  3427          CMPB    STRT_B_PVER(R0), -      ; Check version number
                        12F6  3428                  #LAT$C_CUR_VER
            F3    12    12F6  3429          BNEQ    50$                     ; Must be equal
                        12F8  3430          ;
                        12F8  3431          ;   Check the circuit timer value (in multiples of 10 ms).
                        12F8  3432          ;   We will accept anything in the range 10 to 1270 ms; legal
                        12F8  3433          ;   values are 1 to 127 in this byte field.
                        12F8  3434          ;
         06 A0    95    12F8  3435          TSTB    STRT_B_CIR_TIMR(R0)     ; Is this field positive and non-zero?
            EE    15    12FB  3436          BLEQ    50$                     ; Br if not, illegal value
                        12FD  3437 ;
                        12FD  3438 ;    ECO is not checked
                        12FD  3439 ;
                        12FD  3440 ;       R4 -> CSB table address -4
                        12FD  3441 ;
                        12FD  3442          ;
                        12FD  3443          ;  Compute size needed for UCB list at end of CSB,
                        12FD  3444          ;  size is based on MIN [LAT$C_MAX_SLOTS, STRT_B_MAXSLOTS(R0)]
                        12FD  3445          ;
   55    40 8F    9A    12FD  3446          MOVZBL  #LAT$C_MAX_SLOTS,R5     ; Assume ours is MIN value
   55    04 A0    91    1301  3447          CMPB    STRT_B_MAXSLOTS(R0),R5  ; Is rcvd slot count larger?
            04    1E    1305  3448          BGEQU   70$                     ; Br if yes, use ours
   55    04 A0    90    1307  3449          MOVB    STRT_B_MAXSLOTS(R0),R5  ; Else, use smaller of two
51    55    02    78    130B  3450 70$:     ASHL    #2,R5,R1                ; Multiply by 4, and save in R1
                        130F  3451
         FF27    30    130F  3452          BSBW    LT$CREATECSB            ; Create a CSB
         D6 50   E9    1312  3453          BLBC    R0, 50$                 ; Br if resource error
   69 A8    55    90    1315  3454          MOVB    R5,CSB_B_MAX_SLOTS(R8)  ; Save the maximum allowed slot count
                        1319  3455          ;
                        1319  3456          ;  Save the server's name
                        1319  3457          ;
                        1319  3458          ASSUME  STRT_B_SYS_LEN EQ STRT_T_NODE+GHB$K_NAMELEN
   50    08 A6    9E    1319  3459          MOVAB   RCV_T_MDATA(R6),R0      ; Get address of first slot
   51    0C A0    9A    131D  3460          MOVZBL  STRT_B_NODE_LEN(R0),R1  ; Get length of dest node name
   52    0D A0    9E    1321  3461          MOVAB   STRT_T_NODE(R0),R2      ; Get address of dest node name
      52    51    C0    1325  3462          ADDL    R1,R2                   ; Skip dest node name, point to src name
      51    62    9A    1328  3463          MOVZBL  (R2),R1                 ; Get length of src (server) name
```

```
                10  51  D1  132B  3464          CMPL    R1,#GHB$K_NAMELEN        ; Is the name length okay?
                    06  15  132E  3465          BLEQ    75$                      ; Br if yes, continue
                51  10  9A  1330  3466          MOVZBL  #GHB$K_NAMELEN,R1        ; Else, use maximum allowed
                62  51  90  1333  3467          MOVB    R1,(R2)                  ; Make size of name valid
                    51  D6  1336  3468  75$:    INCL    R1                       ; Account for length byte
                    09  BB  1338  3469          PUSHR   #^M<R0,R3>               ; Save registers
            1C A8   62  51  28  133A  3470      MOVC3   R1,(R2),CSB_B_SERVER(R8) ; Copy the server name
                    09  BA  133F  3471          POPR    #^M<R0,R3>               ; Restore registers
                            1341  3472          ;
                            1341  3473          ;  Make the circuit timeout value twice the keep_alive value
                            1341  3474          ;  received from the terminal server.
                            1341  3475          ;
            66 A8   0E10 8F     B0  1341  3476  MOVW    #3600, CSB_W_TIMRESET(R8); Assume a default of one hour
                51    07 A0     9A  1347  3477  MOVZBL  STRT_B_KPA_TIMR(R0), R1 ; Make timer reset value
                    11  13  134B  3478          BEQL    80$                      ; Not a good value
                51  05  C4  134D  3479          MULL2   #5,R1                    ; Look for five times the period,
                    51  D6  1350  3480          INCL    R1                       ;  and one more
                55  51  3C  1352  3481          MOVZWL  R1,R5                    ; Check for overflow
                55  51  D1  1355  3482          CMPL    R1,R5                    ; Did we overflow?
                    04  12  1358  3483          BNEQ    80$                      ; Br if yes, use default value
            66 A8   51  B0  135A  3484          MOVW    R1, CSB_W_TIMRESET(R8)   ; Set the value in the CSB
                            135E  3485  80$.    ;
                            135E  3486          ;  Compute maximum buffer size to use
                            135E  3487          ;
                05D4 8F     B0  135E  3488      MOVW    #XMT_C_MAXDATA,-         ; Assume ours is maximum allowed
                  6A A8         1362  3489                 CSB_W_MAX_MSGSIZ(R8)  ;    message size
                    08  A2  1364  3490          SUBW    #XMT_C_HDRLEN,-          ; Account for header is remotes size
                    60          1366  3491                 STRT_W_MSGSIZ(R0)
                    0A  15  1367  3492          BLEQ    90$                      ; Br if not large enough
                    60  B1  1369  3493          CMPW    STRT_W_MSGSIZ(R0),-      ; Is remotes buffer larger?
                  6A A8         136B  3494                 CSB_W_MAX_MSGSIZ(R8)
                    04  1E  136D  3495          BGEQU   90$                      ; Br if yes, use ours
                    60  B0  136F  3496          MOVW    STRT_W_MSGSIZ(R0),-      ; Else, use smaller of two
                  6A A8         1371  3497                 CSB_W_MAX_MSGSIZ(R8)
                            1373  3498  90$:    ;
                            1373  3499          ;  Note that locidn and xmit queue headers were setup by CREATECSB
                            1373  3500          ;
          5E A8   04 A6     B0  1373  3501      MOVW    RCV_W_SRCID(R6),CSB_W_REMID(R8) ; In case of retransmit
                  5D A8     94  1378  3502      CLRB    CSB_B_NUM_SLOTS(R8)      ; Mode default is balanced
                  52 A8     B4  137B  3503      CLRW    CSB_W_XMTTMO(R8)         ; Not timed state
                  66 A8     B0  137E  3504      MOVW    CSB_W_TIMRESET(R8),-     ; Set a nonzero timeout value
                  50 A8         1381  3505                 CSB_Q_TIMEOUT(R8)     ; even if the normal value is zero.
                  2D A8     94  1383  3506      CLRB    CSB_B_REFC(R8)           ; No UCBs here
                     14     B0  1386  3507      MOVW    #LAT$C_PROGRESS,-        ; Set for allowing progress
                  54 A8         1388  3508                 CSB_W_PROGRESS(R8)    ; while we start up.
                            138A  3509
                  28 A3     D0  138A  3510      MOVL    CXB$Q_STATION(R3),-      ; Store remotes's NI address.
                  2E A8         138D  3511                 CSB_Z_DST(R8)
                  2C A3     B0  138F  3512      MOVW    CXB$Q_STATION+4(R3),-    ;        ...
                  32 A8         1392  3513                 CSB_Z_DST+4(R8)
                     00     90  1394  3514      MOVB    #CSB_C_STATE_HALT,-      ; Circuit is halted
                  0B A8         1396  3515                 CSB_B_STATE(R8)
              62 A8   01     AE  1398  3516      MNEGW   #1,CSB_W_XSEQ(R8)        ; Initialize transmit sequence
        64 A8   FF00 8F     B0  139C  3517      MOVW    #<-1a8105>,CSB_W_RSEQ(R8); Initialize receive sequence
                            13A2  3518          ;
                            13A2  3519          ;  Obtain all necessary transmit buffers for use
                            13A2  3520          ;
```

```
          58 A8   94  13A2  3521          CLRB    CSB_B_XMTBSY(R8)        ; Not busy transmitting
          59 A8   94  13A5  3522          CLRB    CSB_B_XMTCNT(R8)        ; No buffers waiting to transmit
    54    34 A8   DE  13A8  3523          MOVAL   CSB_Q_XBUFQ(R8), R4     ; Address of the transmit queue head
       55    02   9A  13AC  3524          MOVZBL  #LAT$C_XMT_BUFFERS, R5  ; Set number of buffers
                    13AF  3525
51    0624 8F   3C  13AF  3526 100$:     MOVZWL  #XMT_C_MAXLEN, R1       ; Size of the buffers
 00000000'GF   16  13B4  3527          JSB     G^EXE$ALONONPAGED       ; Allocate the buffer
          25 50   E9  13BA  3528          BLBC    R0, 110$                ; No such buffer
    08 A2    51   B0  13BD  3529          MOVW    R1, CXB$W_SIZE(R2)      ; Store the size,
                    13C1  3530          ASSUME  CXB$B_CODE EQ CXB$B_TYPE+1
    0A A2    1B   B0  13C1  3531          MOVW    #DYN$C_CXB, -           ; the block type
                    13C5  3532                  CXB$B_TYPE(R2)          ; and indicate not multicast buffer
                    13C5  3533
    0C A2    58   D0  13C5  3534          MOVL    R8,CXB$L_T_CSB(R2)      ; Save CSB address in CXB
          2E A8   7D  13C9  3535          MOVQ    CSB_Z_DST(R8),-         ; Initialize the destination address
          28 A2       13CC  3536                  CXB$Q_STATION(R2)
          48 A2   7C  13CE  3537          CLRQ    XMT_T_DATA(R2)          ; Zero circuit header
       0048 8F   B0  13D1  3538          MOVW    #XMT_T_DATA,-           ; Store offset to start of data
          18 A2       13D5  3539                  CXB$Q_BOFF(R2)
       64    62   0E  13D7  3540          INSQUE  (R2), (R4)              ; Queue the buffer
          D2 55   F5  13DA  3541          SOBGTR  R5, 100$                ; for each of the buffers
                    13DD  3542
    50    00'8F   9A  13DD  3543          MOVZBL  #SS$_NORMAL, R0
                 05  13E1  3544          RSB
                    13E2  3545
                    13E2  3546          ;
                    13E2  3547          ;   Resource error return
                    13E2  3548          ;
       54    58   D0  13E2  3549 110$:   MOVL    R8, R4                  ; Move the CSB address for the call
             0003 30  13E5  3550          BSBW    LT$DEALOCSB             ; Clean up the CSB before we leave
                50   D4  13E8  3551          CLRL    R0                      ; Return an error
                 05  13EA  3552          RSB
                    13EB  3553
```

```
                        13EB  3555          .SBTTL  LT$DEALOCSB  - Deallocate a CSB
                        13EB  3556 ;++
                        13EB  3557 ; LT$DEALOCSB
                        13EB  3558 ;
                        13EB  3559 ; FUNCTIONAL DESCRIPTION:
                        13EB  3560 ;
                        13EB  3561 ;        Deallocate the transmit buffers.
                        13EB  3562 ;
                        13EB  3563 ; INPUTS:
                        13EB  3564 ;        R4 = CSB address
                        13EB  3565 ;
                        13EB  3566 ; OUTPUTS:
                        13EB  3567 ;        R0-R3 are destroyed
                        13EB  3568 ;
                        13EB  3569 ;--
                        13EB  3570
                        13EB  3571 LT$DEALOCSB:
        51    01    9A  13EB  3572          MOVZBL  #LAT$C_XMT_BUFFERS-1,R1 ; Get number of transmits -1
50      44 A441    D0  13EE  3573 10$:     MOVL    CSB_L_XCXB(R4)[R1], R0  ; Is an FFI transmit in progress ?
              03    13  13F3  3574          BEQL    20$                     ; no
           0C A0    D4  13F5  3575          CLRL    CXB$L_T_CSB(R0)         ; Delete backpointer to CSB in CXB
           F3 51    F4  13F8  3576 20$:     SOBGEQ  R1,10$                  ; Loop if more
50         34 B4    0F  13FB  3577 25$:     REMQUE  @CSB_Q_XBUFQ(R4), R0    ; Now dump all the buffers
              05    1D  13FF  3578          BVS     30$                     ; No more buffers
            079C    30  1401  3579          BSBW    LT$DEALPOOL             ; Dump a buffer
              F5    11  1404  3580          BRB     25$                     ;
                        1406  3581
50         3C B4    0F  1406  3582 30$:     REMQUE  @CSB_Q_XWAITQ(R4), R0   ; Now dump all the waiters
              05    1D  140A  3583          BVS     40$                     ; No more buffers
            0791    30  140C  3584          BSBW    LT$DEALPOOL             ; Dump a buffer
              F5    11  140F  3585          BRB     30$                     ;
                        1411  3586
53         68 A4    9A  1411  3587 40$:     MOVZBL  CSB_B_INX(R4), R3       ; Find our place in the table
              06    12  1415  3588          BNEQ    50$                     ; Great
                        1417  3589          DEBUG                           ; Not properly initialized CSB
                        141D  3590
         EC96 CF43    96  141D  3591 50$:     INCB    GHB_B_LOC_CCHK[R3]      ; Bump the slot check digit
50       ECOD CF43    DE  1422  3592          MOVAL   GHB_L_CSBTABLE-4[R3],R0 ; Address in table
           54    60    D1  1428  3593          CMPL    (R0), R4                ; Check this address with the one
              06    13  142B  3594          BEQL    60$                     ; we have.
                        142D  3595          DEBUG                           ; Table is clobbered or CSB addr bad
                        1433  3596
              60    D4  1433  3597 60$:     CLRL    (R0)                    ; No CSB address here anymore
                        1435  3598          ;
                        1435  3599          ;  We will now save this CSB on the OLD CSB queue, but we will
                        1435  3600          ;  have to remove an old entry if we exceed our limit.
                        1435  3601          ;
           50    54    D0  1435  3602          MOVL    R4,R0                   ; Copy CSB address just in case
                        1438  3603                                          ;  we want to deallocate it
           0C A4    D5  1438  3604          TSTL    CSB_Z_LCB+LCB_L_MSG_XMT(R4) ; Any information in counters?
              08    12  143B  3605          BNEQ    70$                     ; Br if yes, save it
           10 A4    D5  143D  3606          TSTL    CSB_Z_LCB+LCB_L_MSG_RCV(R4) ; Any information in counters?
              03    12  1440  3607          BNEQ    70$                     ; Br if yes, save it
            008C    31  1442  3608          BRW     180$                    ; Else, nothing interesting here
                        1445  3609 70$:     ;
                        1445  3610          ;  We will now scan the CSB list for any old CSBs that match this
                        1445  3611          ;  one, that way we will only save 1 entry per server.
```

```
                        1445  3612          ;
51   ECE7 CF    9E      1445  3613          MOVAB   GHB$Q_OLD_CSBS+GHB_AREA,R1 ; Get listhead of old CSBs
     52   51    D0      144A  3614          MOVL    R1,R2                      ; ..twice
     52   62    D0      144D  3615 80$:     MOVL    (R2),R2                    ; Get next in queue
     51   52    D1      1450  3616          CMPL    R2,R1                      ; Are we back to start?
          68    13      1453  3617          BEQL    100$                       ; Br if yes, no more - insert new entry
        2E A4   D1      1455  3618          CMPL    CSB_Z_DST(R4),-            ; Does the Ethernet address match?
        2E A2           1458  3619                  CSB_Z_DST(R2)
          F1    12      145A  3620          BNEQ    80$                        ; Br if no, try next
        32 A4   B1      145C  3621          CMPW    CSB_Z_DST+4(R4),-          ; Full address match?
        32 A2           145F  3622                  CSB_Z_DST+4(R2)
          EA    12      1461  3623          BNEQ    80$                        ; Br if no, try next
53    1C A4     9A      1463  3624          MOVZBL  CSB_B_SERVER(R4),R3        ; Get length of server name
1C A2   53      91      1467  3625          CMPB    R3,CSB_B_SERVER(R2)        ; Lengths match?
          E0    12      146B  3626          BNEQ    80$                        ; Br if no, try next
          07    BB      146D  3627          PUSHR   #^M<R0,R1,R2>              ; Save registers
1D A4   53      29      146F  3628          CMPC3   R3,CSB_T_SERVER(R4),-      ; Does the server name match?
      1D A2             1473  3629                  CSB_T_SERVER(R2)
          07    BA      1475  3630          POPR    #^M<R0,R1,R2>
          D4    12      1477  3631          BNEQ    80$                        ; Br if no, try next
                        1479  3632          ;
                        1479  3633          ;    Take sum of counters and store in CSB @ R2
                        1479  3634          ;
                        1479  3635          ADD_CTR CSB_Z_LCB+LCB_L_MSG_XMT(R4),- ; XMITs
                        1479  3636                  CSB_Z_LCB+LCB_L_MSG_XMT(R2)
                        1484  3637          ADD_CTR CSB_Z_LCB+LCB_L_MSG_RCV(R4),- ; RCVs
                        1484  3638                  CSB_Z_LCB+LCB_L_MSG_RCV(R2)
                        148F  3639          ADD_CTR CSB_Z_LCB+LCB_L_MSG_REXMT(R4),- ; REXMTs
                        148F  3640                  CSB_Z_LCB+LCB_L_MSG_REXMT(R2)
                        149A  3641          ADD_CTR CSB_Z_LCB+LCB_W_SEQ_ERR(R4),- ; SEQ_ERRs
                        149A  3642                  CSB_Z_LCB+LCB_W_SEQ_ERR(R2),W
                        14A5  3643          ADD_CTR CSB_Z_LCB+LCB_B_INV_MSG(R4),- ; INV_MSGs
                        14A5  3644                  CSB_Z_LCB+LCB_B_INV_MSG(R2),B
                        14B0  3645          ADD_CTR CSB_Z_LCB+LCB_B_INV_SLOT(R4),- ; INV_SLOTs
                        14B0  3646                  CSB_Z_LCB+LCB_B_INV_SLOT(R2),B
          14    11      14BB  3647          BRB     180$                       ; And delete the CSB @ R4
                        14BD  3648 100$:    ;
                        14BD  3649          ;  Insert new CSB into list
                        14BD  3650          ;
EC72 DF   64    0E      14BD  3651          INSQUE  (R4), @GHB$Q_OLD_CSBS+ -; Save CSB at end of CSB queue
                        14C2  3652                   GHB_AREA+4                ;   since queue is time ordered
     ECC0 CF    97      14C2  3653          DECB    GHB_B_OLD_CSBCNT           ; Use up one more slot on queue
          0C    18      14C6  3654          BGEQ    190$                       ; Br if still within CSB limits
     ECBA CF    96      14C8  3655          INCB    GHB_B_OLD_CSBCNT           ; Else, restore count
50   EC60 DF    0F      14CC  3656          REMQUE  @GHB$Q_OLD_CSBS+ -         ; Remove oldest entry
                        14D1  3657                   GHB_AREA,R0
        06CC    30      14D1  3658 180$:    BSBW    LT$DEALPOOL                ; Deallocate the CSB block
                05      14D4  3659 190$:    RSB
```

```
                14D5  3661          .SBTTL  LT$CREATEUCB  - Create an LT UCB
                14D5  3662 ;++
                14D5  3663 ; LT$CREATEUCB - Create an LT UCB
                14D5  3664 ;
                14D5  3665 ;        Create a UCB by cloning it from the master UCB zero and
                14D5  3666 ;        initialize it as part of the io database.  We can't lock
                14D5  3667 ;        the io database for write access, because we aren't a
                14D5  3668 ;        process.  So we just do it and hope for the best.  The
                14D5  3669 ;        modifications are such that code scanning the database will
                14D5  3670 ;        not be fooled if they are straight forward.  It is possible
                14D5  3671 ;        to write code that will be confused when a new ucb appears,
                14D5  3672 ;        but we don't have any solution other than waiting for the
                14D5  3673 ;        the request to come here when the database is not locked.
                14D5  3674 ;        That would probably mean that many connection attempts would
                14D5  3675 ;        time out.
                14D5  3676 ;
                14D5  3677 ;                        NB !!
                14D5  3678 ;
                14D5  3679 ;        The I/O data base is not locked!!
                14D5  3680 ;
                14D5  3681 ; Inputs:
                14D5  3682 ;        none
                14D5  3683 ;
                14D5  3684 ; Outputs:
                14D5  3685 ;        R5 = new UCB address
                14D5  3686 ;        R0 = success or failure
                14D5  3687 ;
                14D5  3688 ;        R1-R4 clobbered
                14D5  3689 ;--
                14D5  3690
                14D5  3691 LT$CREATEUCB:
55   EC93 CF    D0  14D5  3692          MOVL    GHB_L_UCB0, R5          ; Get the UCB to clone from
 00000000'GF    16  14DA  3693          JSB     G^IOC$CLONE_UCB         ; Clone the UCB
       06 50    E9  14E0  3694          BLBC    R0,90$                  ; Exit on error
     55   52    D0  14E3  3695          MOVL    R2,R5                   ; Else, copy new UCB address
       5C A5    B4  14E6  3696          CLRW    UCB$W_REFC(R5)          ; No refs yet!
                05  14E9  3697 90$:     RSB
```

```
                        14EA  3699          .SBTTL  LT$DESTROYUCB  - Destroy an LT UCB
                        14EA  3700 ;++
                        14EA  3701 ; LT$DESTROYUCB - Destroy an LT UCB
                        14EA  3702 ;
                        14EA  3703 ;        Delete a ucb from the io data base that has been disconnected.
                        14EA  3704 ;        This routine must be called at high IPL when the I/O database
                        14EA  3705 ;        is not in use by anyone.  We are going to deallocate a block
                        14EA  3706 ;        and if anyone has its address in a register, we are going
                        14EA  3707 ;        to go crash with a bang.  This situation is not quite the
                        14EA  3708 ;        same as the creation case where one can be much more cavalier.
                        14EA  3709 ;
                        14EA  3710 ;        Also note, that we must ensure that the LT UCB has been detached
                        14EA  3711 ;        from any VT UCB, and that the REFC has gone to zero. There is
                        14EA  3712 ;        a condition where the class driver will not detach a VT UCB
                        14EA  3713 ;        even though the disconnect entry point is called. This occurs if
                        14EA  3714 ;        loginout has not been run yet, but is about to be run. In this
                        14EA  3715 ;        case, the class driver will not have detached the LT UCB from the
                        14EA  3716 ;        VT UCB and we can check for that condition.
                        14EA  3717 ;
                        14EA  3718 ;                NB !!!
                        14EA  3719 ;        The I/O database is NOT locked for write access, BUT the MUTEX
                        14EA  3720 ;        is available!
                        14EA  3721 ;
                        14EA  3722 ; Inputs:
                        14EA  3723 ;        R5 = UCB to destroy
                        14EA  3724 ;
                        14EA  3725 ; Outputs:
                        14EA  3726 ;        R5 = UCB address
                        14EA  3727 ;        R0-R1 are destroyed.
                        14EA  3728 ;--
                        14EA  3729
                        14EA  3730 LT$DESTROYUCB:
00C0 C5   55    D1      14EA  3731          CMPL    R5,UCB$L_TT_LOGUCB(R5)  ; Is this UCB ready for deletion?
          12    12      14EF  3732          BNEQ    50$                     ; Br if no, requeue it
       5C A5    B5      14F1  3733          TSTW    UCB$W_REFC(R5)          ; All refs gone?
          0D    12      14F4  3734          BNEQ    50$                     ; Br if no, requeue it
                        14F6  3735          SETBIT  #UCB$V_DELETEUCB, -     ; Else, allow UCB to be deleted
                        14F6  3736                  UCB$L_STS(R5)
 00000000'GF    16      14FB  3737          JSB     G^IOC$DELETE_UCB        ; Delete the UCB
          0E    11      1501  3738          BRB     90$                     ; leave
                        1503  3739
          04    90      1503  3740 50$:     MOVB    #UCB$C_LT_STATE_KILL,-  ; Enter the KILL UCB state
     0144 C5            1505  3741                   UCB$B_LT_STATE(R5)
                        1508  3742          CLRBIT  #UCB$V_LT_DEAD,-        ; Clear the dead indicator
                        1508  3743                   UCB$B_LT_LATSTS(R5)
        014A    30      150E  3744          BSBW    LT$HANGUP_UCB           ; And hangup the UCB again!
                05      1511  3745 90$:     RSB                             ; Return
```

```
                        1512  3747          .SBTTL   LT$ALC_UCB - Allocate New LT UCB
                        1512  3748 ;++
                        1512  3749 ; LT$ALC_UCB - Allocate a new LT UCB
                        1512  3750 ;
                        1512  3751 ; Subroutine to select a UCB for a slot.
                        1512  3752 ;
                        1512  3753 ; Inputs:
                        1512  3754 ;        R7 =  receive buffer slot address
                        1512  3755 ;        R8 =  CSB address
                        1512  3756 ;
                        1512  3757 ; Outputs:
                        1512  3758 ;        R0 =  success or failure
                        1512  3759 ;        R5 =  UCB address
                        1512  3760 ;        R7 =  receive buffer slot address
                        1512  3761 ;        R8 =  CSB address
                        1512  3762 ;        R1,R2,R3,R4 modified
                        1512  3763 ;--
                        1512  3764
                        1512  3765 LT$ALC_UCB:
          50    D4      1512  3766          CLRL    R0                      ; Return failure if no UCB
                        1514  3767
                        1514  3768 ;;&&     *TEMP*  concentrator bug
                        1514  3769
54   6C A8    DE        1514  3770          MOVAL   CSB_L_UCBLST(R8), R4    ; Search the list for this CSB
   53    54   D0        1518  3771          MOVL    R4, R3                  ; Save the address
51   69 A8    9A        151B  3772          MOVZBL  CSB_B_MAX_SLOTS(R8), R1 ; Get maximum in list
   55    84   D0        151F  3773 1$:      MOVL    (R4)+, R5               ; Look in the table
         0B   13        1522  3774          BEQL    2$                      ; A free slot, my word
      01 A7   91        1524  3775          CMPB    SLT_B_SRCID(R7),-       ; Is this the same terminal calling
    0140 C5             1527  3776                  UCB$B_LT_REMID(R5)      ;  again?
         03   12        152A  3777          BNEQ    2$                      ; No, then continue
       010C   31        152C  3778          BRW     95$                     ; Yes, then setup message again
      ED 51   F5        152F  3779 2$:      SOBGTR  R1, 1$                  ; look again
                        1532  3780
                        1532  3781 ;;&&     *TEMP* concentrator bug
                        1532  3782
                        1532  3783
54   6C A8    DE        1532  3784          MOVAL   CSB_L_UCBLST(R8), R4    ; Search the list for this CSB
   53    54   D0        1536  3785          MOVL    R4, R3                  ; save the address
51   69 A8    9A        1539  3786          MOVZBL  CSB_B_MAX_SLOTS(R8), R1 ; Get maximum in list
   55    84   D0        153D  3787 10$:     MOVL    (R4)+, R5               ; Look in the table
         06   13        1540  3788          BEQL    20$                     ; A free slot, my word
      F8 51   F5        1542  3789          SOBGTR  R1, 10$                 ; look again
       00F2   31        1545  3790 15$:     BRW     90$                     ; None to be had here, so fail
                        1548  3791
      02 A7   91        1548  3792 20$:     CMPB    SLT_B_COUNT(R7),-       ; Is slot big enough?
         05             154B  3793                  #<SLT_B_DST_NAMLEN-SLT_T_DATA>+2
         0A   18        154C  3794          BGEQ    30$                     ; Br if yes, okay
                        154E  3795          INC_CTR CSB_Z_LCB+LCB_B_INV_SLOT(R8) ; Else, say invalid slot
         ED   11        1556  3796          BRB     15$                     ; And exit
                        1558  3797                                          ;
         54   DD        1558  3798 30$:     PUSHL   R4                      ; Save the address in table
   54    53   C2        155A  3799          SUBL2   R3, R4                  ; Compute the index for this CSB
   54    04   C6        155D  3800          DIVL2   #4, R4                  ;  as a small integer
         54   DD        1560  3801          PUSHL   R4                      ; Save the index too
       FF70   30        1562  3802          BSBW    LT$CREATEUCB            ; Create a raw lt UCB
         54 8ED0        1565  3803          POPL    R4                      ; Obtain the index
```

```
                53 8ED0  1568  3804          POPL    R3                          ;  and its address in CSB table
                         156B  3805                                              ;
             D7 50   E9  156B  3806          BLBC    R0, 15$                     ; Do we have one?, nope, too bad
          73    55   D0  156E  3807          MOVL    R5, -(R3)                   ; Store its address in CSB table
             F4F7      30  1571  3808          BSBW    LT$UCB_INIT                 ; Initialize the ucb
     0141 C5    54   90  1574  3809          MOVB    R4,UCB$B_LT_LOCID(R5)       ; Set local slot index
          01 A7       90  1579  3810          MOVB    SLT_B_SRCID(R7),-           ; Load remote circuit id
     0140 C5                157C  3811                  UCB$B_LT_REMID(R5)          ;
50   03 A7   04   00  EF  157F  3812          EXTZV   #0,#4,SLT_B_CRED(R7),R0     ; Get credit total extended
   C146 C5   50   01  81  1585  3813          ADDB3   #1,R0,UCB$B_LT_TCRED(R5)    ; Give ourself an extra credit for
                         158B  3814                                              ;   sending the start slot
          91 8F       90  158B  3815          MOVB    #SLT_C_STR_SLOT@4!LAT$C_MAX_RCRED,- ; Set slot type and credits
     0148 C5                158E  3816                  UCB$B_LT_RCRED(R5)
                 01   8E  1591  3817          MNEGB   #LAT$C_MAX_RCRED,-          ; Set credits extended to remote end
     0147 C5                1593  3818                  UCB$B_LT_XCRED(R5)          ;   kept as negative value
     0082 C5     B4  1596  3819          CLRW    UCB$W_ERRCNT(R5)            ; Reset the error counter that
                         159A  3820                                              ; we bump as we shut down.
                 A8  159A  3821          BISW    #<UCB$M_INT!-                   ; Expecting interrupts & online
                         159B  3822                  UCB$M_ONLINE>,-             ;
          64 A5   12     159B  3823                  UCB$L_STS(R5)               ;
     0134 C5    58   D0  159E  3824          MOVL    R8,UCB$L_LT_CSB(R5)         ; Load CSB address
          2D A8      96  15A3  3825          INCB    CSB_B_REFC(R8)              ; One more UCB here
                         15A6  3826          ;
                         15A6  3827          ;  Compute maximum allowed slot size
                         15A6  3828          ;
          FF 8F       90  15A6  3829          MOVB    #LAT$C_MAX_SLOTSIZ,-        ; Assume ours is the maximum
     0149 C5                15A9  3830                  UCB$B_LT_SLOTSZ(R5)         ;  slot size
          06 A7       91  15AC  3831          CMPB    SLT_B_DT_SLTSIZ(R7),-       ; Is remotes size smaller?
     0149 C5                15AF  3832                  UCB$B_LT_SLOTSZ(R5)
                 06   1E  15B2  3833          BGEQU   40$                         ; Br if no, use ours
          06 A7       90  15B4  3834          MOVB    SLT_B_DT_SLTSIZ(R7),-       ; Else, take remotes
     0149 C5                15B7  3835                  UCB$B_LT_SLOTSZ(R5)
                 02   90  15BA  3836 40$:     MOVB    #UCB$C_LT_STATE_RUN,-       ; Set the initial state to RUN
     0144 C5                15BC  3837                  UCB$B_LT_STATE(R5)
                 05   90  15BF  3838          MOVB    #GHB_C_STRT_SLOTL,-         ; Build a start slot
     014B C5                15C1  3839                  UCB$B_LT_CORC(R5)
                 55   DD  15C4  3840          PUSHL   R5                          ; Save UCB address
                 05   28  15C6  3841          MOVC3   #GHB_C_STRT_SLOTL,-         ; Make it look like output data
     EED2 CF                15C8  3842                  GHB_T_STRT_SLOT,-
     014C C5                15CB  3843                  UCB$B_LT_CBUF(R5)
                 55 8ED0  15CE  3844          POPL    R5                          ; Save UCB address
                 01   88  15D1  3845          BISB    #UCB$M_LT_DATA,-            ; Force data through output slot
     0142 C5                15D3  3846                  UCB$B_LT_DATAW(R5)
                         15D6  3847          ;
                         15D6  3848          ;  Check for the disable login flag
                         15D6  3849          ;
   51    07 A7       9E  15D6  3850          MOVAB   SLT_B_DST_NAMLEN(R7),R1     ; Skip to dest. slot name field
      50    81       9A  15DA  3851          MOVZBL  (R1)+,R0                    ; Get size of dest. name field
      51    50       C0  15DD  3852          ADDL    R0,R1                       ; Skip the dest. name field
      50    81       9A  15E0  3853          MOVZBL  (R1)+,R0                    ; Get size of src name field
      51    50       C0  15E3  3854          ADDL    R0,R1                       ; Skip the src name field
   50    04 A7       9E  15E6  3855          MOVAB   SLT_T_DATA(R7),R0           ; Compute size of slot skipped
52    51    50       C3  15EA  3856          SUBL3   R0,R1,R2                    ;          ...
   50    02 A7       9A  15EE  3857          MOVZBL  SLT_B_COUNT(R7),R0          ; Get size of entire slot
      50    52       C2  15F2  3858          SUBL    R2,R0                       ; Compute bytes remaining
            30       15  15F5  3859          BLEQ    70$                         ; Br if no more data, do login
      52    81       9A  15F7  3860 60$:     MOVZBL  (R1)+,R2                    ; Get parameter code
```

```
                2B   13  15FA  3861          BEQL    70$                        ; Br if none, assume auto-login
                50   D7  15FC  3862          DECL    R0                         ; One less byte left
                27   15  15FE  3863          BLEQ    70$                        ; Br if no more data, do login
           53   81   9A  1600  3864          MOVZBL  (R1)+,R3                   ; Get size of parameter
           50   53   C2  1603  3865          SUBL    R3,R0                      ; Get remaining byte count
                1F   15  1606  3866          BLEQ    70$                        ; Br if no more data, do login
           54   51   D0  1608  3867          MOVL    R1,R4                      ; Save current parameter address
           51   53   C0  160B  3868          ADDL    R3,R1                      ; Skip to next parameter
           01   52   D1  160E  3869          CMPL    R2,#SLT_C_START_FLAG       ; Is this the start slot flag byte?
                E4   12  1611  3870          BNEQ    60$                        ; Br if no, look for next
        0C 64   02   E1  1613  3871          BBC     #SLT_FLAG_V_BIND,(R4),65$  ; Br if not a bind request
                         1617  3872          CLRBIT  #TT2$V_HANGUP,-            ; Else, set terminal to NOHANGUP
                         1617  3873                  UCB$L_TT_DECHA1(R5)        ;   on logout
                         161D  3874          SETBIT  #UCB$V_LT_BOUND,-          ; Don't delete UCB after STOP slot
                         161D  3875                  UCB$B_LT_LATSTS(R5)        ; ... only if we must KILL the UCB!
        0F 64   01   E0  1623  3876 65$:     BBS     #SLT_FLAG_V_LOGIN,(R4),80$ ; Br if login is disabled
                         1627  3877          ;
                         1627  3878          ;  Ask the class driver to log us in
                         1627  3879          ;
     51   0114 C5   D0  1627  3880 70$:     MOVL    UCB$L_TT_CLASS(R5), R1     ; Class dispatch vector
53   00002000 8F   D0  162C  3881          MOVL    #1@13, R3                  ; Framming error with zero byte
           14 B1   16  1633  3882          JSB     @CLASS_READERROR(R1)       ; Cause read error
     50   00'8F   9A  1636  3883 80$:     MOVZBL  #SS$_NORMAL, R0            ; Do we have a UCB for you.
                   05  163A  3884 90$:     RSB
                         163B  3885
                         163B  3886 ;;&&     *TEMP* concentrator  bug
                         163B  3887
                         163B  3888 95$:     SETBIT  #GHB$V_REPCREATE, -        ; Set mask bit
                         163B  3889                  GHB$L_PROTOMASK+GHB_AREA ; Repeat create of slot
                         1641  3890          INC_CTR GHB$L_PROTOCOL+GHB_AREA ; Increment protocol error counter
              FF6C   31  164B  3891          BRW     40$
                         164E  3892 ;;&&     *TEMP*  concentrator bug
```

```
                 164E          .SBTTL  LT$HANGUP_UCB  - Cause Terminal Hangup On UCB
                 164E  ;++
                 164E  ; LT$HANGUP_UCB - Hangup an LT UCB
                 164E  ;
                 164E  ; Cause all the hangup effects on a terminal device.  This should cause
                 164E  ; process rundown and deallocation of all channels to the UCB.
                 164E  ;
                 164E  ; Inputs:
                 164E  ;       R5 = UCB address
                 164E  ;
                 164E  ; Outputs:
                 164E  ;       none.
                 164E  ;
                 164E  ;--
                 164E          .ENABL  LSB
                 164E  LT$HANGUP_UCB_NOW:
                 164E          ;
                 164E          ;  Dispatch on UCB (slot) state
                 164E          ;
                 164E          $DISPATCH  UCB$B_LT_STA   (R5),TYPE=B,-
                 164E          <-      ; state             tion
                 164E          <UCB$C_LT_STATE_STOP        ILLUCB>,-   ; Already disconnected,
                 164E          >
                 1656
                 1656          ;  Else
                 1656
        04   90  1656          MOVB    #UCB$C_LT_STATE_KILL,-          ; Enter the KILL UCB state
     0144 C5     1658                   UCB$B_LT_STATE(R5)
                 165B
                 165B  LT$HANGUP_UCB:
        1F   BB  165B          PUSHR   #^M<R0,R1,R2,R3,R4>             ; Save registers
  64 A5  02  AA  165D          BICW    #UCB$M_INT,UCB$L_STS(R5)        ; No interrupt expected (allows
                 1661                                                  ;   STARTIO to be called again
                 1661                                                  ;   to flush data).
                 1661          SETBIT  #UCB$V_LT_HANGUP,-              ; Indicate that we're shutting
                 1661                   UCB$B_LT_LATSTS(R5)            ;   down
50   0114 C5  D0  1667          MOVL    UCB$L_TT_CLASS(R5),R0           ; Class driver vector address
       18 B0  16  166C          JSB     @CLASS_DISCONNECT(R0)           ; Call routine
        1F   BA   166F          POPR    #^M<R0,R1,R2,R3,R4>             ; Restore the regs
             05   1671  90$:    RSB
                  1672
                  1672          .DSABL  LSB
```

```
                        1672  3937          .SBTTL  LT$DISCONNECT  - Port Disconnect
                        1672  3938 ;++
                        1672  3939 ; LT$DISCONNECT - Disconnect the terminal Port driver
                        1672  3940 ;
                        1672  3941 ;       Port driver entry to disconnect a terminal.  called when
                        1672  3942 ;       Last reference is gone.  here we free the slot in the buffer
                        1672  3943 ;       And bump the sequence number.  also we set the UCB offline.
                        1672  3944 ;
                        1672  3945 ; Inputs:
                        1672  3946 ;       R5 = UCB address
                        1672  3947 ;       R0 -> lbc = delete UCB
                        1672  3948 ;       R0 -> lbs = do not delete UCB
                        1672  3949 ;
                        1672  3950 ; Outputs:
                        1672  3951 ;       R5 = UCB address
                        1672  3952 ;       All other registers are preserved.
                        1672  3953 ;
                        1672  3954 ; Implicit inputs:
                        1672  3955 ;       UCB$B_LT_STATE = KILL -> kill UCB immediately.
                        1672  3956 ;                        STOP -> send stop slot, then kill UCB.
                        1672  3957 ;                        ???? -> all others, set to STOP state.
                        1672  3958 ;
                        1672  3959 ;--
                        1672  3960
                        1672  3961 LT$DISCONNECT:
                        1672  3962         ;
                        1672  3963         ; Do not touch the UCB if it is the template UCB0
                        1672  3964         ;
EAF5 CF    55    D1     1672  3965         CMPL    R5,GHB_L_UCB0           ; Is this the template ucb ?
           23    13     1677  3966         BEQL    90$                     ; Br if yes, don't touch it!
                        1679  3967         ;
                        1679  3968         ;   Dispatch on our UCB state
                        1679  3969         ;
                        1679  3970         $DISPATCH  UCB$B_LT_STATE(R5),TYPE=B,-
                        1679  3971         <-      ; state         ; action
                        1679  3972         <UCB$C_LT_STATE_KILL    50$>,-  ; KILLed, just kill the UCB now
                        1679  3973         <UCB$C_LT_STATE_STOP    90$>,-  ; STOPed, we've been here before
                        1679  3974         >
                        1683  3975                                         ; ALL other states,
                        1683  3976         ;
                        1683  3977         ;  If the circuit is gone, or there is no CSB for the UCB
                        1683  3978         ;  then just delete the UCB immediately
                        1683  3979         ;
0134 C5    D5           1683  3980         TSTL    UCB$L_LT_CSB(R5)        ; Do we have a circuit?
           10    13     1687  3981         BEQL    50$                     ; Br if none
                        1689  3982         ;
                        1689  3983         ;   Conditionally skip deallocation of UCB, on "hangup".
                        1689  3984         ;
  10 50    E8           1689  3985         BLBS    R0,90$                  ; Br if not to "hangup" UCB
     03    90           168C  3986         MOVB    #UCB$C_LT_STATE_STOP,-  ; Else, set STOP state,
0144 C5                 168E  3987                  UCB$B_LT_STATE(R5)     ;  to send STOP slot
                        1691  3988         ;
                        1691  3989         ;  Make sure that no data is given to CLASS driver, after
                        1691  3990         ;  we've been here!
                        1691  3991         ;
                        1691  3992         SETBIT  #UCB$V_LT_HANGUP,-      ; Indicate that we're shutting
                        1691  3993                  UCB$B_LT_LATSTS(R5)    ;   down
```

```
   03   11  1697  3994          BRB     90$                     ; Delete UCB when STOP is sent!
                1699  3995
 0001   30  1699  3996 50$:     BSBW    LT$KILLUCB              ; Get rid of the UCB now
                169C  3997
        05  169C  3998 90$:     RSB                             ; Done
                169D  3999
```

```
                169D          .SBTTL  LT$KILLUCB  - Delete a dried up UCB
                169D  ;++
                169D  ; LT$KILLUCB - Delete an LT UCB
                169D  ;
                169D  ; The UCB is all dried up one way or another.  So we must delete it if
                169D  ; possible, else put it on the deadlink list to be deleted on the next
                169D  ; timer tick.
                169D  ;
                169D  ; Inputs:
                169D  ;       R5 = UCB address
                169D  ;
                169D  ; Outputs:
                169D  ;       R5 = UCB address
                169D  ;       All other registers preserved.
                169D  ;
                169D  ;--
                169D
                169D  LT$KILLUCB:
        1F   BB 169D          PUSHR   #^M<R0,R1,R2,R3,R4>     ; Save some volatile regs
             AA 169F          BICW    #<UCB$M_INT! -          ; No interrupt expected
                16A0                  UCB$M_ONLINE>,-         ; Not online, but free now
64 A5    12     16A0                  UCB$L_STS(R5)           ; For use again
      F583   30 16A3          BSBW    LT$FLUSH_DATA           ; Flush all output data
      0027   30 16A6          BSBW    LT$SIDELINEUCB          ; Set the UCB aside from the circuit
        01   E3 16A9          BBCS    #UCB$V_LT_DEAD,-        ; If we already linked UCB to deadlist
12 0143 C5      16AB                  UCB$B_LT_LATSTS(R5),20$ ;  then we have made an error
                16AF          SETBIT  #GHB$V_REPDISC,-        ; KILLUCB called twice
                16AF                  GHB$L_PROTOMASK+GHB_AREA;
                16B5          INC_CTR GHB$L_PROTOCOL+GHB_AREA ; Increment protocol error counter
        0C   11 16BF          BRB     90$                     ;
                16C1  20$:    ;
                16C1          ;  We must destroy the UCB (not called from $DASSGN, so queue the
                16C1          ;  UCB to the deadlist so it can be deallocated by the timer service.
                16C1          ;  This is because the CLASS driver still uses the UCB even if we
                16C1          ;  deallocate it!
                16C1          ;
     EAAB CF    D0 16C1          MOVL    GHB_L_DEADLINK,-        ; Link UCB into chain of dead UCBs
     0138 C5       16C5                  UCB$L_LT_DEADLINK(R5)   ; for disposal at timeout
EAA3 CF    55    D0 16C8          MOVL    R5, GHB_L_DEADLINK      ;
        1F   BA 16CD  90$:    POPR    #^M<R0,R1,R2,R3,R4>     ; Restore the regs
             05 16CF          RSB
```

```
                    16D0  4043          .SBTTL  LT$SIDELINEUCB  - Set a UCB aside
                    16D0  4044 ;++
                    16D0  4045 ; LT$SIDELINEUCB
                    16D0  4046 ;
                    16D0  4047 ; Functional description:
                    16D0  4048 ;
                    16D0  4049 ; Set a UCB aside to dry up on its own.  Disconnect it from a running
                    16D0  4050 ; circuit.
                    16D0  4051 ;
                    16D0  4052 ; Inputs:
                    16D0  4053 ;        R5 = UCB address
                    16D0  4054 ;
                    16D0  4055 ; Outputs:
                    16D0  4056 ;        none
                    16D0  4057 ;        All registers preserved.
                    16D0  4058 ;
                    16D0  4059 ;--
                    16D0  4060
                    16D0  4061 LT$SIDELINEUCB:
          18   BB   16D0  4062          PUSHR   #^M<R3,R4>                  ; Save registers
54   0134 C5   D0   16D2  4063          MOVL    UCB$L_LT_CSB(R5), R4        ; Get the CSB address
          15   13   16D7  4064          BEQL    10$                         ; None, this is already done
53   0141 C5   9A   16D9  4065          MOVZBL  UCB$B_LT_LOCID(R5), R3      ; Use the local slot index in UCB
     68 A443   D4   16DE  4066          CLRL    CSB_L_UCBLST-4(R4)[R3]      ; to adjust state of UCB list in CSB
     0134 C5   D4   16E2  4067          CLRL    UCB$L_LT_CSB(R5)            ; Remember we have been here
       2D A4   97   16E6  4068          DECB    CSB_B_REFC(R4)              ; One less UCB on this circuit
          03   12   16E9  4069          BNEQ    10$                         ; We have more UCBs here
        0044   30   16EB  4070          BSBW    LT$STOPCIRC                 ; Then don't wait for return traffic
          18   BA   16EE  4071 10$:     POPR    #^M<R3,R4>                  ; Restore registers
               05   16F0  4072          RSB                                 ; Return to caller
```

```
                    16F1  4074          .SBTTL  LT$CIRCDEAD  - Declare circuit dead
                    16F1  4075 ;++
                    16F1  4076 ; LT$CIRCDEAD  - Declare circuit dead.
                    16F1  4077 ;
                    16F1  4078 ; Functional description:
                    16F1  4079 ;
                    16F1  4080 ;        Log off all the processes on each UCB and waiting for them
                    16F1  4081 ;        to free their UCBs.  This will cause the CSB to go free too.
                    16F1  4082 ;        If this CSB has no ucb's attached, then the CSB is just cleaned up.
                    16F1  4083 ;
                    16F1  4084 ; Inputs:
                    16F1  4085 ;        R8 = CSB address
                    16F1  4086 ;
                    16F1  4087 ; Outputs:
                    16F1  4088 ;        All UCB's hungup.
                    16F1  4089 ;
                    16F1  4090 ;--
                    16F1  4091
                    16F1  4092 LT$CIRCDEAD:
   03FF 8F    BB    16F1  4093          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9> ; Save registers
     2D A8    95    16F5  4094          TSTB    CSB_B_REFC(R8)                     ; Do we have any UCBs?
        18    13    16F8  4095          BEQL    100$                               ; Br if no, so deallocate CSB
57   69 A8    9A    16FA  4096          MOVZBL  CSB_B_MAX_SLOTS(R8), R7            ; Else, number of UCBs to scan
56   6C A8    DE    16FE  4097          MOVAL   CSB_L_UCBLST(R8), R6               ; List of the UCBs
   55   86    D0    1702  4098 20$:     MOVL    (R6)+, R5                          ; Get next UCB address
        03    18    1705  4099          BGEQ    60$                                ; Not system address
      FF44    30    1707  4100          BSBW    LT$HANGUP_UCB_NOW                  ; Hangup the process on this UCB
      F5 57   F5    170A  4101 60$:     SOBGTR  R7, 20$                            ; For all the units here
                    170D  4102
   03FF 8F    BA    170D  4103 90$:     POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9> ; Restore registers
        05    05    1711  4104          RSB                                        ; Return to caller
                    1712  4105          ;
                    1712  4106          ; Refc has gone to zero, send a stop message
                    1712  4107          ;
54      58    D0    1712  4108 100$:    MOVL    R8, R4                             ; Setup CSB address
        1B    10    1715  4109          BSBB    LT$STOPCIRC                        ; Send a stop message on circ
        F4    11    1717  4110          BRB     90$                                ; And exit
                    1719  4111
```

```
                                  1719  4113          .SBTTL  LT$STOPCIRC  - Transmit stop message
                                  1719  4114 ;++
                                  1719  4115 ; LT$STOPCIRC - Transmit stop message
                                  1719  4116 ;
                                  1719  4117 ; Functional description:
                                  1719  4118 ;
                                  1719  4119 ; Format and transmit stop message
                                  1719  4120 ;
                                  1719  4121 ; Inputs
                                  1719  4122 ;        R4 = CSB address
                                  1719  4123 ;        CSB_L_STOPREASON        addr of stop reason block
                                  1719  4124 ;
                                  1719  4125 ; Outputs:
                                  1719  4126 ;        none
                                  1719  4127 ;
                                  1719  4128 ;--
                                  1719  4129
                                  1719  4130 DEFREASN:
6E 69 6D 72 65 74 20 65 64 6F 4E 00' 1719  4131          .ASCIC  /Node terminated circuit./
74 69 75 63 72 69 63 20 64 65 74 61  1725
                                 2E  1731
                                 18  1719
                                  1732  4132
                                  1732  4133 LT$STOPCIRC:
                     03FF 8F     BB  1732  4134          PUSHR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9> ; Save registers
                     58   54     D0  1736  4135          MOVL    R4, R8                           ; Set CSB addr in r8
                       2D A8     95  1739  4136          TSTB    CSB_B_REFC(R8)                   ; We are not supposed to send
                          06     13  173C  4137          BEQL    10$                              ; stop messages with
                                     173E  4138          DEBUG                                    ; outstanding ucbs on circ
                       0B A8     91  1744  4139 10$:     CMPB    CSB_B_STATE(R8),-                ; Have we been here before?
                          00         1747  4140                  #CSB_C_STATE_HALT                ;
                          03     12  1748  4141          BNEQ    15$                              ; Br if no
                        008A     31  174A  4142          BRW     90$                              ; Else, exit
                 66 A8    01     B0  174D  4143 15$:     MOVW    #1,CSB_W_TIMRESET(R8)            ; Set timers for quick timeout
                 50 A8    01     B0  1751  4144          MOVW    #1,CSB_W_TIMEOUT(R8)             ;
                          00     90  1755  4145          MOVB    #CSB_C_STATE_HALT,-              ; and set state to halted
                    0B A8            1757  4146                  CSB_B_STATE(R8)                  ;
                    60 A8     B4  1759  4147          CLRW    CSB_W_LOCID(R8)                  ; Zero the local ID
                 50 34 A8     7E  175C  4148          MOVAQ   CSB_Q_XBUFQ(R8), R0              ; Transmit buffer queue
                 57 00 B0     0F  1760  4149          REMQUE  @(R0), R7                        ; Do we have a free buffer?
                          0F     1C  1764  4150          BVC     30$                              ; Yep, got it
                 57 3C B8     0F  1766  4151          REMQUE  @CSB_Q_XWAITQ(R8), R7            ; Get a waiting bfr
                          06     1C  176A  4152          BVC     20$                              ; Ok, so use it
                                     176C  4153          DEBUG                                    ; All buffers gone?????
                    59 A8     97  1772  4154 20$:     DECB    CSB_B_XMTCNT(R8)                 ; One less buffer on waitq
                 60       67     0E  1775  4155 30$:     INSQUE  (R7), (R0)                       ; Make it a free buffer
                 59       57     D0  1778  4156          MOVL    R7, R9                           ; Save bfr header adr
                 57 48 A7        9E  177B  4157          MOVAB   XMT_T_DATA(R7), R7               ; Stop message data
                          08     90  177F  4158          MOVB    #MTYP_C_HALT@FLAG_V_MTYPE,-      ; Send a HALT message
                    5C A8            1781  4159                  CSB_B_FLAG(R8)                   ;
                    5D A8     94  1783  4160          CLRB    CSB_B_NUM_SLOTS(R8)              ; No slot data
                 53 50 A9     9E  1786  4161          MOVAB   XMT_T_MDATA(R9), R3              ; Set default end of data
                          83     94  178A  4162          CLRB    (R3)+                            ; Set default reason code
                 50 8A AF     9E  178C  4163          MOVAB   DEFREASN, R0                     ; Store a default text string
                 51       60     9A  1790  4164          MOVZBL  (R0), R1                         ; the count
                          51     D6  1793  4165          INCL    R1                               ; including the count
              63 60       51     28  1795  4166          MOVC3   R1, (R0), (R3)                   ; move count and string
```

```
56   4C A8   D0  1799  4167          MOVL    CSB_L_STOPREASON(R8), R6        ; Do we have a reason
        29   13  179D  4168          BEQL    40$                             ; No reason
50   02 A6   9A  179F  4169          MOVZBL  2(R6), R0                       ; Set the reason bit
                 17A3  4170          SETBIT  R0,GHB$L_PROTOMASK+GHB_AREA     ; and count the error
                 17A9  4171          INC_CTR GHB$L_PROTOCOL+GHB_AREA         ; as a halt
50 A9   03 A6   90  17B3  4172          MOVB    3(R6), -                        ; Store the reason code
                 17B8  4173                  STOP_B_RCODE+XMT_T_MDATA(R9)    ;
50   66   3C  17B8  4174          MOVZWL  (R6), R0                        ; Offset to the string
50   56   C0  17BB  4175          ADDL    R6, R0                          ; Set the address
51   60   9A  17BE  4176          MOVZBL  (R0), R1                        ; Get the count
        51   D6  17C1  4177          INCL    R1                              ; and set the count
51 A9   60   51  28  17C3  4178          MOVC3   R1, (R0), -                     ; copy the counted string
                 17C8  4179                  STOP_B_RLEN+XMT_T_MDATA(R9)     ; Leave r3 pointing beyond
                 17C8  4180 40$:
1C A9   53   D0  17C8  4181          MOVL    R3, CXB$L_T_ENDADR(R9)          ; and store in buffer
54   58   D0  17CC  4182          MOVL    R8,R4                           ; copy CSB address
     0264   30  17CF  4183          BSBW    LT$XMIT                         ; and transmit the msg
03FF 8F   BA  17D2  4184 50$:   POPR    #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9> ; Restore registers
        05  17D6  4185          RSB                                     ; Return to caller
                 17D7  4186
                 17D7  4187 ; Stop message sent once
                 17D7  4188 ; Just wipe CSB now
                 17D7  4189
54   58   D0  17D7  4190 90$:    MOVL    R8, R4                          ; CSB address to R4
     FC0E   30  17DA  4191          BSBW    LT$DEALOCSB                     ; Deallocate CSB to pool
        F3   11  17DD  4192          BRB     50$                             ; And exit
                 17DF  4193
```

```
                        17DF          .SBTTL  LT$SET_TIMER  - Start Timer
                        17DF  ;++
                        17DF  ; LT$SET_TIMER - Start the timer
                        17DF  ;
                        17DF  ; Functional description:
                        17DF  ;
                        17DF  ;       Start timer ticking to scan the CSB's for dead circuits
                        17DF  ;
                        17DF  ; Inputs:
                        17DF  ;       R0-R2 available
                        17DF  ;
                        17DF  ; Implicit Inputs:
                        17DF  ;
                        17DF  ;       IPL = LAT$C_IPL (8)
                        17DF  ;
                        17DF  ; Outputs:
                        17DF  ;       R0-R2 are destroyed.
                        17DF  ;
                        17DF  ;--
                        17DF
                        17DF  LT$SET_TIMER:
50      E8F9 CF    9E   17DF          MOVAB   GHB_G_TQE, R0           ; Address of timer queue element
            01     E2   17E4          BBSS    #GHB_STS_V_ACTIVE,-     ; Br if timer already active
   1F E99B CF           17E6                   GHB_B_STATUS,10$        ;   and mark as active
000D'CF     04     88   17EA          BISB    #DPT$M_NOUNLOAD,DPT$TAB+DPT$B_FLAGS ; Make driver unloadable
            30     B0   17EF          MOVW    #TQE$C_LENGTH,-         ; Setup timer queue entry block
        08  A0          17F1                   TQE$W_SIZE(R0)          ; For its first use
            0F     90   17F3          MOVB    #DYN$C_TQE,-            ; With both the size and the type
        0A  A0          17F5                   TQE$B_TYPE(R0)          ; Code.
            38     BB   17F7          PUSHR   #^M<R3,R4,R5>           ; Save the fork context
        55  50         D0   17F9          MOVL    R0, R5                  ; Use TQE as a fork block
                        17FC  ;
                        17FC  ; Use the TQE as a fork block to start the timer. This is because the
                        17FC  ; timer must be started at or below QUEUEAST to synchronize access to the
                        17FC  ; TIMER QUEUE.
                        17FC  ;
            06     90   17FC          MOVB    #IPL$_QUEUEAST,-        ; Set delivery IPL
        0B  A5          17FE                   FKB$B_FIPL(R5)          ;  of fork block
            08     10   1800          BSBB    40$                     ; Create fork process
E91D CF     01     CE   1802          MNEGL   #1,GHB$L_TIM_ACT+GHB_AREA ; Mark the timer as active for LATCP
            38     BA   1807          POPR    #^M<R3,R4,R5>           ; Restore our context
            05          1809  10$:    RSB                             ; And return
                        180A
                        180A  ;
                        180A  ;       Fork process to start the timer.  We need to go to synch from
                        180A  ;       fork ipl to start the timer.  We do this by going through
                        180A  ;       Queueast IPL to be safe - using the TQE as a fork block.
                        180A  ;
                        180A
00000000'GF     16      180A  40$:    JSB     G^EXE$FORK              ; Fork to start the timer ticking
                        1810          DSBINT  #IPL$_SYNCH             ; Raise to synch
    1853'CF     9E      1816          MOVAB   W^LT$TICK,-             ; Address of tick routine
        0C  A5          181A                  TQE$L_FPC(R5)           ;  is the fork pc
00000000'GF     D0      181C          MOVL    G^TTY$GL_DELTA,-        ; The tick time is from tty sysgen
        20  A5          1822                  TQE$Q_DELTA(R5)         ;  parameter for modem control
            05     90   1824          MOVB    #TQE$C_SSREPT,-         ; Repeating tick
        0B  A5          1826                  TQE$B_RQTYPE(R5)        ;
```

```
50   00000000'GF   7D  1828  4252        MOVQ    G^EXE$GQ_SYSTIME, R0     ; Set the time
50   00000000'GF   C0  182F  4253        ADDL    G^TTY$GL_DELTA, R0       ;    from now
          51   00   D8  1836  4254        ADWC    #0, R1                   ; Carry to top longword
     00000000'GF   16  1839  4255        JSB     G^EXE$INSTIMQ            ; Queue the timer element
                       183F  4256        ENBINT                           ; Restore IPL and leave
                   05  1842  4257        RSB
```

```
                        1843  4259          .SBTTL  LT$TICK  - Timer Service Routine
                        1843  4260 ;++
                        1843  4261 ; LT$TICK - Timer Service Routine
                        1843  4262 ;
                        1843  4263 ; Functional description:
                        1843  4264 ;
                        1843  4265 ;       Enter here on each timer tick.  We scan the CSB table and
                        1843  4266 ;       timeout each UCB with a matching CSB for each CSB that has
                        1843  4267 ;       timed out.  For each receive on a circuit that has not timed
                        1843  4268 ;       out we will reset its timer.
                        1843  4269 ;
                        1843  4270 ; Inputs:
                        1843  4271 ;       R5 = TQE address
                        1843  4272 ;
                        1843  4273 ;       IPL = TIMER
                        1843  4274 ;
                        1843  4275 ; Outputs:
                        1843  4276 ;       R0-R4 are destroyed.
                        1843  4277 ;
                        1843  4278 ;--
                        1843  4279          .ENABLE LOCAL_BLOCK
                        1843  4280
                        1843  4281 STOP_TQE:                                        ; Stop the TQE
   0B A5   04   8A      1843  4282          BICB    #TQE$M_REPEAT,TQE$B_RQTYPE(R5)  ; Clear the repeat flag
 000D'CF   04   8A      1847  4283          BICB    #DPT$M_NOUNLOAD,DPT$TAB+DPT$B_FLAGS ; Make driver reloadable again
     E8D4 CF    D4      184C  4284          CLRL    GHB$L_TIM_ACT+GHB_AREA          ; Tell LATCP we are no longer
                        1850  4285                                                  ;    active
        01DB    31      1850  4286          BRW     300$                            ; And leave
                        1853  4287
                        1853  4288 LT$TICK:
                        1853  4289          DSBINT  #LAT$C_IPL                      ; Raise to synchronize LAT
     0FE0 8F    BB      1859  4290          PUSHR   #^M<R5,R6,R7,R8,R9,R10,R11>     ; Save all the registers
          01    E1      185D  4291          BBC     #GHB_STS_V_ACTIVE,-             ; Br if we went inactive
  E0 E922 CF            185F  4292                   GHB_B_STATUS,STOP_TQE
                        1863  4293 ;
                        1863  4294 ; Scan all CSBs to retransmit in balanced mode and time out circuits
                        1863  4295 ; that appear to be dead.
                        1863  4296 ;
     5B   20    D0      1863  4297          MOVL    #LAT$C_MAX_CSBS, R11            ; Scan the CSB table for
 5A  E7CE CF    DE      1866  4298          MOVAL   GHB_L_CSBTABLE, R10             ; Active entries and check
     58   8A    D0      186B  4299 20$:     MOVL    (R10)+ , R8                     ; Get the CSB address
          43    13      186E  4300          BEQL    90$                             ; Br if no CSB in this slot
                        1870  4301 ;
                        1870  4302 ;       Are we coming out of balanced mode and need to retransmit the
                        1870  4303 ;       First message again??
                        1870  4304 ;
  02   0B A8    91      1870  4305          CMPB    CSB_B_STATE(R8), -              ; State must be running
                        1874  4306                  #CSB_C_STATE_RUN                ;   for active circuits
          26    12      1874  4307          BNEQ    60$                             ; Br if not
       52 A8    B5      1876  4308          TSTW    CSB_W_XMTTMO(R8)                ; Timing out restart message?
          21    13      1879  4309          BEQL    60$                             ; Br if no
       52 A8    B7      187B  4310          DECW    CSB_W_XMTTMO(R8)                ; Check for timeout
          1C    14      187E  4311          BGTR    60$                             ; Br if not time yet
                        1880  4312          INC_CTR GHB$L_RETRANS+GHB_AREA          ; Count a retransmission
                        188A  4313          INC_CTR CSB_Z_LCB+LCB_L_MSG_REXMT(R8)   ;       ...
     54   58    D0      1892  4314          MOVL    R8, R4                          ; CSB address for call
        0203    30      1895  4315          BSBW    LT$REXMIT                       ; Transmit the message
```

```
                52 A8   02   B0  1898  4316          MOVW    #LAT$C_HOST_TIMER,CSB_W_XMTTMO(R8) ; Reset the timeout
                                 189C  4317 ;
                                 189C  4318 ; Timeout circuit if needed
                                 189C  4319 ;
                   50 A8   B7    189C  4320 60$:     DECW    CSB_W_TIMEOUT(R8)                  ; Check the circuit timeout
                      12   12    189F  4321          BNEQ    90$                                ; Br if not expired
                                 18A1  4322          INC_CTR GHB$L_CIRCDOWN+GHB_AREA            ; Count circ down event
                    FE43   30    18AB  4323          BSBW    LT$CIRCDEAD                        ; Declare the circuit dead
                   66 A8   B0    18AE  4324          MOVW    CSB_W_TIMRESET(R8),-               ; Reset the timeout to wait
                   50 A8         18B1  4325                  CSB_W_TIMEOUT(R8)                  ; a while.
                   B5 5B   F5    18B3  4326 90$:     SOBGTR  R11, 20$                           ; Loop till we are done
                                 18B6  4327
                                 18B6  4328 ;
                                 18B6  4329 ;        Now check to see if there are dead ucbs waiting to be cleaned up
                                 18B6  4330 ;        and if noone is using the io database right now, we can delete them
                                 18B6  4331 ;        all in a flash.
                                 18B6  4332 ;
                                 18B6  4333
          50   00000000'GF   DE   18B6  4334 100$:   MOVAL   G^IOC$GL_MUTEX, R0                  ; Is anyone owning the io data
               FFFF 8F    60   B1   18BD  4335          CMPW    MTX$W_OWNCNT(R0), #-1              ; base?  If so, then we must
                          1A   12   18C2  4336          BNEQ    130$                               ; try again later.
               54    E8A8 CF   D0   18C4  4337          MOVL    GHB_L_DEADLINK, R4                 ; Good we own it then
                     E8A3 CF   D4   18C9  4338          CLRL    GHB_L_DEADLINK                     ; Reset list pointer
                          54   D5   18CD  4339 110$:    TSTL    R4                                 ; Any work to do?
                          0D   13   18CF  4340          BEQL    130$                               ; All done
                     55   54   D0   18D1  4341          MOVL    R4, R5                             ; UCB to delete
               54    0138 C5   D0   18D4  4342          MOVL    UCB$L_LT_DEADLINK(R5), R4          ; Save link to next one
                       FC0E   30   18D9  4343          BSBW    LT$DESTROYUCB                      ; Destroy this UCB
                          EF   11   18DC  4344          BRB     110$                               ; and go back for more
                                 18DE  4345
                                 18DE  4346 130$:
                                 18DE  4347
                                 18DE  4348 ;
                                 18DE  4349 ;        Transmit a configuration message if its time.  We tell all
                                 18DE  4350 ;        concentrators our node name, announcement string every
                                 18DE  4351 ;        so often so they know we exist.
                                 18DE  4352 ;
                                 18DE  4353
                     EAF8 CF   B7   18DE  4354          DECW    GHB_W_MULTIMR                      ; Time to transmit?
                          03   15   18E2  4355          BLEQ    150$                               ; Br if yes, do it
                        0147   31   18E4  4356 140$:    BRW     300$                               ; Else, still time to go
                                 18E7  4357
               52    EAF1 CF   D0   18E7  4358 150$:    MOVL    GHB_L_MULTIBFR, R2                 ; Get multicast buffer
                          0D   13   18EC  4359          BEQL    155$                               ; Br if none
                                 18EE  4360          ASSUME  GHB_STS_V_MULTI EQ 0
                  F1 E892 CF   E8   18EE  4361          BLBS    GHB_B_STATUS,140$                  ; Ignore, if already in progress
            4E A2    E88E CF   91   18F3  4362          CMPB    GHB_B_INCARN, -                    ; Has it been rebuilt?
                                 18F9  4363                  XMT_B_MC_INCARN(R2)                ;  if so, incarn has been
                          06   13   18F9  4364          BEQL    160$                               ;  bumped to say so
                       F037   30   18FB  4365 155$:    BSBW    LT$SETENTRY                        ; Go rebuild the message
                     E3 50    E9   18FE  4366          BLBC    R0, 140$                           ; Br if failure
          EAD2 CF    E885 CF   9B   1901  4367 160$:    MOVZBW  <XMT_B_MC_MULTIMR-XMT_B_MC_SET>-   ; Reset multicast timer
                                 1908  4368                   +GHB_T_MC_DATA,GHB_W_MULTIMR      ;     from LATCP set data
                          05   12   1908  4369          BNEQ    161$                               ; Br if value is okay
               EACB CF    0A   B0   190A  4370          MOVW    #LAT$C_MULTI_TIMER,GHB_W_MULTIMR   ; Reset timer
               52    EAC9 CF   D0   190F  4371 161$:    MOVL    GHB_L_MULTIBFR, R2                 ; Get multicast buffer
                                 1914  4372 ;
```

```
                                  1914  ; Adjust host rating
                                  1914  ;
51    54 A2    9E                 1914          MOVAB   XMT_B_MC_GRP_LEN(R2), R1        ; Compute address of node name:
      50   81    9A                1918          MOVZBL  (R1)+, R0                       ;   get count of groups
      51   50    C0                191B          ADDL    R0, R1                          ;   add in start address
      50   81    9A                191E          MOVZBL  (R1)+, R0                       ; Compute address of node desc.
      51   50    C0                1921          ADDL    R0, R1                          ;   add in start address
      50   81    9A                1924          MOVZBL  (R1)+, R0                       ; Calc start address of services
      51   50    C0                1927          ADDL    R0, R1                          ;   add in start address
      54   51    D0                192A          MOVL    R1, R4                          ; Save address of service count
      02   81    91                192D          CMPB    (R1)+, #2                       ; Do we have two service names?
           03    18                1930          BGEQ    163$                            ; Br if yes, process it
        00BE     31                1932          BRW     170$                            ; Else, skip it
           51    D6                1935  163$:   INCL    R1                              ; Skip rating byte
      50   81    9A                1937          MOVZBL  (R1)+, R0                       ; Calc start of first svc desc.
      51   50    C0                193A          ADDL    R0, R1                          ;   ...
      50   81    9A                193D          MOVZBL  (R1)+, R0                       ; Calc start of second svc rate
      51   50    C0                1940          ADDL    R0, R1                          ;   ...
                                  1943          ;
                                  1943          ;  Compute the service rating as follows:
                                  1943          ;
                                  1943          ;        cpu_idle_time * 255
                                  1943          ;       -------------------- * cpu_weight
                                  1943          ;       100 * multicast timer
                                  1943          ;
                                  1943          ;
                                  1943          ;   where:  cpu_weight is 1 fo  * 790, 1/2 for 785, 3/8 for 780, etc.
                                  1943          ;
                                  1943          ;
50    E82E DF    E835 CF    C3    1943          SUBL3   GHB_L_LASTCPU,@GHB_L_NULLCPU,R0 ; Calculate "idle time"
      E82A CF    E829 DF    D0    194B          MOVL    @GHB_L_NULLCPU,GHB_L_LASTCPU    ; Set new last cpu time
      56   00000000'GF      D0    1952          MOVL    G^EXE$GL_MP,R6                  ; Get ptr to MP code
                      17    13    1959          BEQL    164$                            ; Br if not there
           E821 CF          C3    195B          SUBL3   GHB_L_NULLSEC,-                 ; Else, compute secondary idle
      57   0000'C6                195F                   MPS$GL_NULLCPU(R6),R7           ;   time
           0000'C6          D0    1963          MOVL    MPS$GL_NULLCPU(R6),-            ; And set new secondary
           E816 CF                1967                   GHB_L_NULLSEC                   ;   null time
      57   57   FF 8F       78    196A          ASHL    #-1,R7,R7                       ; Get 50% of secondary null time
           50   57          C2    196F          SUBL    R7,R0                           ; Let secondary account for
                                  1972                                                  ;   only 50% additional power
      50   000000FF 8F       C4    1972  164$:   MULL    #255,R0                         ; Multiply by highest rating possibl
           53   01          9A    1979          MOVZBL  #1,R3                           ; Start with 1
                                  197C          CPUDISP <<790, EIGHT>,-                 ; Give 790 the highest rating
                                  197C                   <785, FOUR>,-                  ; Rate 785 as four
                                  197C                   <780, THREE>,-                 ; Rate 780 as three
                                  197C                   <750, TWO>,-                   ; Rate 750 as two
                                  197C                   <730, ONE>,-                   ; etc.
                                  197C                   <UV1, ONE>>                    ;
                                  19AE
           53   04          C0    19AE  EIGHT:  ADDL    #4,R3                           ; Rating will be highest
                53          D6    19B1  FOUR:   INCL    R3                              ; next rating
                53          D6    19B3  THREE:  INCL    R3                              ;   etc.
                53          D6    19B5  TWO:    INCL    R3                              ;
                                  19B7  ONE:
           50   53          C4    19B7          MULL    R3,R0                           ; Compute adjusted rating
      50   00000320 8F       C6    19BA          DIVL    #800,R0                         ; Divide by highest rating * %
      53   E7C5 CF          9A    19C1          MOVZBL  <XMT_B_MC_MULTIMR-XMT_B_MC_SET>- ; Get multicast timer
```

```
                        19C6  4430                  +GHB_T_MC_DATA,R3
                50   53  C6  19C6  4431        DIVL    R3,R0                          ; Normalize rating by time
                     0D  13  19C9  4432        BEQL    165$                           ; Br if zero rating
        000000FF 8F  50  D1  19CB  4433        CMPL    R0,#255                        ; Ridiculous rating ?
                     06  15  19D2  4434        BLEQ    167$                           ; Br if no, continue
            50   FE 8F  9A  19D4  4435        MOVZBL  #254,R0                        ; Else, set highest rating
                             19D8  4436                                               ;  let INCL up it to 255
                             19D8  4437 165$:  ;
                             19D8  4438        ;  Don't let rating fall to zero, unless the joblimit is reached
                             19D8  4439        ;  or the jobcnt if greater than or eqal to the joblim.
                             19D8  4440        ;
                     50  D6  19D8  4441        INCL    R0                             ; Else, set to minimum
00000000'GF  00000000'GF  B1  19DA  4442 167$:  CMPW    G^SYS$GW_IJOBCNT,G^SYS$GW_IJOBLIM ; Are we maxed out?
                     02  1F  19E5  4443        BLSSU   169$                           ; Br if no, continue
                     50  D4  19E7  4444        CLRL    R0                             ; Else, set to mimimum
                61   50  91  19E9  4445 169$:  CMPB    R0,(R1)                        ; New second service rating?
                     05  13  19EC  4446        BEQL    170$                           ; Br if no
                61   50  90  19EE  4447        MOVB    R0,(R1)                        ; Else, set new 2nd svc rating
                     0A  11  19F1  4448        BRB     175$
                             19F3  4449
        01 A4  00000000'GF  91  19F3  4450 170$:  CMPB    G^SYS$GW_IJOBCNT, 1(R4)        ; New number of interactive
                             19FB  4451                                               ;     users?
                     13  13  19FB  4452        BEQL    180$                           ; Br if no
        01 A4  00000000'GF  90  19FD  4453 175$:  MOVB    G^SYS$GW_IJOBCNT, 1(R4)        ; Use the interactive count
                             1A05  4454                                               ;  as the second rating
                   4E A2  96  1A05  4455        INCB    XMT_B_MC_INCARN(R2)            ; set new incarnation
                E779 CF  96  1A08  4456        INCB    GHB_B_INCARN                   ; no call LT$SETENTRY next time
             4F A2   08  8C  1A0C  4457        XORB    #XMT_CHFLG_M_RATE,XMT_B_MC_CHG_FLAG(R2) ; flag rate change
                             1A10  4458
          54    E760 CF  D0  1A10  4459 180$:  MOVL    GHB_L_FFI, R4                  ; Get FFI block address
                     17  13  1A15  4460        BEQL    300$                           ; Br if none
                             1A17  4461        SETBIT  #GHB_STS_V_MULTI,GHB_B_STATUS  ; Indicate that multicast is active
                53   52  D0  1A1D  4462        MOVL    R2,R3                          ; Copy CXB address
                E9B0 CF  7D  1A20  4463        MOVQ    GHB_Q_MULTIADD,-               ; Copy the destination address
                  28 A3      1A24  4464                CXB$Q_STATION(R3)              ;
                             1A26  4465        ASSUME  CXB$W_BCNT EQ CXB$W_BOFF+2
     14 A3      18 A3  D0  1A26  4466        MOVL    CXB$W_BOFF(R3),CXB$L_T_SAVE(R3) ; Save BOFF and BCNT
                             1A2B  4467
                10 B4  16  1A2B  4468        JSB     @FFI$L_XMIT(R4)                ; Start XMIT operation
                             1A2E  4469
                0FE0 8F  BA  1A2E  4470 300$:  POPR    #^M<R5,R6,R7,R8,R9,R10,R11>    ; Restore the regs
                             1A32  4471        ENBINT                                 ; Restore priority
                     05      1A35  4472        RSB
                             1A36  4473
                             1A36  4474        .DISABLE LOCAL_BLOCK
```

```
                        1A36  4476          .SBTTL  LT$XMIT  - Transmit a message
                        1A36  4477 ;++
                        1A36  4478 ; LT$XMIT - Transmit a message
                        1A36  4479 ;
                        1A36  4480 ; FUNCTIONAL DESCRIPTION:
                        1A36  4481 ;
                        1A36  4482 ;        If the CXB is outstanding then record the fact that there is
                        1A36  4483 ;        a waiting message.  If the CXB is not outstanding, then ready
                        1A36  4484 ;        it for transmission and send it to the other driver as a write.
                        1A36  4485 ;
                        1A36  4486 ;        Transmissions must make progress.  That is the sequence and acks
                        1A36  4487 ;        of the messages must change over time or we decide that the
                        1A36  4488 ;        circuit is really deadlocked and we take it down.  We are generous
                        1A36  4489 ;        with the counter for detecting this deadlock.
                        1A36  4490 ;
                        1A36  4491 ; Input:
                        1A36  4492 ;        R4 = CSB address
                        1A36  4493 ;
                        1A36  4494 ; Output:
                        1A36  4495 ;        R0-R5 clobbered
                        1A36  4496 ;-
                        1A36  4497
                        1A36  4498          .ENABLE LOCAL_BLOCK
                        1A36  4499
                        1A36  4500 LT$XMIT:
53   34 B4    0F        1A36  4501          REMQUE  @CSB_Q_XBUFQ(R4), R3     ; Remove the buffer from queue
        06    1C        1A3A  4502          BVC     5$                       ; We had better have a buffer here.
                        1A3C  4503          DEBUG                            ; Fatal error, no buffer
     59 A4    96        1A42  4504 5$:      INCB    CSB_B_XMTCNT(R4)         ; One more buffer to transmit
     62 A4    96        1A45  4505          INCB    CSB_B_XSEQ(R4)           ; Next message to send
     5C A4    7D        1A48  4506          MOVQ    CSB_T_CIRCHDR(R4),-      ; Set the circuit header for the
     48 A3              1A4B  4507                  XMT_T_DATA(R3)           ; message
50   48 A3    9E        1A4D  4508          MOVAB   XMT_T_DATA(R3), R0       ; Start of data to send
1C A3   50    C2        1A51  4509          SUBL    R0, CXB$L_T_ENDADR(R3)   ; Make length from end address
     1C A3    B0        1A55  4510          MOVW    CXB$L_T_ENDADR(R3),-     ; Copy length to CXB format
     1A A3              1A58  4511                  CXB$W_BCNT(R3)
40 B4   63    0E        1A5A  4512          INSQUE  (R3), @CSB_Q_XWAITQ+4(R4) ; Queue to tail of waiting buffers
     58 A4    95        1A5E  4513          TSTB    CSB_B_XMTBSY(R4)         ; Transmitter busy?
        3D    13        1A61  4514          BEQL    60$                      ; Br if no, start transmitting
     58 A4    96        1A63  4515 10$:     INCB    CSB_B_XMTBSY(R4)         ; Else, count another transmit
                        1A66  4516                                           ;  we just placed one on the wait queue
              05        1A66  4517 20$:     RSB
                        1A67  4518
                        1A67  4519 ;
                        1A67  4520 ;        Transmit just the one buffer, not all the ones waiting.
                        1A67  4521 ;
                        1A67  4522 LT$XMITONE:
53   34 B4    0F        1A67  4523          REMQUE  @CSB_Q_XBUFQ(R4), R3     ; Get the one buffer of interest
        06    1C        1A6B  4524          BVC     30$                      ; Got it
                        1A6D  4525          DEBUG                            ; No buffer for this transmit
     59 A4    96        1A73  4526 30$:     INCB    CSB_B_XMTCNT(R4)         ; Count one more on wait queue
     62 A4    96        1A76  4527          INCB    CSB_B_XSEQ(R4)           ; Next message to send
     5C A4    7D        1A79  4528          MOVQ    CSB_T_CIRCHDR(R4),-      ; Set the circuit header for the
     48 A3              1A7C  4529                  XMT_T_DATA(R3)           ; message
50   48 A3    9E        1A7E  4530          MOVAB   XMT_T_DATA(R3), R0       ; Start of data to send
1C A3   50    C2        1A82  4531          SUBL    R0, CXB$L_T_ENDADR(R3)   ; Make length from end address
     1C A3    B0        1A86  4532          MOVW    CXB$L_T_ENDADR(R3),-     ; Copy length to CXB format
```

```
          1A A3        1A89  4533                    CXB$W_BCNT(R3)
          58 A4   95   1A8B  4534          TSTB     CSB_B_XMTBSY(R4)          ; Are we busy??
             05   12   1A8E  4535          BNEQ     40$                       ; Br if yes, so carry on
          58 A4   96   1A90  4536          INCB     CSB_B_XMTBSY(R4)          ; Just one buffer
             25   11   1A93  4537          BRB      90$                       ; Go
                      1A95  4538
       40 B4  63   0E   1A95  4539 40$:     INSQUE   (R3), @CSB_Q_XWAITQ+4(R4) ; Place it on the queue
             C8   11   1A99  4540          BRB      10$                       ; And transmit later
                      1A9B  4541
                      1A9B  4542 LT$REXMIT:                                  ; Entry to start a retransmit
          58 A4   95   1A9B  4543          TSTB     CSB_B_XMTBSY(R4)          ; Is the transmit busy??
             C6   12   1A9E  4544          BNEQ     20$                       ; Br if yes, just leave
    58 A4 59 A4   90   1AA0  4545 60$:     MOVB     CSB_B_XMTCNT(R4), -       ; Start a series of retransmits
                      1AA5  4546                   CSB_B_XMTBSY(R4)          ;
                      1AA5  4547
                      1AA5  4548
                      1AA5  4549 ;;**      debug code
          59 A4   91  1AA5  4550          CMPB     CSB_B_XMTCNT(R4),-        ; All buffers on the transmit queue?
             02       1AA8  4551                   #LAT$C_XMT_BUFFERS        ;
             09   19  1AA9  4552          BLSS     80$                       ; Nope
             06   13  1AAB  4553          BEQL     70$                       ; Yes all of them
                      1AAD  4554          DEBUG                              ; More than all of them
                   01 1AB3  4555 70$:     NOP
                      1AB4  4556 80$:
                      1AB4  4557 ;;**      debug code
                      1AB4  4558
                      1AB4  4559 LT$XMTGO:                                   ; Enter to transmit for waiters
       53 3C B4   0F  1AB4  4560          REMQUE   @CSB_Q_XWAITQ(R4), R3     ; Obtain the next waiting buffer
             AC   1D  1AB8  4561          BVS      20$                       ; Br if no buffer to transmit
          56 A4   B1  1ABA  4562 90$:     CMPW     CSB_W_PROGSEQ(R4),-       ; Are we making progre s?
          62 A4       1ABD  4563                   CSB_W_XSEQ(R4)            ; are the seq and ack fields changing?
             1F   12  1ABF  4564          BNEQ     100$                      ; Br if yes, splendid
          54 A4   B7  1AC1  4565          DECW     CSB_W_PROGRESS(R4)        ; Nope, give them some time then
             23   14  1AC4  4566          BGTR     110$                      ; We have more time then
       34 A4  63   0E  1AC6  4567          INSQUE   (R3), CSB_Q_XBUFQ(R4)     ; Put the buffer back into the CSB
                      1ACA  4568          INC_CTR  GHB$L_CIRCDOWN+GHB_AREA   ; Count a circuit timeout
             58   DD  1AD4  4569          PUSHL    R8                        ; No more time.  Clobber the circuit
       58    54   D0  1AD6  4570          MOVL     R4, R8                    ; Since it appears to be deadlocked.
          FC15   30   1AD9  4571          BSBW     LT$CIRCDEAD               ; and clean it all up
             58 8ED0  1ADC  4572          POPL     R8                        ;
                   05 1ADF  4573          RSB
                      1AE0  4574
          62 A4   B0  1AE0  4575 100$:     MOVW     CSB_W_XSEQ(R4),-          ; Reset the current sequence and ack
          56 A4       1AE3  4576                   CSB_W_PROGSEQ(R4)         ; to check against
             14   B0  1AE5  4577          MOVW     #LAT$C_PROGRESS,-         ; and reset the progress counter
          54 A4       1AE7  4578                   CSB_W_PROGRESS(R4)        ;
                      1AE9  4579
                      1AE9  4580 110$:
                      1AE9  4581 ;
                      1AE9  4582 ;         Build the CXB to transmit
                      1AE9  4583 ;
    55 E087 CF   D0   1AE9  4584          MOVL     GHB_L_FFI, R5             ; Get the FFI block address
             2A   13  1AEE  4585          BEQL     150$                      ; Br if none
       51    5A A4   9A  1AF0  4586          MOVZBL   CSB_B_XCXB_INX(R4), R1    ; Get XMIT CXB index slot
    44 A441  53   D0  1AF4  4587          MOVL     R3, CSB_L_XCXB(R4)[R1]    ; Save address of CXB
             51   96  1AF9  4588          INCB     R1                        ; Skip to next slot
       51    FE 8F   8A  1AFB  4589          BICB     #^C<LAT$C_XMT_BUFFERS-1>,R1 ; Modulo maximum
```

```
5A A4    51   90  1AFF  4590          MOVB    R1, CSB_B_XCXB_INX(R4)  ; Save for next time
         54   DD  1B03  4591          PUSHL   R4                      ; Save R4
   54    55   D0  1B05  4592          MOVL    R5,R4                   ; Else, copy the FFI block address
                 1B08  4593          ASSUME  CXB$W_BCNT EQ CXB$W_BOFF+2
14 A3   18 A3   D0  1B08  4594          MOVL    CXB$W_BOFF(R3),CXB$L_T_SAVE(R3) ; Save BOFF and BCNT
                 1B0D  4595                                                  ;  progress
                 1B0D  4596          .IF DEFINED LT_HISTORY
                 1B0D  4597 ;
                 1B0D  4598 ;        Save a history of transmits
                 1B0D  4599 ;
50      48 A3    9E  1B0D  4600          MOVAB   XMT_B_FLAG(R3), R0      ; Address of data to save
         EBE3    30  1B11  4601          BSBW    LT$HISTORY              ; Save the data
                 1B14  4602
                 1B14  4603          .ENDC   ;; DEFINED LT_HISTORY
                 1B14  4604
      10 B4      16  1B14  4605          JSB     @FFI$L_XMIT(R4)         ; Start the XMIT operation
         54 8ED0     1B17  4606          POPL    R4                      ; Restore R4
               05    1B1A  4607 150$:    RSB                             ; Return to caller
                 1B1B  4608
                 1B1B  4609          .DISABLE LOCAL_BLOCK
                 1B1B  4610
```

```
                        1B1B  4612          .SBTTL  LT$XMT_FFIDONE  - FFI Transmit done routine
                        1B1B  4613 ;++
                        1B1B  4614 ; LT$XMT_FFIDONE - FFI Transmit completion routine
                        1B1B  4615 ;
                        1B1B  4616 ; FUNCTIONAL DESCRIPTION:
                        1B1B  4617 ;
                        1B1B  4618 ;        Post routine for a transmit.  Called directly from datalink
                        1B1B  4619 ;        driver with CXB address in R3.  We have r0 - r5 to work with.
                        1B1B  4620 ;
                        1B1B  4621 ; INPUTS:
                        1B1B  4622 ;        R0 =  Status of XMIT request
                        1B1B  4623 ;        R3 = CXB address
                        1B1B  4624 ;        R4 = FFI block address
                        1B1B  4625 ;
                        1B1B  4626 ;        IPL = SYNCH
                        1B1B  4627 ;
                        1B1B  4628 ; OUTPUTS:
                        1B1B  4629 ;        R1,R2,R4,R5 clobbered
                        1B1B  4630 ;--
                        1B1B  4631          ASSUME  IPL$_SYNCH EQ LAT$C_IPL
                        1B1B  4632 LT$XMT_FFIDONE:
                 36  3B 1B1B  4633          PUSHR   #^M<R1,R2,R4,R5>        ; Save registers
                        1B1D  4634          ASSUME  CXB$W_BCNT EQ CXB$W_BOFF+2
        18 A3    14 A3  D0 1B1D  4635          MOVL    CXB$L_T_SAVE(R3),CXB$W_BOFF(R3) ; Restore BOFF and BCNT
                        1B22  4636          INC_CTR GHB$L_XCOUNT+GHB_AREA   ; Count one more buffer away
                 0F 50  E8 1B22  4637          BLBS    R0, 20$                 ; Br if no error
         E618 CF    50  3C 1B2F  4638          MOVZWL  R0,GHB$L_XERRCOD+GHB_AREA ; Save error code
                        1B34  4639          INC_CTR GHB$L_XERR+GHB_AREA     ; Count one more error, but ignore it
             4C 0B A3  E8 1B3E  4640 20$:    BLBS    CXB$B_CODE(R3),110$     ; Br if multicast buffer
          54    0C A3  D0 1B42  4641          MOVL    CXB$L_T_CSB(R3),R4      ; Get CSB address
                 08  12 1B46  4642          BNEQ    40$                     ; Got it, continue
             50  53  D0 1B48  4643          MOVL    R3,R0                   ; Toss this CXB then since the
                0052  30 1B4B  4644          BSBW    LT$DEALPOOL             ;   CSB seems to be gone.
                 3B  11 1B4E  4645          BRB     70$                     ; Leave quietly
                        1B50  4646
             50  01  9A 1B50  4647 40$:     MOVZBL  #LAT$C_XMT_BUFFERS-1,R0 ; Get number of CXBs
        44 A440  53  D1 1B53  4648 45$:     CMPL    R3,CSB_L_XCXB(R4)[R0]   ; Same as we sent?
                 09  13 1B58  4649          BEQL    50$                     ; Br if yes, remove it
              F6 50  F4 1B5A  4650          SOBGEQ  R0,45$                  ; Loop if more
                        1B5D  4651          DEBUG                           ;;&& oops
            44 A440  D4 1B63  4652 50$:     CLRL    CSB_L_XCXB(R4)[R0]      ; Zero entry
                        1B67  4653          INC_CTR CSB_Z_LCB+LCB_L_MSG_XMT(R4) ; Count one more transmit
         40 B4    63  0E 1B6F  4654          INSQUE  (R3),@CSB_Q_XWAITQ+4(R4) ; Queue to tail for retransmit
            58 A4  97  1B73  4655          DECB    CSB_B_XMTBSY(R4)        ; Any more to transmit??
                 05  13 1B76  4656          BEQL    60$                     ; Nope all done
              FF39  30 1B78  4657          BSBW    LT$XMTGO                ; Transmit for waiter
                 0E  11 1B7B  4658          BRB     70$                     ; Not done transmitting all yet
                        1B7D  4659 60$:
                        1B7D  4660 ;
                        1B7D  4661 ; If all UCBs are gone and we have transmitted a stop message last,
                        1B7D  4662 ; then we can make this CSB go away.
                        1B7D  4663 ;
                        1B7D  4664
            2D A4  95 1B7D  4665          TSTB    CSB_B_REFC(R4)          ; Are all the UCB's gone?
                 09  12 1B80  4666          BNEQ    70$                     ;
            5C A4  91 1B82  4667          CMPB    CSB_B_FLAG(R4),-        ; Is the circuit dead?
                 08        1B85  4668                  #MTYP_C_HALT@FLAG_V_MTYPE
```

```
  03   12  1B86  4669          BNEQ    70$                         ; Br if not
F860   30  1B88  4670          BSBW    LT$DEALOCSB                 ; Get rid of the circuit block
  36   BA  1B8B  4671  70$:    POPR    #^M<R1,R2,R4,R5>            ; Restore registers
       05  1B8D  4672          RSB
           1B8E  4673          ;
           1B8E  4674          ; Multicast buffer completed
           1B8E  4675          ;
           1B8E  4676  110$:   CLRBIT  #GHB_STS_V_MULTI,-          ; Multicast is not busy now
           1B8E  4677                   GHB_B_STATUS
   F5  11  1B94  4678          BRB     70$                         ; And exit
           1B96  4679
```

```
                        1B96          .SBTTL  LT$FFIPOSTDONE  - Garbage transmit posting routine
                        1B96  ;++
                        1B96  ; LT$FFIPOSTDONE - Garbage transmit posting routine
                        1B96  ;
                        1B96  ; FUNCTIONAL DESCRIPTION:
                        1B96  ;
                        1B96  ;       Final post routine to dry up a CXB I/O when we are done
                        1B96  ;       with the channel.  We simply deallocate any transmit buffer returned.
                        1B96  ;
                        1B96  ; INPUTS:
                        1B96  ;       R3 = CXB address
                        1B96  ;       R4 = FFI block address
                        1B96  ;
                        1B96  ; OUTPUTS:
                        1B96  ;       R0 clobbered
                        1B96  ;--
                        1B96
                        1B96  LT$FFIPOSTDONE:
        36   BB         1B96          PUSHR   #^M<R1,R2,R4,R5>        ; Save registers
   50   53   D0         1B98          MOVL    R3,R0                   ; Copy CXB address
        03   10         1B9B          BSBB    LT$DEALPOOL             ; Drop the buffer
        36   BA         1B9D          POPR    #^M<R1,R2,R4,R5>        ; Restore registers
                        1B9F  LT$DROPCXB:                             ; Drop all RECEIVE CXBs
             05         1B9F          RSB
                        1BA0
                        1BA0  LT$DEALPOOL:
00000000'GF  17         1BA0          JMP     G^EXE$DEANONPAGED       ; Deallocate the FFI block
                        1BA6
                        1BA6  LT$END:                                 ; End of driver
                        1BA6
                        1BA6          .END
```

```
$$$                             = 00000020 R     02
$$BASE                          = 00000001
$$DISPL                         = 0000000A
$$GENSW                         = 00000001
$$HIGH                          = 00000009
$$LIMIT                         = 00000008
$$LOW                           = 00000001
$$MNSW                          = 00000001
$$MXSW                          = 00000001
$$OP                            = 00000002
AT$_NULL                          ********   X   02
BIT...                          = 00000004
BUG$_UNSUPRTCPU                   ********   X   03
BUILD_TQE                         000008B2 R     03
CLASS_DDT                       = 00000010
CLASS_DISCONNECT                = 00000018
CLASS_GETNXT                    = 00000000
CLASS_PUTNXT                    = 00000004
CLASS_READERROR                 = 00000014
CLASS_SETUP_UCB                 = 00000008
CRB$L_INTD                      = 00000024
CRB$W_REFC                      = 0000000C
CREDIT_CHECK                      000010AE R     03
CSB_B_FLAG                        0000005C  G
CSB_B_INX                         00000068  G
CSB_B_LOCIDN                      00000060  G
CSB_B_LOCIDS                      00000061  G
CSB_B_MAX_SLOTS                   00000069  G
CSB_B_NUM_SLOTS                   0000005D  G
CSB_B_RACK                        00000065  G
CSB_B_REFC                        0000002D  G
CSB_B_REMIDN                      0000005E  G
CSB_B_REMIDS                      0000005F  G
CSB_B_RSEQ                        00000064  G
CSB_B_SERVER                      0000001C  G
CSB_B_SPARE                       0000005B  G
CSB_B_STATE                       0000000B  G
CSB_B_TYPE                        0000000A  G
CSB_B_XACK                        00000063  G
CSB_B_XCXB_INX                    0000005A  G
CSB_B_XMTBSY                      00000058  G
CSB_B_XMTCNT                      00000059  G
CSB_B_XSEQ                        00000062  G
CSB_C_FIXLENGTH                   0000006C  G
CSB_C_STATE_HALT                = 00000000  G
CSB_C_STATE_RUN                 = 00000002  G
CSB_C_STATE_START               = 00000001  G
CSB_L_STOPREASON                  0000004C  G
CSB_L_UCBLST                      0000006C  G
CSB_L_XCXB                        00000044  G
CSB_Q_XBUFQ                       00000034  G
CSB_Q_XWAITQ                      0000003C  G
CSB_S_CIRCHDR                   = 00000008  G
CSB_T_CIRCHDR                     0000005C  G
CSB_T_SERVER                      0000001D  G
CSB_W_LOCID                       00000060  G
CSB_W_MAX_MSGSIZ                  0000006A  G
CSB_W_PROGRESS                    00000054  G
CSB_W_PROGSEQ                     00000056  G
CSB_W_REMID                       0000005E  G
CSB_W_RSEQ                        00000064  G
CSB_W_SIZE                        00000008  G
CSB_W_TIMEOUT                     00000050  G
CSB_W_TIMRESET                    00000066  G
CSB_W_XMTTMO                      00000052  G
CSB_W_XSEQ                        00000062  G
CSB_Z_DST                         0000002E  G
CSB_Z_LCB                         0000000C  G
CXB$B_CODE                      = 0000000B
CXB$B_TYPE                      = 0000000A
CXB$C_HEADER                    = 00000048
CXB$L_END_ACTION                = 0000001C
CXB$L_IRP                       = 00000014
CXB$L_T_CSB                       0000000C  G
CXB$L_T_ENDADR                    0000001C  G
CXB$L_T_SAVE                      00000014  G
CXB$Q_STATION                   = 00000028
CXB$W_BCNT                      = 0000001A
CXB$W_BOFF                      = 00000018
CXB$W_LENGTH                    = 0000000C
CXB$W_OFFSET                    = 0000000E
CXB$W_SIZE                      = 00000008
DC$_TEPM                          ********   X   02
DDB$L_DDT                       = 0000000C
DDB$L_UCB                       = 00000004
DEFREASN                          00001719 R     03
DEV$M_AVL                       = 00040000
DEV$M_CCL                       = 00000002
DEV$M_IDV                       = 04000000
DEV$M_NNM                       = 00000200
DEV$M_ODV                       = 08000000
DEV$M_REC                       = 00000001
DEV$M_TRM                       = 00000004
DPT$B_FLAGS                     = 0000000D
DPT$C_LENGTH                    = 00000038
DPT$C_VERSION                   = 00000004
DPT$INITAB                        00000038 R     02
DPT$M_NOUNLOAD                  = 00000004
DPT$REINITAB                      000000D3 R     02
DPT$TAB                           00000000 R     02
DPT$W_VECTOR                    = 0000001E
DYN$C_BUFIO                     = 00000013
DYN$C_CDB                       = 00000033
DYN$C_CRB                       = 00000005
DYN$C_CXB                       = 0000001B
DYN$C_DDB                       = 00000006
DYN$C_DPT                       = 0000001E
DYN$C_ORB                       = 00000049
DYN$C_TQE                       = 0000000F
DYN$C_UCB                       = 00000010
ECHO_ERROR                        00000FED R     03
EIGHT                             000019AE R     03
EXE$ALLOCTQE                      ********   X   03
EXE$ALONONPAGED                   ********   X   03
```

```
EXE$AL_QENOREPT          ********  X   03
EXE$DEANONPAGED          ********  X   03
EXE$FORK                 ********  X   03
EXE$GB_CPUDATA           ********  X   03
EXE$GB_CPUTYPE           ********  X   03
EXE$GL_MP                ********  X   03
EXE$GQ_SYSTIME           ********  X   03
EXE$INSTIMQ              ********  X   03
FFI$B_SPARE            = 0000000B
FFI$B_TYPE             = 0000000A
FFI$C_LENGTH           = 0000003E
FFI$L_BL               = 00000004
FFI$L_CTX_DL           = 0000000C
FFI$L_DL_UCB           = 00000034
FFI$L_ERROR            = 0000001C
FFI$L_FL               = 00000000
FFI$L_PID              = 00000038
FFI$L_RECV_DONE        = 00000018
FFI$L_SHUT_DONE        = 00000020
FFI$L_SPARE0           = 00000024
FFI$L_SPARE1           = 00000028
FFI$L_SPARE2           = 0000002C
FFI$L_SPARE3           = 00000030
FFI$L_XMIT             = 00000010
FFI$L_XMIT_DONE        = 00000014
FFI$W_CHAN             = 0000003C
FFI$W_SIZE             = 00000008
FKB$B_FIPL             = 0000000B
FLAG_M_MASTER          = 00000002  G
FLAG_M_MTYPE           = 000000FC  G
FLAG_M_RRF             = 00000001  G
FLAG_S_MASTER          = 00000001  G
FLAG_S_MTYPE           = 00000006  G
FLAG_S_RRF             = 00000001  G
FLAG_V_MASTER          = 00000001  G
FLAG_V_MTYPE           = 00000002  G
FLAG_V_RRF             = 00000000  G
FOUR                     000019B1 R     03
FUNCTAB_LEN            = 00000000
GET_MORE                 00001162 R     03
GHB$B_LATECO           = 00000003
GHB$B_LATVERSION       = 00000002
GHB$K_IDLEN            = 00000040
GHB$K_NAMELEN          = 00000010
GHB$L_CIRCDOWN         = 00000048
GHB$L_CSBLST           = 00000020
GHB$L_DUPLMSG          = 00000034
GHB$L_HISTORY          = 00000008
GHB$L_NODE             = 0000001C
GHB$L_NOXBFR           = 00000058
GHB$L_PROTOCOL         = 0000004C
GHB$L_PROTOMASK        = 00000050
GHB$L_RCOUNT           = 0000002C
GHB$L_RESOURCE         = 00000054
GHB$L_RETRANS          = 00000044
GHB$L_SETENTRY         = 00000004
GHB$L_SHUTENTRY        = 0000000C
GHB$L_STRENTRY         = 00000010
GHB$L_TIM_ACT          = 00000018
GHB$L_UCB              = 00000014
GHB$L_XCOUNT           = 00000038
GHB$L_XERR             = 0000003C
GHB$L_XERRCOD          = 00000040
GHB$Q_OLD_CSBS         = 00000024
GHB$S_COMM_AREA        = 00000060
GHB$T_CSBCTR           = 0000000C
GHB$V_BADCREDITS       = 00000008
GHB$V_CSBINVALID       = 00000003
GHB$V_CSBRANGE         = 00000002
GHB$V_CSBSTALE         = 00000004
GHB$V_CSBZERO          = 00000001
GHB$V_HALT             = 00000005
GHB$V_INVALIDLOCID     = 00000007
GHB$V_INVALIDREMID     = 00000006
GHB$V_INVALIDSEQ       = 0000000A
GHB$V_REPCREATE        = 00000009
GHB$V_REPDISC          = 0000000B
GHB$V_START            = 00000000
GHB$W_VMSVERSION       = 00000000
GHB_AREA                 0000010C R     03
GHB_B_INCARN             00000185 R     03
GHB_B_LOC_CCHK           000000B8 R     03
GHB_B_OLD_CSBCNT         00000186 R     03
GHB_B_RATE               00000187 R     03
GHB_B_STATUS             00000184 R     03
GHB_C_STRT_SLOTL       = 00000005
GHB_G_TQE                000000DC R     03
GHB_L_AREA               000004A4 R     03
GHB_L_CSBTABLE           00000038 R     03
GHB_L_DEADLINK           00000170 R     03
GHB_L_FFI                00000174 R     03
GHB_L_LASTCPU            0000017C R     03
GHB_L_MULTIBFR           000003DC R     03
GHB_L_NULLCPU            00000178 R     03
GHB_L_NULLSEC            00000180 R     03
GHB_L_UCBO               0000016C R     03
GHB_Q_MULTIADD           000003D4 R     03
GHB_STS_M_ACTIVE       = 00000002  G
GHB_STS_M_MULTI        = 00000001  G
GHB_STS_M_SHUT         = 00000004  G
GHB_STS_M_TQE          = 00000008  G
GHB_STS_S_ACTIVE       = 00000001  G
GHB_STS_S_MULTI        = 00000001  G
GHB_STS_S_SHUT         = 00000001  G
GHB_STS_S_TQE          = 00000001  G
GHB_STS_V_ACTIVE       = 00000001  G
GHB_STS_V_MULTI        = 00000000  G
GHB_STS_V_SHUT         = 00000002  G
GHB_STS_V_TQE          = 00000003  G
GHB_T_MC_DATA            0000018A R     03
GHB_T_MC_MSG             000003E0 R     03
GHB_T_STRT_MSG           000003EC R     03
GHB_T_STRT_SLOT          0000049D R     03
GHB_T_STRT_VAR           000003F8 R     03
```

```
GHB_W_MC_SIZE                   00000188 R     03
GHB_W_MULTIMR                   000003DA R     03
GHB_W_STRT_LEN                  000003EA R     03
GOT_A_LOT                       00001173 R     03
HBF_L_BUFEND                  = 00000004
HBF_L_NEXT                    = 00000000
HBF_Z_DATA                    = 0000000C
INB_B_SPARE                     000000B  G
INB_B_TYPE                      0000000A G
INB_C_LENGTH                    0000000C G
INB_L_SPARE                     00000004 G
INB_W_BCNT                      00000002 G
INB_W_BOFF                      00000000 G
INB_W_SIZE                      00000008 G
IOC$CLONE_UCB                   ********   X   03
IOC$DELETE_UCB                  ********   X   03
IOC$GL_MUTEX                    ********   X   03
IOC$MNTVER                      ********   X   03
IOC$RETURN                      ********   X   03
IPL$_ASTDEL                   = 00000002
IPL$_QUEUEAST                 = 00000006
IPL$_SYNCH                    = 00000008
IPL$_TIMER                    = 00000008
LAT$C_CIR_TIMER               = 00000050
LAT$C_CUR_ECO                 = 00000000
LAT$C_CUR_VER                 = 00000005
LAT$C_GRP_LEN                 = 00000020
LAT$C_HI_VER                  = 00000005
LAT$C_HOST_TIMER              = 00000002
LAT$C_IPL                     = 00000008
LAT$C_KEEP_CSB                = 0000000A
LAT$C_LOC_LEN                 = 00000040
LAT$C_LO_VER                  = 00000005
LAT$C_MAX_CSBS                = 00000020
LAT$C_MAX_MSGSIZ              = 000005DC
LAT$C_MAX_RCRED               = 00000001
LAT$C_MAX_SERVICES            = 00000006
LAT$C_MAX_SLOTS               = 00000040
LAT$C_MAX_SLOTSIZ             = 000000FF
LAT$C_MULTIADD1               = 00000009
LAT$C_MULTIADD2               = 0000002B
LAT$C_MULTIADD3               = 00000F00
LAT$C_MULTI_TIMER             = 000000A
LAT$C_PROGRESS                = 00000014
LAT$C_SHUTDELAY               = 004C4B40
LAT$C_SVC_LEN                 = 00000008
LAT$C_UCB_BUFSIZ              = 00000050
LAT$C_UCB_SLOTSIZ             = 00000004
LAT$C_UCB_TIMOUT              = 00000002
LAT$C_VMS                     = 00000003
LAT$C_VMS_VER                 = 00000004
LAT$C_XMT_BUFFERS             = 00000002
LAT_C_CRSN_DISC               = 00000005
LAT_C_CRSN_DONE               = 00000001
LAT_C_CRSN_IVMCI              = 00000004
LAT_C_CRSN_IVMSG              = 00000002
LAT_C_CRSN_IVSCI              = 00000003
LAT_C_CRSN_PROGRESS           = 00000006
LAT_C_CRSN_RESOURCE           = 00000009
LAT_C_CRSN_RETRANS            = 00000008
LAT_C_CRSN_TIMRANG            = 0000000A
LAT_C_CRSN_TMO                = 00000007
LAT_C_RSN_XXX                 = 00000000
LCB_B_INV_MSG                 = 0000000E
LCB_B_INV_SLOT                = 0000000F
LCB_C_LENGTH                  = 00000010
LCB_L_MSG_RCV                 = 00000004
LCB_L_MSG_REXMT               = 00000008
LCB_L_MSG_XMT                 = 00000000
LCB_W_SEQ_ERR                 = 0000000C
LOAD_CREDITS                    00000F66 R     03
LOOP_ABORT                      00000F18 R     03
LOOP_EXIT                       0000120A R     03
LT$ABORT                        00000C3A R     03
LT$ALC_UCB                      00001512 R     03
LT$ASYNCERR                     00000934 R     03
LT$CIRCDEAD                     000016F1 R     03
LT$CREATECSB                    00001239 R     03
LT$CREATEUCB                    000014D5 R     03
LT$CTRL_INIT                    00000644 R     03
LT$DDT                          00000000 RG    03
LT$DEALOCSB                     000013EB R     03
LT$DEALPOOL                     00001BA0 R     03
LT$DESTROYUCB                   000014EA R     03
LT$DISCONNECT                   00001672 R     03
LT$DPT                          00000000 RG    02
LT$DROPCXB                      00001B9F R     03
LT$END                          00001BA6 R     03
LT$FFIPOSTDONE                  00001B96 R     03
LT$FFI_RCV_MSG                  00000CEC R     03
LT$FLOW_CHANGE                  00000C48 R     03
LT$FLUSH_DATA                   00000C29 R     03
LT$FORCE_HALT                   00000E1C R     03
LT$GETFFI                       000007B0 R     03
LT$HANGUP_UCB                   0000165B R     03
LT$HANGUP_UCB_NOW               0000164E R     03
LT$HISTORY                      000006F7 R     03
LT$KILLUCB                      0000169D R     03
LT$NEW_CIRCUIT                  0000129C R     03
LT$NULL                         000004E4 R     03
LT$RCV_HALT_MSG                 00000E32 R     03
LT$RCV_RUN_MSG                  00000E45 R     03
LT$RCV_START_MSG                00000E38 R     03
LT$REPLY_ALT                    00001226 R     03
LT$REPLY_DONE                   0000122C R     03
LT$REPLY_MSG                    00001222 R     03
LT$REPLY_REXMIT                 00001231 R     03
LT$REXMIT                       00001A9B R     03
LT$SETENTRY                     00000935 R     03
LT$SET_TIMER                    000017DF R     03
LT$SHUTENTRY                    00000805 R     03
LT$SHUT_DONE                    00000923 RG    03
LT$SIDELINEUCB                  000016D0 R     03
LT$STARTIO                      00000AAE R     03
```

```
LT$STATE_ENTER_RUN          00000E76 R    03
LT$STATE_HALT               00000E2C R    03
LT$STATE_RUN                00000E7E R    03
LT$STATE_START              00000E52 R    03
LT$STOPCIRC                 00001732 R    03
LT$STRENTRY                 0000072D R    03
LT$TICK                     00001853 R    03
LT$UCB_INIT                 00000A6B R    03
LT$UNIT_INIT                000009FF R    03
LT$VEC                      0000[illegible]8 R    03
LT$VECEND                   0000[illegible]4E0 R    03
LT$VECTOR                   000004A8 R    03
LT$XMIT                     00001A36 R    03
LT$XMITONE                  00001A67 R    03
LT$XMTGO                    00001AB4 R    03
LT$XMT_FFIDONE              00001B1B R    03
LT$XOFF                     00000C9E R    03
LT$XON                      00000C4E R    03
LT_HISTORY                = 00000001
LT_RSN_BADCREDITS           000005D5 R    03
LT_RSN_CSBINVALID           0000054D R    03
LT_RSN_CSBRANGE             0000052D R    03
LT_RSN_CSBSTALE             00000568 R    03
LT_RSN_CSBZERO              00000515 R    03
LT_RSN_HALT                 00000584 R    03
LT_RSN_INVALIDLOCID         0000059F R    03
LT_RSN_I VALIDREMID         000005BA R    03
LT_RSN_RL CREATE            000005F8 R    03
LT_RSN_REPDISC              0000061C R    03
LT_RSN_START                000004E5 R    03
MMG$GL_MAXPFN               ********   X    03
MPS$GL_NULLCPU              ********   X    03
MTX$W_OWNCNT              = 00000000
MTYP_C_CONF               = 0000000A  G
MTYP_C_HALT               = 00000002  G
MTYP_C_RUN                = 00000000  G
MTYP_C_START              = 00000001  G
NEXT_SLOT                   00000FDA R    03
NEXT_SLOTO                  00000F63 R    03
ONE                         000019B7 R    03
ONE_LESS                    0000101F R    03
ORB$B_FLAGS               = 0000000B
ORB$L_OWNER               = 00000000
ORB$M_PROT_16             = 00000001
ORB$W_PROT                = 00000018
PCB$L_PHD                 = 0000006C
PHD$L_CPUTIM              = 00000038
PORT_ABORT                = 00000020
PORT_DISCONNECT           = 00000004
PORT_LENGTH               = 00000038
PORT_SET_LINE             = 00000008
PORT_STARTIO              = 00000000
 ORT_XOFF                 = 00000014
PORT_XON                  = 00000010
PR$_IPL                   = 00000012
PR$_SID_TYP730            = 00000003
PR$_SID_TYP750            = 00000002
PR$_SID_TYP780            = 00000001
PR$_SID_TYP785            = 00000009
PR$_SID_TYP790            = 00000004
PR$_SID_TYPMAX            = 000000J8
PR$_SID_TYPUV1            = 00000007
RCV_B_ACK                   00000007  G
RCV_B_DSTIDN                00000002  G
RCV_B_DSTIDS                00000003  G
RCV_B_FLAG                  00000000  G
RCV_B_NUM_SLOTS             00000001  G
RCV_B_SEQ                   00000006  G
RCV_B_SRCIDN                00000004  G
RCV_B_SRCIDS                00000005  G
RCV_T_DATA                  00000000  G
RCV_T_MDATA                 00000008  G
RCV_W_DSTID                 00000002  G
RCV_W_SRCID                 00000004  G
RSN$_ASTWAIT              = 00000001
SCH$GL_CURPCB               ********   X    03
SCH$GL_NULLPCB              ********   X    03
SCH$RAVAIL                  ********   X    03
SCH$RWAIT                   ********   X    03
SEND_STOP_SLOT              00000FF3 R    03
SHUT_TIMER                  00000906 R    03
SIZ...                    = 00000001
SLOT_LOOP_IN                00000EA9 R    03
SLOT_LOOP_OUT               00001017 R    03
SLOT_SIZE                 = 00000020
SLT_ATT_M_ABORT           = 00000020  G
SLT_ATT_S_ABORT           = 00000001  G
SLT_ATT_V_ABORT           = 00000005  G
SLT_B_ATT_CONTROL           00000004  G
SLT_B_AT_SLTSIZ             00000005  G
SLT_B_COUNT                 00000002  G
SLT_B_CRED                  00000003  G
SLT_B_DB_CONTROL            00000004  G
SLT_B_DB_IFOF               00000007  G
SLT_B_DB_IFON               00000008  G
SLT_B_DB_OFOF               00000005  G
SLT_B_DB_OFON               00000006  G
SLT_B_DSTID                 00000000  G
SLT_B_DST_NAMLEN            00000007  G
SLT_B_DT_SLTSIZ             00000006  G
SLT_B_PARAMS                00000029  G
SLT_B_REAS                  00000003  G
SLT_B_REJ_RLEN              00000004  G
SLT_B_SRCID                 00000001  G
SLT_B_SRC_NAME              00000019  G
SLT_B_SRC_NAMLEN            00000018  G
SLT_B_STP_RLEN              00000004  G
SLT_B_SVCC                  00000004  G
SLT_B_TYPE                  00000003  G
SLT_C_ATT_SLOT            = 0000000B  G
SLT_C_DA_SLOT             = 00000000  G
SLT_C_DB_SLOT             = 0000000A  G
SLT_C_INVSLOT             = 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           = 00000001  G
SLT_S_REJ_SLOT          = 00000005  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            00000005  G
SLT_T_START               00000000  G
SLT_T_STP_REAS            00000005  G
SS$_INSFMEM               ********   X   03
SS$_NORMAL                ********   X   03
STOP_B_RCODE              00000000  G
STOP_B_RLEN               00000001  G
STOP_TQE                  00001843 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_NODE_LEN           0000000C  G
STRT_B_PARAM              000000B0  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_NODE               0000000D  G
STRT_T_SYS                0000001E  G
STRT_W_FAC_NUM            00000008  G
STRT_W_MSGSIZ             00000000  G
STRT_W_PROD_TYP           0000000A  G
SYS$GW_IJOBCNT            ********   X   03
SYS$GW_IJOBLIM            ********   X   03
THREE                     000019B3 R     03
TQE$B_RQTYPE            = 0000000B
TQE$B_TYPE              = 0000000A
TQE$C_LENGTH            = 00000030
TQE$C_SSREPT            = 00000005
TQE$C_SSSNGL            = 00000001
TQE$L_FPC               = 0000000C
TQE$M_REPEAT            = 00000004
TQE$Q_DELTA             = 00000020
TQE$W_SIZE              = 00000008
TT$M_HOSTSYNC           = 00000010
TT$M_TTSYNC             = 00000020
TT$V_REMOTE             = 0000000D
TT$_UNKNOWN             = 00000000
TT2$V_AUTOBAUD          = 00000001
TT2$V_HANGUP            = 00000002
TTY$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   03
TTY$GL_OWNUIC             ********   X   02
TTY$GW_DEFBUF             ********   X   02
TTY$GW_PROT               ********   X   02
TTY$V_PC_NOTIME         = 00000000
TTY_C_BECL              = 00000007
TWO                       000019B5 R     03
UCB$B_DEVCLASS          = 00000040
UCB$B_DEVTYPE           = 00000041
UCB$B_DIPL              = 0000005E
UCB$B_ERTCNT            = 00000080
UCB$B_FIPL              = 0000000B
UCB$B_LT_CBUF             0000014C  G
UCB$B_LT_CURC             0000014B  G
UCB$B_LT_DATAW            00000142  G
UCB$B_LT_EBUF             0000019C  G
UCB$B_LT_LATSTS           00000143  G
```

```
UCB$B_LT_LOCID          00000141  G
UCB$B_LT_MAXC           0000014A  G
UCB$B_LT_RCRED          00000148  G
UCB$B_LT_REASON         00000145  G
UCB$B_LT_REMID          00000140  G
UCB$B_LT_SLOTSZ         00000149  G
UCB$B_LT_STATE          00000144  G
UCB$B_LT_TCRED          00000146  G
UCB$B_LT_XCRED          00000147  G
UCB$B_TT_DEPARI       = 000000EC
UCB$B_TT_DETYPE       = 000000F0
UCB$B_TT_OUTYPE       = 0000010B
UCB$C_LT_LENGTH         0000019C  G
UCB$C_LT_STATE_KILL   = 00000004  G
UCB$C_LT_STATE_RUN    = 00000002  G
UCB$C_LT_STATE_START  = 00000001  G
UCB$C_LT_STATE_STOP   = 00000003  G
UCB$C_TT_LENGTH       = 00000134
UCB$L_CRB             = 00000024
UCB$L_DDB             = 00000028
UCB$L_DDT             = 00000088
UCB$L_DEVCHAR         = 00000038
UCB$L_DEVCHAR2        = 0000003C
UCB$L_DEVDEPEND       = 00000044
UCB$L_LT_CSB            00000134  G
UCB$L_LT_DEADLINK       00000138  G
UCB$L_LT_INPBUF         0000013C  G
UCB$L_STS             = 00000064
UCB$L_TT_CLASS        = 00000114
UCB$L_TT_DECHA1       = 000000C8
UCB$L_TT_DECHAR       = 000000C4
UCB$L_TT_DEVDP1       = 00000048
UCB$L_TT_GETNXT       = 0000010C
UCB$L_TT_LOGUCB       = 000000C0
UCB$L_TT_OUTADR       = 0000011C
UCB$L_TT_PORT         = 00000118
UCB$L_TT_PUTNXT       = 00000110
UCB$L_TT_RTIMOU       = 000000B4
UCB$L_TT_WBLINK       = 000000D0
UCB$L_TT_WFLINK       = 000000CC
UCB$M_INT             = 00000002
UCB$M_LT_ABORT        = 00000004  G
UCB$M_LT_BOUND        = 00000008  G
UCB$M_LT_DATA         = 00000001  G
UCB$M_LT_DEAD         = 00000002  G
UCB$M_LT_FLOW         = 00000002  G
UCB$M_LT_HANGUP       = 00000001  G
UCB$M_LT_INPUT        = 00000010  G
UCB$M_LT_OVFLOW       = 00000004  G
UCB$M_ONLINE          = 00000010
UCB$S_LT_ABORT        = 00000001  G
UCB$S_LT_BOUND        = 00000001  G
UCB$S_LT_DATA         = 00000001  G
UCB$S_LT_DEAD         = 00000001  G
UCB$S_LT_FLOW         = 00000001  G
UCB$S_LT_HANGUP       = 00000001  G
UCB$S_LT_INPUT        = 00000001  G
UCB$S_LT_OVFLOW       = 00000001  G
UCB$V_DELETEUCB       = 00000010
UCB$V_LT_ABORT        = 00000002  G
UCB$V_LT_BOUND        = 00000003  G
UCB$V_LT_DATA         = 00000000  G
UCB$V_LT_DEAD         = 00000001  G
UCB$V_LT_FLOW         = 00000001  G
UCB$V_LT_HANGUP       = 00000000  G
UCB$V_LT_INPUT        = 00000004  G
UCB$V_LT_OVFLOW       = 00000002  G
UCB$V_POWER           = 00000005
UCB$V_TEMPLATE        = 0000000D
UCB$W_DEVBUFSIZ       = 00000042
UCB$W_DEVSTS          = 00000068
UCB$W_ERRCNT          = 00000082
UCB$W_MB_SEED         = 00000000
UCB$W_REFC            = 0000005C
UCB$W_TT_DESIZE       = 000000F1
UCB$W_TT_DESPEE       = 000000E8
UCB$W_TT_OUTLEN       = 00000120
UCB$W_TT_PRTCTL       = 00000122
UCB$W_TT_SPEED        = 000000F4
UCB$W_UNIT            = 00000054
USE_CREDIT              000011A2  R     03
USE_CREDIT1             00001'A8  R     03
VEC$L_INITIAL         = 0000000C
VEC$L_START           = 0000001C
VEC$L_UNITINIT        = 00000018
XMT_B_ACK               0000004F  G
XMT_B_DSTIDN            0000004A  G
XMT_B_DSTIDS            0000004B  G
XMT_B_FLAG              00000048  G
XMT_B_MC_CHG_FLAG       0000004F  G
XMT_B_MC_CIR_TIMER      00000049  G
XMT_B_MC_CUR_ECO        0000004D  G
XMT_B_MC_CUR_VER        0000004C  G
XMT_B_MC_GRP_LEN        00000054  G
XMT_B_MC_HI_VER         0000004A  G
XMT_B_MC_ID_LEN         00000099  G
XMT_B_MC_INCARN         0000004E  G
XMT_B_MC_LO_VER         0000004B  G
XMT_B_MC_MULTIMR        00000052  G
XMT_B_MC_NAME_LEN       00000088  G
XMT_B_MC_NODE_LEN       00000075  G
XMT_B_MC_NUM_SVCS       00000086  G
XMT_B_MC_RATE           00000087  G
XMT_B_MC_SET            00000052  G
XMT_B_MC_STATUS         00000053  G
XMT_B_MC_SVC            0000027A  G
XMT_B_MC_SVC_LEN        00000279  G
XMT_B_NUM_SLOTS         00000049  G
XMT_B_SEQ               0000004E  G
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_HDRLEN               = 00000008  G
XMT_C_MAXDATA              = 000005D4  G
XMT_C_MAXLEN               = 00000624  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               00000055  G
XMT_T_MC_ID                  0000009A  G
XMT_T_MC_MORE_SVC            000000DA  G
XMT_T_MC_NAME                00000089  G
XMT_T_MC_NODE                00000076  G
XMT_T_MDATA                  00000050  G
XMT_W_DSTID                  0000004A  G
XMT_W_MC_MSG_SIZ             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
$$$105_PROLOGUE	000000DE (222.)	02 (2.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC BYTE
$$$115_DRIVER	00001BA6 (7078.)	03 (3.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC LONG

+-----------------------+
! Performance indicators !
+-----------------------+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.11	00:00:01.98
Command processing	129	00:00:00.77	00:00:07.18
Pass 1	909	00:00.46.36	00:01:34.16

```
Symbol table sort                 0     00:00:04.64    00:00:06.17
Pass 2                          426     00:00:13.09    00:00:30.02
Symbol table output               1     00:00:00.57    00:00:00.98
Psect synopsis output             0     00:00:00.03    00:00:00.03
Cross-reference output            0     00:00:00.00    00:00:00.00
Assembler run totals           1496     00:01:05.58    00:02:20.52

The working set limit was 2850 pages.
232981 bytes (456 pages) of virtual memory were used to buffer the intermediate code.
There were 160 pages of symbol table space allocated to hold 2610 non-local and 349 local symbols.
4711 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
------------------                       --------------
_$255$DUA28:[LAT.OBJ]LAT.MLB;1                     1
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                    39
_$255$DUA28:[SYSLIB]STARLET.MLB;2                 10
TOTALS (all libraries)                            50

2819 GETS were required to define 50 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:LTDRIVER/OBJ=OBJ$:LTDRIVER MSRC$:LTDRIVER/UPDATE=(ENH$:LTDRIVER)+EXECML$/LIB+LIB$:LAT/LIB
```

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

LAT

LATCPCLD
CLD

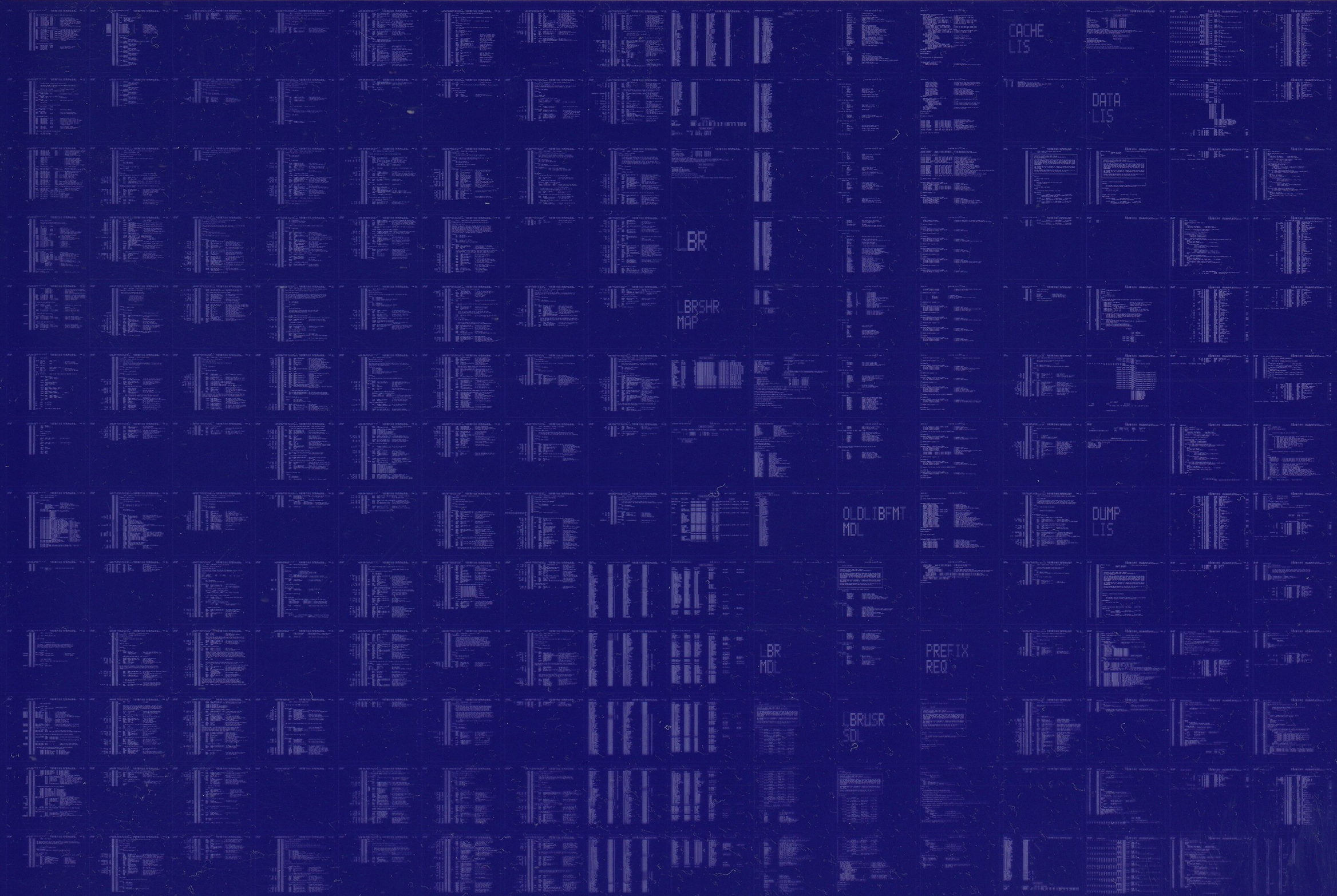
LATCP
MAP

LATDEF
SDL

LATCPMSG
LIS

LATCP
LIS

LTDRIVER
MAP

LTDRIVER
LIS

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY