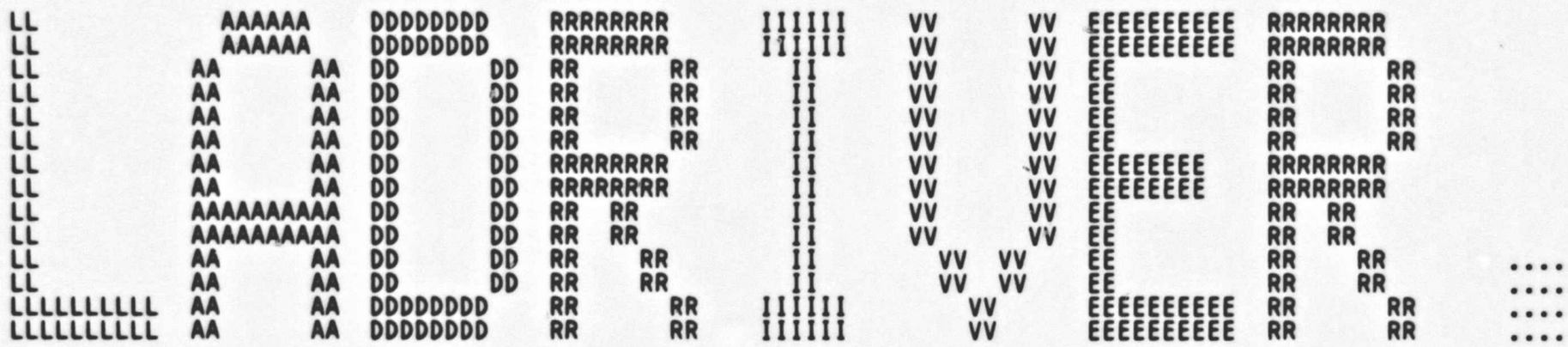
```
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRRRRRRRRRRR           III      VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD          DDD RRRRRRRRRRRR           III      VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD          DDD RRRRRRRRRRRR           III      VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD          DDD RRR   RRR              III      VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR   RRR              III      VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR   RRR              III      VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR       RRR          III        VVV     VVV    EEE              RRR       RRR
DDD          DDD RRR       RRR          III        VVV     VVV    EEE              RRR       RRR
DDD          DDD RRR       RRR          III        VVV     VVV    EEE              RRR       RRR
DDDDDDDDDDDD     RRR          RRR    IIIIIIIII       VVV         EEEEEEEEEEEEEEE  RRR          RRR
DDDDDDDDDDDD     RRR          RRR    IIIIIIIII       VVV         EEEEEEEEEEEEEEE  RRR          RRR
DDDDDDDDDDDD     RRR          RRR    IIIIIIIII       VVV         EEEEEEEEEEEEEEE  RRR          RRR
```

FILEID**LADRIVER



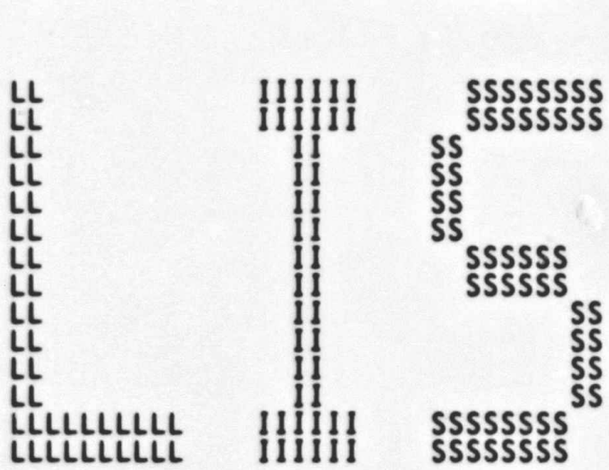


Table of contents

(2) 58 DECLARATIONS
(4) 301 LOAD_MICROCODE - FDT ROUTINE TO LOAD MICROCODE
(5) 409 RESET - RESET MICROPROCESSOR
(6) 470 STARTMP_FDT START MICROPROCESSOR FDT ROUTINE
(8) 508 INIT_FDT - INITIALIZE FDT ROUTINE
(9) 570 SETCLOCK_FDT - SET CLOCK FDT ROUTINE
(10) 605 STARTDATA_FDT - START DATA FDT ROUTINE
(11) 773 QSTOP_FDT - QUEUE STOP FDT ROUTINE
(12) 823 QUE_PRT - QUEUE I/O PACKET TO DRIVER
(13) 859 STARTIO - MAIN DRIVER ENTRY POINT
(14) 1028 SETCHAR - SET CHARACTERISTICS
(15) 1090 SDATA - START DATA PROCESSING
(17) 1187 REQUEST COMPLETE PROCESSING
(18) 1272 UNLOCK - UNLOCK PAGES AND DEALLOCATE SIP
(19) 1328 SETMAPREG - ALLOCATE AND LOAD UBA MAP REGISTERS
(20) 1407 ALLOCATE UBA MAP REGISTERS FROM LOCAL POOL
(20) 1451 ALTER LOCAL UBA MAP REGISTER BITMAP
(20) 1484 REL_MRDP - RELEASE UBA MAP REGISTERS AND DATAPATH
(21) 1567 READY IN INTERRUPT SERVICE
(22) 1622 READY OUT INTERRUPT SERVICE
(23) 1833 QUEUE_STOP_REQ - QUEUE A STOP REQUEST
(24) 1879 SIGNAL_BFR_FULL - SIGNAL BUFFER FULL (OR EMPTY) TO USER
(25) 1983 DODIAGERL - DO DIAG. AND ERROR LOGGING STUFF
(26) 2059 LA_REGDUMP - REGISTER DUMP ROUTINE
(28) 2096 CANCEL_IO - CANCEL I/O
(29) 2191 COMPLETE_ALL - COMPLETE ALL DATA TRANSFER REQUESTS
(30) 2230 UNIT_INIT - LPA-11 UNIT INITIALIZATION

```
0000     1          .TITLE  LADRIVER - LPA-11 DRIVER
0000     2          .IDENT  'V04-000'
0000     3
0000     4  ;
0000     5  ;*****************************************************************************
0000     6  ;*                                                                           *
0000     7  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9  ;*  ALL RIGHTS RESERVED.                                                     *
0000    10  ;*                                                                           *
0000    11  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    12  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    13  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    14  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    15  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    16  ;*  TRANSFERRED.                                                             *
0000    17  ;*                                                                           *
0000    18  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    19  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    20  ;*  CORPORATION.                                                             *
0000    21  ;*                                                                           *
0000    22  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    23  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24  ;*                                                                           *
0000    25  ;*                                                                           *
0000    26  ;*****************************************************************************
0000    27  ;
0000    28
0000    29  ;++
0000    30  ; FACILITY:     EXECUTIVE, I/O DRIVERS
0000    31  ;
0000    32  ; ABSTRACT:
0000    33  ;       THIS MODULE IS THE DRIVER FOR THE LPA-11 (LABORATORY PERIPHERAL
0000    34  ;       ACCELERATOR).
0000    35  ;
0000    36  ; ENVIRONMENT:  KERNEL MODE, NON-PAGED
0000    37  ;
0000    38  ; AUTHOR:  STEVE BECKHARDT,     CREATION DATE:  7-APR-78
0000    39  ;
0000    40  ; MODIFIED BY:
0000    41  ;
0000    42  ;       V03-004 RNH0001         Richard N. Holstein     28-Aug-1984
0000    43  ;               Missing number sign in V03-002 caused ACCVIO.
0000    44  ;
0000    45  ;       V03-003 KDM0059         Kathleen D. Morse       14-Jul-1983
0000    46  ;               Change time-wait loop to use new TIMEDWAIT macro.
0000    47  ;               Add $DEVDEF.
0000    48  ;
0000    49  ;       V03-002 LJA0072         Laurie J. Anderson      17-Jun-1983
0000    50  ;               Correct DODIAGERL to properly recover from insufficient space
0000    51  ;               in error log buffers error condition.
0000    52  ;
0000    53  ;       V03-001 KDM0002         Kathleen D. Morse       28-Jun-1982
0000    54  ;               Added $DCDEF and $SSDEF.
0000    55  ;
0000    56  ;--
```

```
                     58          .SBTTL  DECLARATIONS
                     59 ;
                     60 ; INCLUDE FILES:
                     61 ;
                     62          $ACBDEF                 ; AST CONTROL BLOCK OFFSETS
                     63          $ADPDEF                 ; ADP OFFSETS
                     64          $CCBDEF                 ; CCB OFFSETS
                     65          $CRBDEF                 ; CRB OFFSETS
                     66          $DCDEF                  ; DEFINE DEVICE TYPE CODES
                     67          $DDBDEF                 ; DDB OFFSETS
                     68          $DEVDEF                 ; DEFINE DEVICE CHARACTERISTICS
                     69          $DPTDEF                 ; DRIVER PROLOGUE TABLE DEFINITIONS
                     70          $DYNDEF                 ; STRUCTURE TYPE CODE DEFINITIONS
                     71          $EMBDEF                 ; EMB OFFSETS
                     72          $FKBDEF                 ; FKB OFFSETS
                     73          $IDBDEF                 ; IDB OFFSETS
                     74          $IPLDEF                 ; IPL DEFINITIONS
                     75          $IODEF                  ; I/O FUNCTION CODES
                     76          $IRPDEF                 ; IRP OFFSETS
                     77          $LADEF                  ; LPA-11 DEFINITIONS
                     78          $PCBDEF                 ; PCB OFFSETS
                     79          $PRDEF                  ; PROCESSOR REGISTER DEFINITIONS
                     80          $PRIDEF                 ; PRIORITY INCREMENT CLASS DEFINITIONS
                     81          $SSDEF                  ; SYSTEM STATUS CODES
                     82          $UCBDEF                 ; UCB OFFSETS
                     83          $VADEF                  ; VIRTUAL ADDRESS FIELD DEFINITIONS
                     84          $VECDEF                 ; INTERRUPT DISPATCH VECTOR OFFSETS
                     85
                     86 ;
                     87 ; MACROS:
                     88 ;
                     89
                     90 ;
                     91 ; EQUATED SYMBOLS:
                     92 ;
                     93
                     94 ;
                     95 ; QIO ARGUMENT LIST OFFSETS
                     96 ;
                     97
00000000             98 P1=0
00000004             99 P2=4
00000008            100 P3=8
0000000C            101 P4=12
                    102
                    103 ;
                    104 ; MISC. DEFINITIONS
                    105 ;
                    106
00000002            107 DEVADDR=2                       ; OFFSET TO DEVICE ADDRESSES IN DMDT
00000003            108 STOP_MODE=3                     ; MODE FOR STOP RDA
00000048            109 IRP$[_SIP=IRP$L_SEGVBN          ; POINTER TO SIP IN IRP
0000003C            110 IRP$L_BFR_AST=IRP$B_CARCON      ; BUFFER FULL AST ADDRESS IN IRP
00000040            111 IRP$L_OVR_AST=IRP$W_ABCNT       ; BUFFER OVERRUN AST ADDRESS IN IRP
00000040            112 IRP$L_RDAMAPREG=IRP$W_ABCNT     ; MAP REG. ALLOCATED FOR INTIALIZE
                    113
                    114 ;
```

```
                        ; LPA-11 DEVICE REGISTER OFFSETS
                        ;
                
                                $DEFINI LA
                
0000    120 $DEF        LA_CISR                 .BLKW   1       ; CONTROL IN STATUS REGISTER
0002    121             _VIELD  LA_CISR,0,<-
0002    122                     <GO,,M>,-                       ; GO BIT
0002    123                     <,1>,-                          ; RESERVED BIT
0002    124                     <MEX,2>,-                       ; MEMORY EXTENSIOON BITS
0002    125                     <,2>,-                          ; RESERVED BITS
0002    126                     <IE,,M>,-                       ; READY IN INTERRUPT ENABLE
0002    127                     <RDY,,M>,-                      ; READY IN
0002    128                     <,2>,-                          ; RESERVED BITS
0002    129                     <ROMO,,M>,-                     ; ROM OUTPUT BIT
0002    130                     <ENA,,M>,-                      ; ENABLE ARBITRATION
0002    131                     <,1>,-                          ; RESERVED BIT
0002    132                     <CRAM,,M>,-                     ; CRAM WRITE
0002    133                     <RESET,,M>,-                    ; RESET (MASTER CLEAR)
0002    134                     <RUN,,M>,-                      ; RUN
0002    135             >
0002    136
0002    137 $DEF        LA_COSR                 .BLKW   1       ; CONTROL OUT STATUS REGISTER
0004    138             _VIELD  LA_COSR,0,<-
0004    139                     <USER,3>,-                      ; USER INDEX
0004    140                     <,3>,-                          ; RESERVED BITS
0004    141                     <IE,,M>,-                       ; READY OUT INTERRUPT ENABLE
0004    142                     <RDY,,M>,-                      ; READY OUT
0004    143                     <ERRCD,5>,-                     ; ERROR CODE
0004    144                     <ERRTP,2>,-                     ; ERROR TYPE
0004    145                     <ERROR,,M>,-                    ; ERROR BIT
0004    146             >
0004    147
0004    148 $DEF        LA_RDA                  .BLKW   1       ; RDA ADDRESS REGISTER
0006    149
0006    150 $DEF        LA_MAINT                .BLKW   1       ; MAINTENANCE STATUS REGISTER
0008    151
0008    152             $DEFEND LA
0000    153
0000    154
                        ;
                        ; LPA-11 SPECIFIC UCB OFFSETS
                        ;
                
                                $DEFINI UCB
                
000000A0    161 .=UCB$L_DPC+4
00A0    162
00A0    163 $DEF        UCB$L_RDABA             .BLKL   1       ; UNIBUS ADDRESS OF RDA IN UCB
00A4    164 $DEF        UCB$L_RDAMR             .BLKL   1       ; RDA IN UCB MAP REGISTER INFO.
00A8    165 $DEF        UCB$L_PREALLOC          .BLKL   1       ; PREALLOCATED MAP REGISTER INFO.
00AC    166 $DEF        UCB$L_INQFL             .BLKL   1       ; INPUT QUEUE FORWARD LINK
00B0    167 $DEF        UCB$L_INQBL             .BLKL   1       ; INPUT QUEUE BACKWARD LINK
00B4    168 $DEF        UCB$L_FORKO             .BLKL   6       ; READY OUT INTERRUPTS FORK BLOCK
00CC    169 $DEF        UCB$L_FORKP             .BLKL   6       ; POWER RECOVERY FORK BLOCK
00E4    170 $DEF        UCB$L_REGSAVE           .BLKL   4       ; REGISTER SAVE AREA
00F4    171 $DEF        UCB$W_RISAVE            .BLKW   4       ; REG. SAVE AREA FOR READY-IN INTERRUPTS
```

```
                00FC   172 $DEF     UCB$W_ROSAVE     .BLKW   4       ; REG. SAVE AREA FOR READY-OUT INTS.
                0104   173 $DEF     UCB$L_RQLIST     .BLKL   8       ; USER REQUEST LIST
                0124   174 $DEF     UCB$W_MRBITMAP   .BLKW   31      ; MAP REGISTER BITMAP
00000164        0162   175                          .BLKW   1       ; SPARE WORD
                0164   176 $DEF     UCB$W_RDA        .BLKW   29      ; RDA
000001A0        019E   177                          .BLKW   1       ; SPARE WORD
                01A0   178
000001A0        01A0   179 UCB$K_SIZE=.
                01A0   180
                01A0   181         $DEFEND UCB
                0000   182
                0000   183 ;
                0000   184 ;       SECONDARY I/O PACKET (SIP) OFFSETS
                0000   185 ;
                0000   186         $DEFINI SIP
                0000   187
                0000   188 $DEF     SIP$W_MODE       .BLKW   1       ; LPA-11 MODE WORD
                0002   189 $DEF     SIP$W_BCNT       .BLKW   1       ; SIZE OF EACH BUFFER (IN BYTES)
00000007        0004   190                          .BLKB   3       ; SPARE BYTES
                0007   191 $DEF     SIP$B_VBFRMASK   .BLKB   1       ; VALID BUFFER MASK
                0008   192 $DEF     SIP$W_SIZE       .BLKW   1       ; SIZE OF SIP
                000A   193 $DEF     SIP$B_TYPE       .BLKB   1       ; TYPE OF DATA STRUCTURE
0000000C        000B   194                          .BLKB   1       ; SPARE
                000C   195 $DEF     SIP$L_SLVDATA    .BLKL   4       ; SLAVE DATA
                001C   196 $DEF     SIP$L_USW_SVAPT  .BLKL   1       ; USW SVAPTE
                0020   197 $DEF     SIP$W_USW_BOFF   .BLKW   1       ; USW BYTE OFFSET
                0022   198 $DEF     SIP$W_USW_BCNT   .BLKW   1       ; USW BYTE COUNT
                0024   199 $DEF     SIP$W_USW_MAPRE  .BLKW   1       ; USW STARTING MAP REGISTER
                0026   200 $DEF     SIP$B_USW_NUMRE  .BLKB   1       ; USW NUMBER OF MAP REGISTERS
                0027   201 $DEF     SIP$B_USW_DATAP  .BLKB   1       ; USW DATAPATH #
                0028   202 $DEF     SIP$L_BFR_SVAPT  .BLKL   1       ; BFR SVAPTE
                002C   203 $DEF     SIP$W_BFR_BOFF   .BLKW   1       ; BFR BYTE OFFSET
                002E   204 $DEF     SIP$W_BFR_BCNT   .BLKW   1       ; BFR BYTE COUNT
                0030   205 $DEF     SIP$W_BFR_MAPRE  .BLKW   1       ; BFR STARTING MAP REGISTER
                0032   206 $DEF     SIP$B_BFR_NUMRE  .BLKB   1       ; BFR NUMBER OF MAP REGISTERS
                0033   207 $DEF     SIP$B_BFR_DATAP  .BLKB   1       ; BFR DATAPATH #
                0034   208 $DEF     SIP$L_RCL_SVAPT  .BLKL   1       ; RCL SVAPTE
                0038   209 $DEF     SIP$W_RCL_BOFF   .BLKW   1       ; RCL BYTE OFFSET
                003A   210 $DEF     SIP$W_RCL_BCNT   .BLKW   1       ; RCL BYTE COUNT
                003C   211 $DEF     SIP$W_RCL_MAPRE  .BLKW   1       ; RCL STARTING MAP REGISTER
                003E   212 $DEF     SIP$B_RCL_NUMRE  .BLKB   1       ; RCL NUMBER OF MAP REGISTERS
                003F   213 $DEF     SIP$B_RCL_DATAP  .BLKB   1       ; RCL DATAPATH #
                0040   214
                0040   215         $DEFEND SIP
                0000   216
```

```
                        ;
                        ; OWN STORAGE:
                        ;

                        ;
                        ; DRIVER PROLOGUE TABLE
                        ;
                        DPTAB   END=LA_END,-                    ; END OF DRIVER
                                ADAPTER=UBA,-                   ; ADAPTER TYPE
                                FLAGS=DPT$M_NOUNLOAD,-          ; DRIVER IS NOT RELOADABLE
                                UCBSIZE=UCB$K_SIZE,-            ; UCB SIZE
                                NAME=LADRIVER                   ; DRIVER NAME

                        DPT_STORE INIT
                        DPT_STORE UCB,UCB$B_FIPL,B,8            ; FORK IPL
                        DPT_STORE UCB,UCB$L_DEVCHAR,L,-         ; DEVICE CHARACTERISTICS
                                <DEV$M_RTM-                     ; REAL TIME DEVICE
                                !DEV$M_AVL-                     ; AVAILABLE
                                !DEV$M_SHR-                     ; SHAREABLE
                                !DEV$M_ELG-                     ; ERROR LOGGING ENABLED
                                !DEV$M_IDV-                     ; INPUT DEVICE
                                !DEV$M_ODV>                     ; OUTPUT DEVICE
                        DPT_STORE UCB,UCB$B_DEVCLASS,B,DC$_REALTIME     ; DEVICE CLASS
                        DPT_STORE UCB,UCB$B_DEVTYPE,B,DT$_LPA11         ; DEVICE TYPE
                        DPT_STORE UCB,UCB$B_DIPL,B,22                   ; DEVICE IPL
                        DPT_STORE UCB,UCB$L_FORKO+8,L,-                 ; READY OUT FORK BLOCK
                                <<8@24>+<DYN$C_FRK@16>+FKB$K_LENGTH>    ; SIZE, TYPE, AND IPL
                        DPT_STORE UCB,UCB$L_FORKP+8,L,-                 ; POWER REC. FORK BLOCK
                                <<8@24>+<DYN$C_FRK@16>+FKB$K_LENGTH>    ; SIZE, TYPE, AND IPL

                        DPT_STORE REINIT
                        DPT_STORE DDB,DDB$L_DDT,D,LA$DDT                ; DDT ADDRESS
                        DPT_STORE CRB,CRB$L_INTD+4,D,LA$RDYOUTINTSV     ; READY OUT INT. SERVICE
                        DPT_STORE CRB,CRB$L_INTD2+4,D,LA$RDYININTSV     ; READY IN INT. SERVICE
                        DPT_STORE CRB,CRB$L_INTD+VEC$L_UNITINIT,D,UNIT_INIT ; UNIT INIT
                        DPT_STORE END

                        ;
                        ; DRIVER DISPATCH TABLE
                        ;
                        DDTAB   LA,-                            ; DEVICE NAME
                                STARTIO,-                       ; START I/O ENTRY POINT
                                0,-                             ; UNSOLICITED INTERRUPT
                                FUNCTABLE,-                     ; FUNCTION DECISION TABLE
                                CANCEL_IO,-                     ; CANCEL I/O
                                LA_REGDUMP,-                    ; REGISTER DUMP ROUTINE
                                <36+24>,-                       ; SIZE OF DIAGNOSTIC BUFFER
                                <EMB$L_DV_REGSAV+4+24>          ; SIZE OF ERROR LOGGING BUFFER

                        ;
                        ; FUNCTION DECISION TABLE
                        ;
FUNCTABLE:
                        FUNCTAB ,<LOADMCODE,STARTMPROC,-        ; LEGAL FUNCTIONS
                                INITIALIZE,SETCLOCK,SETCLOCKP,-
                                STARTDATA,STARTDATAP,-
```

```
                0038   275                 QSTOP>
                0040   276         FUNCTAB                                 ; NO BUFFERED I/O FUNCTIONS
                0048   277         FUNCTAB LOAD_MICROCODE,<LOADMCODE>      ; LOAD MICROCODE
                0054   278         FUNCTAB STARTMP_FDT,<STARTMPROC>        ; START MICROPROCESSOR
                0060   279         FUNCTAB INIT_FDT,<INITIALIZE>           ; INITIALIZE
                006C   280         FUNCTAB SETCLOCK_FDT,<SETCLOCK,-        ; SET CLOCK
                006C   281                                 SETCLOCKP>      ; SET CLOCK (PHYSICAL)
                0078   282         FUNCTAB STARTDATA_FDT,<STARTDATA,-      ; START DATA
                0078   283                                 STARTDATAP>     ; START DATA (PHYSICAL)
                0084   284         FUNCTAB QSTOP_FDT,<QSTOP>               ; QUEUE STOP
                0090   285
                0090   286
                0090   287 ;
                0090   288 ; THE FOLLOWING TABLE IS USED FOR DISPATCHING IN STARTIO.
                0090   289 ; THE ORDER OF THE ENTRIES MUST NOT BE CHANGED!
                0090   290 ;
                0090   291 IOFCTBL:  ; I/O FUNCTION CODE TABLE - USED FOR DISPATCHING IN STARTIO
      02        0090   292         .BYTE   IO$_STARTMPROC
      04        0091   293         .BYTE   IO$_INITIALIZE
      37        0092   294         .BYTE   IO$_SETCLOCK
      05        0093   295         .BYTE   IO$_SETCLOCKP
      38        0094   296         .BYTE   IO$_STARTDATA
      06        0095   297         .BYTE   IO$_STARTDATAP
      03        0096   298         .BYTE   IO$_STOP
00000007        0097   299 IOFCTBLN=.-IOFCTBL
```

```
                     0097   301          .SBTTL  LOAD_MICROCODE - FDT ROUTINE TO LOAD MICROCODE
                     0097   302
                     0097   303 ;++
                     0097   304 ; FUNCTIONAL DESCRIPTION:
                     0097   305 ;
                     0097   306 ;       THIS ROUTINE IS AN FDT ROUTINE WHICH PERFORMS THE LOAD MICROCODE
                     0097   307 ;       QIO.  IT LOCKS THE MICROCODE IMAGE IN MEMORY, CHECKS FOR NO ONGOING
                     0097   308 ;       DATA TRANSFERS, MASTER CLEAR'S THE LPA-11, CLEARS THE MICROCODE VALID
                     0097   309 ;       BIT, AND LOADS AND VERIFIES THE MICROCODE.  AFTER A SUCCESSFUL LOAD,
                     0097   310 ;       THE SHAREABLE BIT IS SET IF MULTIREQUEST MODE MICROCODE WAS LOADED
                     0097   311 ;       AND CLEARED OTHERWISE.  ALSO, THE MICROCODE TYPE IS SAVED AND THE
                     0097   312 ;       MICROCODE VALID BIT IS SET.
                     0097   313 ;
                     0097   314 ; CALLING SEQUENCE:
                     0097   315 ;
                     0097   316 ;       CALLED FROM THE FDT ROUTINE DISPATCHER IN THE QIO SYSTEM SERVICE.
                     0097   317 ;       ON COMPLETION JUMPS TO EXE$FINISHIOC.
                     0097   318 ;
                     0097   319 ; INPUT PARAMETERS:
                     0097   320 ;
                     0097   321 ;       R3      ADDRESS OF I/O PACKET
                     0097   322 ;       R4      CURRENT PROCESS PCB ADDRESS
                     0097   323 ;       R5      ADDRESS OF UCB
                     0097   324 ;       R6      ADDRESS OF CCB
                     0097   325 ;       AP      ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER
                     0097   326 ;
                     0097   327 ; OUTPUT PARAMETERS:
                     0097   328 ;
                     0097   329 ;       R0      THE LOW ORDER WORD CONTAINS A COMPLETION CODE;
                     0097   330 ;               THE HIGH ORDER WORD CONTAINS THE NUMBER OF BYTES OF
                     0097   331 ;               MICROCODE LOADED.
                     0097   332 ;
                     0097   333 ; COMPLETION CODES:
                     0097   334 ;
                     0097   335 ;       THESE ARE IN ADDITION TO THE ONES EXE$WRITELOCK CAN RETURN:
                     0097   336 ;
                     0097   337 ;       SS$_NORMAL      NORMAL
                     0097   338 ;       SS$_DATACHECK   MICROCODE LOAD ERROR
                     0097   339 ;       SS$_DEVACTIVE   DEVICE ACTIVE
                     0097   340 ;
                     0097   341 ; SIDE EFFECTS:
                     0097   342 ;
                     0097   343 ;       R1,R2,R4,R9,R10 ARE NOT SAVED
                     0097   344 ;
                     0097   345 ;--
                     0097   346
                     0097   347 LOAD_MICROCODE:
     50    6C    D0  0097   348          MOVL    P1(AP),R0               ; ADDRESS OF MICROCODE IMAGE
 51    04  AC    3C  009A   349          MOVZWL  P2(AP),R1               ; LENGTH OF IMAGE
     59    50    7D  009E   350          MOVQ    R0,R9                   ; PUT ADDRESS, SIZE INTO R9, R10
00000000'GF    16    00A1   351          JSB     G^EXE$WRITELOCK         ; LOCK IT DOWN
  64  A5    20    AA  00A7   352          BICW    #UCB$M_POWER,UCB$W_STS(R5)  ; CLEAR POWERFAIL BIT
                     00AB   353
                     00AB   354 5$:       ; COME HERE TO TRY AGAIN AFTER A POWERFAIL
     50    59    7D  00AB   355          MOVQ    R9,R0                   ; RESTORE R0, R1
                     00AE   356
                     00AE   357          ; RESET MICROPROCESSOR
```

```
                                00AE   358          DSBINT  UCB$B_FIPL(R5)          ; RAISE IPL TO FORK LEVEL
                         0078   30   00B5   359          BSBW    RESET
                           01   CA   00B8   360          BICL    #LA$M_MCVALID,-      ; CLEAR MICROCODE VALID BIT
                        44 A5           00BA   361                  UCB$L_DEVDEPEND(R5)
                                        00BC   362          ENBINT                       ; LOWER IPL
                52      08 AC   3C   00BF   363          MOVZWL  P3(AP),R2            ; GET MICRO PC TO START LOADING AT
                        00      DD   00C3   364          PUSHL   #0                   ; COUNTER OF WORDS LOADED
          51    51      FF 8F   78   00C5   365          ASHL    #-1,R1,R1            ; CONVERT BYTE TO WORD COUNT
                           32   13   00CA   366          BEQL    15$                  ; WORD COUNT = 0
                                        00CC   367
                                        00CC   368 10$:  ; LOAD NEXT MICROCODE WORD
                           64   B4   00CC   369          CLRW    LA_CISR(R4)          ; CLEAR CONTROL IN STATUS REGISTER
                     04 A4      52   B0   00CE   370          MOVW    R2,LA_RDA(R4)        ; ADDRESS TO LOAD
                     06 A4      60   B0   00D2   371          MOVW    (R0),LA_MAINT(R4)    ; MICROCODE WORD BEING LOADED
                64      0400   8F   B0   00D6   372          MOVW    #LA_CISR_M_ROMO,LA_CISR(R4)  ; SELECT ADDRESS
                64      2000   8F   A8   00DB   373          BISW    #LA_CISR_M_CRAM,LA_CISR(R4)  ; SET CRAM WRITE
                           64   B4   00E0   374          CLRW    LA_CISR(R4)          ; RESET
                                        00E2   375
                                        00E2   376          ; NOW VERIFY WORD WAS LOADED CORRECTLY
                     04 A4      52   B0   00E2   377          MOVW    R2,LA_RDA(R4)        ; MICRO ADDRESS
                64      0400   8F   B0   00E6   378          MOVW    #LA_CISR_M_ROMO,LA_CISR(R4)  ; SELECT CRAM AT ADDRESS
                     06 A4      80   B1   00EB   379          CMPW    (R0)+,LA_MAINT(R4)   ; COMPARE CONTENTS WITH ORIGINAL WORD
                           12   12   00EF   380          BNEQ    20$                  ; ERROR - NOT EQUAL
                           52   B6   00F1   381          INCW    R2                   ; ADD 1 TO MICRO PC
                     D5 6E      51   F2   00F3   382          AOBLSS  R1,(SP),10$          ; GO BACK AND LOAD NEXT WORD
                                        00F7   383
                                        00F7   384          ; SUCCESSFUL LOAD
          02    01      FE A0   F0   00F7   385          INSV    -2(R0),#LA$V_MCTYPE,#LA$S_MCTYPE,-  ; STORE MICROCODE TYPE
                        44 A5           00FC   386                  UCB$L_DEVDEPEND(R5)  ; IN DEVICE DEPENDENT CHARACTERISTICS
                50      01   3C   00FE   387 15$:  MOVZWL  S^#SS$_NORMAL,R0
                           05   11   0101   388          BRB     30$
                                        0103   389
                                        0103   390 20$:  ; ERROR DURING LOAD
                     50    005C 8F   3C   0103   391          MOVZWL  #SS$_DATACHECK,R0
                                        0108   392
                                        0108   393 30$:  ; CONVERT # OF WORDS LOADED TO BYTES AND STORE IN HIGH WORD OF R0
          50    0F      11   8E   F0   0108   394          INSV    (SP)+,#17,#15,R0
                                        010D   395          ; IF POWERFAIL OCCURRED THEN RETRY
                                        010D   396          DSBINT  #31
                06 64 A5        05   E5   0113   397          BBCC    #UCB$V_POWER,UCB$W_STS(R5),40$  ;  BRANCH IF POWER DIDN'T FAIL
                                        0118   398          ENBINT                       ; POWERFAIL OCCURRED, RETRY
                        FF8D    31   011B   399          BRW     5$
                                        011E   400
                                        011E   401 40$:  ; NO POWERFAIL - IF SUCCESSFUL LOAD, THEN SET MICROCODE VALID
                     50      01   B1   011E   402          CMPW    S^#SS$_NORMAL,R0     ; SUCCESSFUL?
                           04   12   0121   403          BNEQ    50$                  ; NO
                           01   88   0123   404          BISB    #LA$M_MCVALID,-      ; YES, SET MICROCODE VALID BIT
                        44 A5           0125   405                  UCB$L_DEVDEPEND(R5)
                                        0127   406 50$:  ENBINT
                00000000'GF     17   012A   407          JMP     G^EXE$FINISHIOC      ; RETURN TO USER
```

```
                     0130   409          .SBTTL  RESET - RESET MICROPROCESSOR
                     0130   410
                     0130   411 ;++
                     0130   412 ; FUNCTIONAL DESCRIPTION:
                     0130   413 ;
                     0130   414 ;       THIS ROUTINE VERIFIES THAT THERE ARE NO ONGOING DATA TRANSFERS,
                     0130   415 ;       AND THAT THE UCB IS NOT BUSY. IF THESE CONDITIONS ARE MET, THEN
                     0130   416 ;       A MASTER CLEAR IS ISSUED TO THE LPA-11.  OTHERWISE, THE I/O
                     0130   417 ;       IS FINISHED WITH AN ERROR STATUS.  THIS ROUTINE MUST BE CALLED
                     0130   418 ;       AT FORK IPL TO AVOID RACE CONDITIONS.
                     0130   419 ;
                     0130   420 ; CALLING SEQUENCE:
                     0130   421 ;
                     0130   422 ;       BSBW    RESET
                     0130   423 ;
                     0130   424 ; INPUT PARAMETERS:
                     0130   425 ;
                     0130   426 ;       R5      ADDRESS OF UCB
                     0130   427 ;
                     0130   428 ; IMPLICIT INPUTS:
                     0130   429 ;
                     0130   430 ;       IPL IS AT FORK LEVEL ON ENTRY
                     0130   431 ;
                     0130   432 ; OUTPUT PARAMETERS:
                     0130   433 ;
                     0130   434 ;       R4      UNIBUS ADDRESS OF FIRST LPA-11 REGISTER
                     0130   435 ;
                     0130   436 ; COMPLETION CODES:
                     0130   437 ;
                     0130   438 ;       SS$_DEVACTIVE   DEVICE ACTIVE (NOT RETURNED TO CALLER - GOES
                     0130   439 ;                       DIRECTLY TO EXE$FINISHIOC)
                     0130   440 ;
                     0130   441 ; SIDE EFFECTS:
                     0130   442 ;
                     0130   443 ;       R2 IS NOT PRESERVED
                     0130   444 ;--
                     0130   445
                     0130   446 RESET:
25 64 A5    08   E0 0130   447          BBS     #UCB$V_BSY,UCB$W_STS(R5),20$    ; MAKE SURE UCB IS NOT BUSY
                     0135   448
                     0135   449          ; MAKE SURE THERE ARE NO ONGOING DATA TRANSFERS
          52     D4 0135   450          CLRL    R2
    0104 C542    D5 0137   451 10$:     TSTL    UCB$L_RQLIST(R5)[R2]    ; A REQUEST HERE?
         1C     12 013C   452          BNEQ    20$                     ; YES, ERROR!
    F5 52  08    F2 013E   453          AOBLSS  #8,R2,10$               ; TRY NEXT SLOT
                     0142   454
                     0142   455          ; GET POINTER TO DEVICE REGISTERS
    54  24 A5   D0 0142   456          MOVL    UCB$L_CRB(R5),R4        ; GET POINTER TO CRB
                     0146   457          ASSUME  IDB$L_CSR EQ 0
    54  2C B4   D0 0146   458          MOVL    @CRB$L_INTD+VEC$L_IDB(R4),R4  ; GET PTR TO 1ST DEVICE REGISTER
                     014A   459
                     014A   460          ; RAISE IPL TO HARDWARE DEVICE LEVEL AND DO A MASTER CLEAR
                     014A   461          DSBINT  UCB$B_DIPL(R5)
64   4000 8F    B0 0151   462          MOVW    #LA_CISR_M_RESET,LA_CISR(R4)  ; DO MASTER CLEAR
                     0156   463          ENBINT
               05    0159   464          RSB
                     015A   465
```

```
                     015A   466 20$:    ; ERROR - LPA-11 IS BUSY
50   02C4 8F    3C   015A   467          MOVZWL  #SS$_DEVACTIVE,R0       ; STATUS
  00000000'GF    17   015F   468          JMP     G^EXE$FINISHIOC         ; FINISH I/O
```

```
                     0165   470          .SBTTL  STARTMP_FDT     START MICROPROCESSOR FDT ROUTINE
                     0165   471
                     0165   472 ;++
                     0165   473 ; FUNCTIONAL DESCRIPTION:
                     0165   474 ;
                     0165   475 ;       THIS ROUTINE IS THE FDT ROUTINE FOR THE START MICROPROCESSOR
                     0165   476 ;       QIO.  IT CHECKS FOR NO ACTIVE USERS, MASTER CLEARS THE LPA-11,
                     0165   477 ;       AND THEN QUEUES THE PACKET ONTO THE UCB'S INPUT QUEUE.
                     0165   478 ;
                     0165   479 ; CALLING SEQUENCE:
                     0165   480 ;
                     0165   481 ;       CALLED BY THE FDT ROUTINE DISPATCHER IN THE QIO SYSTEM SERVICE.
                     0165   482 ;       ON COMPLETION BRANCHES TO QUE_PKT
                     0165   483 ;
                     0165   484 ; INPUT PARAMETERS:
                     0165   485 ;
                     0165   486 ;       R3      ADDRESS OF I/O PACKET
                     0165   487 ;       R5      ADDRESS OF UCB
                     0165   488 ;
                     0165   489 ; OUTPUT PARAMETERS:
                     0165   490 ;
                     0165   491 ;       NONE
                     0165   492 ;
                     0165   493 ; COMPLETION CODES:
                     0165   494 ;
                     0165   495 ;       SS$_DEVACTIVE   DEVICE ACTIVE (GETS RETURNED DIRECTLY TO EXE$FINISHIOC)
                     0165   496 ;
                     0165   497 ; SIDE EFFECTS:
                     0165   498 ;
                     0165   499 ;       R2,R4 ARE NOT PRESERVED
                     0165   500 ;--
                     0165   501
                     0165   502 STARTMP_FDT:
                     0165   503          SETIPL  UCB$B_FIPL(R5)          ; RAISE IPL TO FORK LEVEL
          C5   10    0169   504          BSBB    RESET                   ; RESET MICROPROCESSOR
        01AC   31    016B   505          BRW     QUE_PKT                 ; INITIATE FUNCTION
```

```
                                016E    508          .SBTTL  INIT_FDT - INITIALIZE FDT ROUTINE
                                016E    509
                                016E    510 ;++
                                016E    511 ; FUNCTIONAL DESCRIPTION:
                                016E    512 ;
                                016E    513 ;       THIS ROUTINE IS THE FDT ROUTINE FOR THE INITIALIZE QIO.
                                016E    514 ;       IT CHECKS FOR SEVERAL ERRORS, LOCKS THE INITIALIZE TABLE INTO
                                016E    515 ;       MEMORY, AND FORMATS THE CONFIGURATION BITS WHICH GET STORED
                                016E    516 ;       IN THE DEVICE CHARACTERISTICS IF THE INITIALIZE IS SUCCESSFUL.
                                016E    517 ;
                                016E    518 ; CALLING SEQUENCE:
                                016E    519 ;
                                016E    520 ;       CALLED FROM THE FDT ROUTINE DISPATCHER IN THE QIO SYSTEM SERVICE.
                                016E    521 ;
                                016E    522 ; INPUT PARAMETERS:
                                016E    523 ;
                                016E    524 ;       R3      ADDRESS OF I/O PACKET
                                016E    525 ;       R4      CURRENT PROCESS PCB ADDRESS
                                016E    526 ;       R5      ADDRESS OF UCB
                                016E    527 ;       R6      ADDRESS OF CCB
                                016E    528 ;       AP      ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER
                                016E    529 ;
                                016E    530 ; OUTPUT PARAMETERS:
                                016E    531 ;
                                016E    532 ;       NONE
                                016E    533 ;
                                016E    534 ; COMPLETION CODES:
                                016E    535 ;
                                016E    536 ;       SS$_IVMODE      INVALID MODE
                                016E    537 ;       SS$_IVBUFLEN    INVALID BUFFER LENGTH
                                016E    538 ;       SS$_BUFNOTALIGN BUFFER NOT ALIGNED CORRECTLY
                                016E    539 ;       (THESE ERRORS GET RETURNED DIRECTLY TO EXE$FINISHIOC)
                                016E    540 ;--
                                016E    541
                                016E    542
                                016E    543 INIT_FDT:
      52    0324 8F    3C      016E    544          MOVZWL  #SS$_BUFNOTALIGN,R2     ; ASSUME ALIGNMENT ERROR
            50    6C   D0      0173    545          MOVL    P1(AP),R0               ; GET ADDRESS OF INITIALIZE TABLE
              3C  50   E8      0176    546          BLBS    R0,10$                  ; VERIFY IT'S WORD ALIGNED
            59    50   D0      0179    547          MOVL    R0,R9                   ; SAVE FOR LATER USE
      52    034C 8F    3C      017C    548          MOVZWL  #SS$_IVBUFLEN,R2        ; ASSUME INVALID LENGTH ERROR
         51     04   AC  3C      0181    549          MOVZWL  P2(AP),R1               ; GET LENGTH
00000116 8F     51   D1      0185    550          CMPL    R1,#278                 ; IS IT THE RIGHT LENGTH?
                27   12      018C    551          BNEQ    10$                     ; NO - ERROR
      00000000'GF    16      018E    552          JSB     G^EXE$WRITELOCK         ; YES, LOCK IT DOWN
      52    0354 8F    3C      0194    553          MOVZWL  #SS$_IVMODE,R2          ; ASSUME INVALID MODE ERROR
            69    07   93      0199    554          BITB    #7,(R9)                 ; MAKE SURE MODE = INITIALIZE
                  17   12      019C    555          BNEQ    10$                     ; IT DOESN'T - ERROR
                               019E    556
                               019E    557          ; BUILD CONFIGURATION BITS FOR DEVICE CHARACTERISTICS
                    51   D4    019E    558          CLRL    R1                      ; LOOP COUNTER AND BIT POSITION
      52    02 A941   B0      01A0    559 5$:      MOVW    DEVADDR(R9)[R1],R2      ; GET DEVICE ADDRESS OF NEXT DEVICE
50    01    51    52   F0      01A5    560          INSV    R2,R1,#1,R0             ; STORE LOW BIT OF ADDRESS IN R0
         F2 51    0A   F2      01AA    561          AOBLSS  #10,R1,5$               ; DO NEXT DEVICE
            38 A3    50   D2      01AE    562          MCOML   R0,IRP$L_MEDIA(R3)      ; COMPLEMENT BITS AND SAVE
                  0165   31      01B2    563          BRW     QUE_PKT                 ; QUEUE PACKET TO DRIVER
                               01B5    564
```

```
                        01B5   565 10$:    ; ERROR - EITHER INCORRECT LENGTH, MODE NOT EQUAL TO INIT,
                        01B5   566          ; OR NOT WORD ALIGNED.
          50    52      D0    01B5   567          MOVL    R2,R0                   ; COMPLETION CODE
00000000'GF     17      01B8   568          JMP     G^EXE$FINISHIOC
```

```
                                01BE   570          .SBTTL  SETCLOCK_FDT - SET CLOCK FDT ROUTINE
                                01BE   571
                                01BE   572 ;++
                                01BE   573 ; FUNCTIONAL DESCRIPTION:
                                01BE   574 ;
                                01BE   575 ;       THIS ROUTINE IS THE FDT ROUTINE FOR THE SET CLOCK QIO.
                                01BE   576 ;       IT COPIES THE FUNCTION DEPENDENT PARAMETERS INTO THE I/O
                                01BE   577 ;       PACKET AND THEN STORES THE CLOCK A RATE AND PRESET IN THE
                                01BE   578 ;       SPARE CHARACTERISTICS.  THIS WILL GET STORED IN THE DEVICE
                                01BE   579 ;       CHARACTERISTICS IF THE QIO IS SUCCESSFUL.
                                01BE   580 ;
                                01BE   581 ; CALLING SEQUENCE:
                                01BE   582 ;
                                01BE   583 ;       CALLED BY THE FDT ROUTINE DISPATCHER IN THE QIO SYSTEM SERVICE.
                                01BE   584 ;
                                01BE   585 ; INPUT PARAMETERS:
                                01BE   586 ;
                                01BE   587 ;       R3      ADDRESS OF I/O PACKET
                                01BE   588 ;       R5      UCB ADDRESS
                                01BE   589 ;       AP      ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER
                                01BE   590 ;
                                01BE   591 ; OUTPUT PARAMETERS:
                                01BE   592 ;
                                01BE   593 ;       NONE
                                01BE   594 ;--
                                01BE   595
                                01BE   596 SETCLOCK_FDT:
                                01BE   597          ; COPY P2 - P4 INTO I/O PACKET
        38 A3    04 AC    B0    01BE   598          MOVW    P2(AP),IRP$L_MEDIA(R3)   ; MODE WORD
        3A A3    08 AC    B0    01C3   599          MOVW    P3(AP),IRP$L_MEDIA+2(R3) ; CLOCK STATUS
        3C A3    0C AC    B0    01C8   600          MOVW    P4(AP),IRP$L_MEDIA+4(R3) ; CLOCK PRESET
38 A3    03    00    01    F0    01CD   601          INSV    #1,#0,#3,IRP$L_MEDIA(R3) ; SET MODE TO START CLOCK
                                01D3   602
                 0144    31    01D3   603          BRW     QUE_PKT                  ; QUEUE PACKET TO DRIVER
```

```
                                01D6   605          .SBTTL  STARTDATA_FDT - START DATA FDT ROUTINE
                                01D6   606
                                01D6   607 ;++
                                01D6   608 ; FUNCTIONAL DESCRIPTION:
                                01D6   609 ;
                                01D6   610 ;       THIS ROUTINE IS THE FDT ROUTINE FOR THE START DATA QIO.  IT
                                01D6   611 ;       ALLOCATES A SECONDARY I/O PACKET (SIP), LOCKS THE USW, BUFFERS,
                                01D6   612 ;       AND RCL INTO MEMORY AND LINKS THE SIP TO THE IRP.
                                01D6   613 ;
                                01D6   614 ; CALLING SEQUENCE:
                                01D6   615 ;
                                01D6   616 ;       CALLED FROM THE FDT ROUTINE DISPATCHER IN THE QIO SYSTEM SERVICE
                                01D6   617 ;
                                01D6   618 ; INPUT PARAMETERS:
                                01D6   619 ;
                                01D6   620 ;       R3      ADDRESS OF I/O PACKET
                                01D6   621 ;       R4      CURRENT PROCESS PCB ADDRESS
                                01D6   622 ;       R5      ADDRESS OF UCB
                                01D6   623 ;       R6      ADDRESS OF CCB
                                01D6   624 ;
                                01D6   625 ; OUTPUT PARAMETERS:
                                01D6   626 ;
                                01D6   627 ;       NONE
                                01D6   628 ;
                                01D6   629 ; COMPLETION CODES:
                                01D6   630 ;
                                01D6   631 ;       SS$_INSFMEM     INSUFFICIENT MEMEORY
                                01D6   632 ;       SS$_BUFNOTALIGN ALIGNMENT ERROR
                                01D6   633 ;       SS$_IVBUFLEN    INVALID BUFFER LENGTH
                                01D6   634 ;       (THESE ERRORS GET RETURNED DIRECTLY TO EXE$FINISHIOC)
                                01D6   635 ;
                                01D6   636 ; SIDE EFFECTS:
                                01D6   637 ;
                                01D6   638 ;       R1,R2,R7,R8 ARE NOT PRESERVED
                                01D6   639 ;--
                                01D6   640
                                01D6   641          .ENABL  LSB
                                01D6   642 STARTDATA_FDT:
                                01D6   643          ; FIRST CHECK THAT ARGUMENT BLOCK POINTED TO BY P1 IS THE CORRECT
                                01D6   644          ; LENGTH AND ACCESSIBLE
            5A    D4            01D6   645          CLRL    R10                     ; MEANS NO SIP IN CASE OF ERROR
      51    04 AC 3C            01D8   646          MOVZWL  P2(AP),R1               ; GET LENGTH
         28   51  D1            01DC   647          CMPL    R1,#40                  ; IS IT CORRECT LENGTH?
              03  13            01DF   648          BEQL    5$                      ; YES
                00E3  31        01E1   649          BRW     LENGTHERR               ; NO - ERROR
         50   6C  D0            01E4   650 5$:      MOVL    P1(AP),R0               ; YES, GET POINTER
 00000000'GF  16  E7            01E7   651          JSB     G^EXE$WRITECHK          ; CHECK FOR READ ACCESS
         59   50  D0            01ED   652          MOVL    R0,R9                   ; R9 WILL STEP THRU ARGUMENT BLOCK
                                01F0   653
                                01F0   654          ; NOW ALLOCATE SECONDARY I/O PACKET (SIP)
      51  00C4 8F  3C           01F0   655          MOVZWL  #IRP$C_LENGTH,R1        ; LENGTH
              53  DD            01F5   656          PUSHL   R3                      ; SAVE R3
 00000000'GF  16  F7            01F7   657          JSB     G^EXE$ALONONPAGED       ; ALLOCATE IT
         53   8E  D0            01FD   658          MOVL    (SP)+,R3                ; RESTORE R3
            08 50  E8           0200   659          BLBS    R0,10$                  ; SUCCESSFUL
      50  0124 8F  3C           0203   660          MOVZWL  #SS$_INSFMEM,R0         ; ERROR
                00C3  31        0208   661          BRW     ABORT
```

```
                                 020B   662
                                 020B   663 10$:    ; CLEAR PACKET AND PUT IN SIZE
                          3F     BB     020B   664          PUSHR   #^M<R0,R1,R2,R3,R4,R5>
      62   51   00   62   00     2C     020D   665          MOVC5   #0,(R2),#0,R1,(R2)      ; CLEAR PACKET
                          3F     BA     0213   666          POPR    #^M<R0,R1,R2,R3,R4,R5>
             08 A2  00C4   8F     B0     0215   667          MOVW    #IRP$C_LENGTH,IRP$W_SIZE(R2)
                    5A     52     D0     021B   668          MOVL    R2,R10                  ; R10 WILL POINT TO SIP
                                 021E   669
                                 021E   670          ; START BUILDING SIP FROM ARGUMENT BLOCK
                    6A     89     B0     021E   671          MOVW    (R9)+,SIP$W_MODE(R10)   ; COPY MODE WORD
          6A   03   00     02     F0     0221   672          INSV    #2,#0,#3,SIP$W_MODE(R10); MAKE SURE FUNCTION = START DATA
                    5B     89     3C     0226   673          MOVZWL  (R9)+,R11               ; GET VALID BUFFER MASK
                 07 AA     5B     90     0229   674          MOVB    R11,SIP$B_VBFRMASK(R10) ; STORE IN SIP
                 5B FFF8   8F     AA     022D   675          BICW    #^XFFF8,R11             ; MASK EVERYTHING BUT # OF BUFFERS
                           5B     D6     0232   676          INCL    R11                     ; ADD 1 TO GET TRUE # OF BUFFERS
                                 0234   677
                                 0234   678 20$:    ; CHECK AND LOCK USW
                    50     89     D0     0234   679          MOVL    (R9)+,R0                ; POINTER TO USW
                       59   50     E8     0237   680          BLBS    R0,45$                  ; BRANCH IF NOT WORD ALIGNED (ERROR)
                    51     02     D0     023A   681          MOVL    #2,R1                   ; LENGTH OF USW
                         0094     30     023D   682          BSBW    READLOCK                ; CHECK AND LOCK FOR WRITE ACCESS
                1C AA    2C     A3     7D     0240   683          MOVQ    IRP$L_SVAPTE(R3),SIP$L_USW_SVAPT(R10) ; SAVE SVAPTE, BOFF, BCNT
                                 0245   684
                                 0245   685          ; CHECK DATA BUFFER AREA FOR PROPER ALIGNMENT AND SIZE RESTRICTIONS
                    51     69     3C     0245   686          MOVZWL  (R9),R1                 ; LENGTH OF BUFFER AREA
                    59     04     C0     0248   687          ADDL    #4,R9
                    50     89     D0     024B   688          MOVL    (R9)+,R0                ; POINTER TO BUFFER AREA
                    50     03     D3     024E   689          BITL    #3,R0                   ; MAKE SURE ITS LONGWORD ALIGNED
                           6D     12     0251   690          BNEQ    ALIGNERR                ; IT'S NOT - ERROR!
                           52     D4     0253   691          CLRL    R2
   58   52   51             5B     7B     0255   692          EDIV    R11,R1,R2,R8            ; GET SIZE OF EACH DATA BUFFER
                           6B     13     025A   693          BEQL    LENGTHERR               ; BUFFER LENGTH CAN'T BE ZERO!
                           58     D5     025C   694          TSTL    R8                      ; MAKE SURE REMAINDER IS ZERO
                           67     12     025E   695          BNEQ    LENGTHERR               ; IT'S NOT - ERROR!
                        64 52     E8     0260   696          BLBS    R2,LENGTHERR            ; BUFFER SIZE MUST BE A MULTIPLE
                                 0263   697                                                 ; OF 2 IN MULTIREQUEST MODE.
                 05 6A     03     E0     0263   698          BBS     #3,SIP$W_MODE(R10),27$  ; BR. IF THIS IS A M.R. MODE REQUEST
                    52     03     D3     0267   699          BITL    #3,R2                   ; BUFFER SIZE MUST BE A MULTIPLE
                                 026A   700                                                 ; OF 4 IN DEDICATED MODE.
                           5B     12     026A   701          BNEQ    LENGTHERR               ; IT'S NOT - ERROR!
                 02 AA     52     B0     026C   702 27$:     MOVW    R2,SIP$W_BCNT(R10)      ; STORE BUFFER SIZE IN SIP
                                 0270   703
                                 0270   704          ; NOW CHECK AND LOCK BUFFERS FOR READ OR WRITE ACCESS DEPENDING
                                 0270   705          ; ON TRANSFER DIRECTION
                           6A     95     0270   706          TSTB    SIP$W_MODE(R10)         ; TEST FOR TRANSFER DIRECTION
                           04     19     0272   707          BLSS    30$
                           5E     10     0274   708          BSBB    READLOCK                ; FROM LPA TO MEMORY
                           02     11     0276   709          BRB     40$
                           62     10     0278   710 30$:     BSBB    WRITELOCK               ; FROM MEMORY TO LPA
             28 AA    2C     A3     7D     027A   711 40$:     MOVQ    IRP$L_SVAPTE(R3),SIP$L_BFR_SVAPT(R10) ; SAVE SVAPTE, BOFF, BCNT
                                 027F   712
                                 027F   713          ; REPEAT FOR RCL
                    51     69     3C     027F   714          MOVZWL  (R9),R1                 ; LENGTH OF RCL
                    59     04     C0     0282   715          ADDL    #4,R9
                    50     89     D0     0285   716          MOVL    (R9)+,R0                ; ADDRESS OF RCL
              6A  0300     8F     B3     0288   717          BITW    #^X300,SIP$W_MODE(R10)  ; IS RCL SPECIFIED?
                           1A     12     028D   718          BNEQ    50$                     ; NO
```

```
                     51   B5  028F   719          TSTW    R1                              ; YES, MAKE SURE LENGTH IS NOT ZERO
                     34   13  0291   720          BEQL    LENGTHERR                       ; IT IS ZERO - ERROR
                  2A 50   E8  0293   721 45$:     BLBS    R0,ALIGNERR                     ; RCL MUST BE WORD ALIGNED
                  2E 51   E8  0296   722          BLBS    R1,LENGTHERR                    ; AND A MULTIPLE OF 2 IN LENGTH
                57   50   7D  0299   723          MOVQ    R0,R7                           ; SAVE R0,R1 IN R7,R8
                     3E   10  029C   724          BSBB    WRITELOCK                       ; CHECK ACCESS AND LOCK DOWN
              34 AA  2C A3   7D  029E   725          MOVQ    IRP$L_SVAPTE(R3),SIP$L_RCL_SVAPT(R10) ; SAVE SVAPTE, BCNT, BOFF
          1E FF A748   07   E1  02A3   726          BBC     #7,-1(R7)[R8],LENGTHERR ; MAKE SURE END OF RCL HAS HIGH BIT SET
                               02A9   727
              0C AA    89   7D  02A9   728 50$:     MOVQ    (R9)+,SIP$L_SLVDATA(R10)        ; COPY SLAVE DATA
              14 AA    89   7D  02AD   729          MOVQ    (R9)+,SIP$L_SLVDATA+8(R10)
                               02B1   730          ASSUME  IRP$L_OVR_AST EQ IRP$L_BFR_AST+4
           3C A3    08 AC   7D  02B1   731          MOVQ    P3(AP),IRP$L_BFR_AST(R3)        ; COPY AST ADDRESSES
                  2C A3   7C  02B6   732          CLRQ    IRP$L_SVAPTE(R3)                ; CLEAR SVAPTE, BCNT, AND BOFF IN IRP
              48 A3    5A   D0  02B9   733          MOVL    R10,IRP$L_SIP(R3)               ; LINK SIP TO IRP
                     005A   31  02BD   734          BRW     QUE_PKT                         ; QUEUE PACKET TO DRIVER
                               02C0   735
                               02C0   736
                               02C0   737          ; ERRORS COME HERE
                               02C0   738
                               02C0   739 ALIGNERR:  ; ALIGNMENT ERROR
           50    0324 8F   3C  02C0   740          MOVZWL  #SS$_BUFNOTALIGN,R0
                     05   11  02C5   741          BRB     60$
                               02C7   742
                               02C7   743 LENGTHERR:  ; INVALID LENGTH ERROR
           50    034C 8F   3C  02C7   744          MOVZWL  #SS$_IVBUFLEN,R0
                     17   10  02CC   745 60$:     BSBB    CLEANUP                         ; UNLOCK PAGES, DEALLOCATE SIP
                               02CE   746
            00000000'GF   17   02CE   747 ABORT:   JMP     G^EXE$FINISHIOC
                               02D4   748
                               02D4   749
                               02D4   750          ; LOCAL SUBROUTINES
                               02D4   751
                               02D4   752 READLOCK:
      00000000'GF    16   02D4   753          JSB     G^EXE$READLOCKR                 ; LOCK PAGES FOR WRITE ACCESS
                06   11  02DA   754          BRB     70$
                               02DC   755
                               02DC   756 WRITELOCK:
      00000000'GF    16   02DC   757          JSB     G^EXE$WRITELOCKR                ; LOCK PAGES FOR READ ACCESS
                0F 50   E8  02E2   758 70$:     BLBS    R0,90$                          ; BRANCH IF EVERYTHING IS OK
                               02E5   759
                               02E5   760          ; ERROR OR HAVE TO FAULT PAGES IN.  FALL THROUGH TO ...
                               02E5   761
                               02E5   762
                               02E5   763 CLEANUP:  ; UNLOCK PAGES AND DEALLOCATE SIP
                     3F   BB  02E5   764          PUSHR   #^M<R0,R1,R2,R3,R4,R5>
                  2C A3   7C  02E7   765          CLRQ    IRP$L_SVAPTE(R3)                ; CLEAR SVAPTE, BCNT, AND BOFF IN IRP
                55   5A   D0  02EA   766          MOVL    R10,R5                          ; ADDRESS OF SIP
                     03   13  02ED   767          BEQL    80$                             ; NO SIP - NOTHING TO UNLOCK
                   028A   30  02EF   768          BSBW    UNLOCK                          ; UNLOCK PAGES, DEALLOCATE SIP
                     3F   BA  02F2   769 80$:     POPR    #^M<R0,R1,R2,R3,R4,R5>
                     05   02F4   770 90$:     RSB                                     ; RETURN TO CALLER OR COROUTINE
                               02F5   771          .DSABL  LSB
```

```
                     02F5   773          .SBTTL  QSTOP_FDT - QUEUE STOP FDT ROUTINE
                     02F5   774
                     02F5   775 ;++
                     02F5   776 ; FUNCTIONAL DESCRIPTION:
                     02F5   777 ;
                     02F5   778 ;       THIS ROUTINE IS AN FDT ROUTINE WHICH PERFORMS THE QUEUE STOP
                     02F5   779 ;       QIO.  NOTE THAT THIS QIO DOES NOT ITSELF STOP A DATA TRANSFER;
                     02F5   780 ;       RATHER IT QUEUES THE ORIGINAL START DATA I/O PACKET BACK TO THE
                     02F5   781 ;       DRIVER AS A STOP.  THEREFORE, THIS QIO COMPLETES AS SOON AS
                     02F5   782 ;       THE STOP IS QUEUED.  THE ORIGINAL START DATA COMPLETES AFTER THE
                     02F5   783 ;       DATA TRANSFER HAS ACTUALLY STOPPED.
                     02F5   784 ;
                     02F5   785 ; CALLING SEQUENCE:
                     02F5   786 ;
                     02F5   787 ;       CALLED FROM THE FDT ROUTINE DISPATCHER IN THE QIO SYSTEM SERVICE.
                     02F5   788 ;       ON COMPLETION JUMPS TO EXE$FINISHIOC.
                     02F5   789 ;
                     02F5   790 ; INPUT PARAMETERS:
                     02F5   791 ;
                     02F5   792 ;       R3      ADDRESS OF I/O PACKET
                     02F5   793 ;       R4      CURRENT PROCESS PCB ADDRESS
                     02F5   794 ;       R5      ADDRESS OF UCB
                     02F5   795 ;       AP      ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER
                     02F5   796 ;
                     02F5   797 ; OUTPUT PARAMETERS:
                     02F5   798 ;
                     02F5   799 ;       R0      COMPLETION CODE
                     02F5   800 ;
                     02F5   801 ; COMPLETION CODES:
                     02F5   802 ;
                     02F5   803 ;       SS$_NORMAL      NORMAL
                     02F5   804 ;       SS$_BADPARAM    NO SUCH REQUEST
                     02F5   805 ;
                     02F5   806 ; SIDE EFFECTS:
                     02F5   807 ;
                     02F5   808 ;       R2 IS NOT PRESERVED
                     02F5   809 ;--
                     02F5   810
                     02F5   811 QSTOP_FDT:
 52    04 AC    9A   02F5   812          MOVZBL  P2(AP),R2               ; GET REQUEST NUMBER
 52    F8 8F    8A   02F9   813          BICB    #^XF8,R2                ; CLEAR ALL BUT LOW THREE BITS
    50    14    3C   02FD   814          MOVZWL  #SS$_BADPARAM,R0        ; ASSUME ERROR
                     0300   815          SETIPL  UCB$B_FIPL(R5)          ; RAISE TO FORK IPL
  0104 C542    D5   0304   816          TSTL    UCB$L_RQLIST(R5)[R2]    ; IS THERE A REQUEST IN THIS SLOT?
       09    13   0309   817          BEQL    10$                     ; NO - ERROR
    50    2C    3C   030B   818          MOVZWL  #SS$_ABORT,R0           ; YES - QUEUE A STOP WITH ABORT STATUS
      04FF    30   030E   819          BSBW    QUEUE_STOP_REQ
    50    01    3C   0311   820          MOVZWL  S^#SS$_NORMAL,R0        ; RETURN NORMAL STATUS
00000000'GF    17   0314   821 10$:     JMP     G^EXE$FINISHIOC         ; FINISH I/O
```

```
                        031A   823          .SBTTL  QUE_PKT - QUEUE I/O PACKET TO DRIVER
                        031A   824
                        031A   825 ;++
                        031A   826 ; FUNCTIONAL DESCRIPTION:
                        031A   827 ;
                        031A   828 ;       THIS ROUTINE IS JUMPED TO FROM AN FDT ROUTINE TO QUEUE AN
                        031A   829 ;       I/O PACKET TO THE DRIVER.  IF THE DRIVER IS NOT BUSY, THEN
                        031A   830 ;       THE DRIVER IS CALLED IMMEDIATELY.  THIS ROUTINE IS SIMILAR TO
                        031A   831 ;       THE EXEC'S, EXCEPT IT USES A DIFFERENT QUEUE.
                        031A   832 ;
                        031A   833 ; CALLING SEQUENCE:
                        031A   834 ;
                        031A   835 ;       JUMPED TO FROM AN FDT ROUTINE
                        031A   836 ;
                        031A   837 ; INPUT PARAMETERS:
                        031A   838 ;
                        031A   839 ;       R3      ADDRESS OF I/O PACKET
                        031A   840 ;       R5      ADDRESS OF UCB
                        031A   841 ;
                        031A   842 ; OUTPUT PARAMETERS:
                        031A   843 ;
                        031A   844 ;       NONE
                        031A   845 ;--
                        031A   846
                        031A   847 QUE_PKT:
                        031A   848          DSBINT  UCB$B_FIPL(R5)              ; RAISE IPL TO FORK LEVEL
08 64 A5    08  E2      0321   849          BBSS    #UCB$V_BSY,UCB$W_STS(R5),10$  ; SET BUSY AND SEE IF IT WAS SET
  00000000'GF   16      0326   850          JSB     G^IOC$INITIATE              ; NOT BUSY, INITIATE FUNCTION
          0B    11      032C   851          BRB     20$
                        032E   852
 52   00AC C5   DE      032E   853 10$:     MOVAL   UCB$L_INOFL(R5),R2          ; GET ADDRESS OF I/O QUEUE LISTHEAD
  00000000'GF   16      0333   854          JSB     G^EXE$INSERTIRP             ; INSERT IN QUEUE BY PRIORITY
                        0339   855
                        0339   856 20$:     ENBINT                              ; LOWER IPL
00000000'GF     17      033C   857          JMP     G^EXE$QIORETURN             ; RETURN FROM QIO
```

```
                     0342   859          .SBTTL  STARTIO - MAIN DRIVER ENTRY POINT
                     0342   860
                     0342   861 ;++
                     0342   862 ; FUNCTIONAL DESCRIPTION:
                     0342   863 ;
                     0342   864 ;       THIS ROUTINE IS THE MAIN DRIVER ENTRY POINT.  IT STARTS THE I/O,
                     0342   865 ;       WAITS FOR AN INTERRUPT, COMPLETES THE I/O, AND STARTS THE NEXT ONE.
                     0342   866 ;
                     0342   867 ; CALLING SEQUENCE:
                     0342   868 ;
                     0342   869 ;       CALLED THROUGH THE DRIVER DISPATCH TABLE
                     0342   870 ;
                     0342   871 ; INPUT PARAMETERS:
                     0342   872 ;
                     0342   873 ;       R3      ADDRESS OF I/O PACKET
                     0342   874 ;       R5      ADDRESS OF UCB
                     0342   875 ;
                     0342   876 ; OUTPUT PARAMETERS:
                     0342   877 ;
                     0342   878 ;       NONE
                     0342   879 ;--
                     0342   880
                     0342   881          .ENABL  LSB
                     0342   882 STARTIO:
                     0342   883
                     0342   884          ASSUME  IRP$S_FCODE EQ 6
          52  20 A3  CO 8F  8B  0342   885          BICB3   #^XC0,IRP$W_FUNC(R3),R2 ; GET FUNCTION CODE
                     0348   886
                     0348   887          ; DISPATCH TO APPROPRIATE ROUTINE
      FD42 CF  07  52  3A  0348   888          LOCC    R2,#IOFCTBLN,IOFCTBL    ; LOCATE FUNCTION CODE IN TABLE
            51  24 A5  D0  034E   889          MOVL    UCB$L_CRB(R5),R1        ; GET POINTER TO CRB IN R1
                     0352   890          CASE    TYPE=B,SRC=R0,DISPLIST=<-
                     0352   891                  STRT_NXT_REQ,-          ; INVALID FUNCTION
                     0352   892                  STOP,-                  ; STOP
                     0352   893                  START_DATA,-            ; START DATA (PHYSICAL)
                     0352   894                  START_DATA,-            ; START DATA
                     0352   895                  SET_CLOCK,-             ; SET CLOCK (PHYSICAL)
                     0352   896                  SET_CLOCK,-             ; SET CLOCK
                     0352   897                  INITIALIZE,-            ; INITIALIZE
                     0352   898                  >
                     0364   899
                     0364   900          ; FALL THROUGH TO ...
                     0364   901
                     0364   902 ;
                     0364   903 ;       S T A R T   M I C R O P R O C E S S O R
                     0364   904 ;
                     0364   905          ; NOTE:  THIS QIO COMES HERE DIRECTLY FROM THE FDT ROUTINE.
                     0364   906          ; THEREFORE R4 POINTS TO LPA-11 CSR.
                     0364   907          ; CHECK FOR VALID MICROCODE BEFORE STARTING MICROPROCESSOR
                     0364   908          ASSUME  LA$M_MCVALID EQ 1
                     0364   909          DSBINT  #31                     ; DON'T ALLOW INTERRUPTS (LIKE PWRFAIL)
        03 44 A5  E8  036A   910          BLBS    UCB$L_DEVDEPEND(R5),10$ ; BRANCH IF MICROCODE IS VALID
             0085  31  036E   911          BRW     MCNVALID                ; BRANCH IF MICROCODE IS NOT VALID
                     0371   912 10$:
                     0371   913
                     0371   914          ; ACTUALLY START MICROPROCESSOR
         8800 8F  B0  0371   915          MOVW    #LA_CISR_M_RUN!LA_CISR_M_ENA,-  ; SET RUN AND ENABLE
```

```
                64        0375   916                     LA_CISR(R4)                     ; ARBITRATION BITS
                          0376   917             ENBINT                                 ; ALLOW INTERRUPTS
                          0379   918
                          0379   919             ; WAIT FOR AT LEAST 1 MICROSECOND BEFORE ENABLING INTERRUPTS
                          0379   920             TIMEDWAIT TIME=#1                      ; 1 10MS WAIT LOOP
                          0397   921
                          0397   922             DSBINT  #31                            ; CHECK FOR VALID MICROCODE AGAIN
          55 44 A5   E9   039D   923             BLBC    UCB$L_DEVDEPEND(R5),MCNVALID  ; BRANCH IF MICROCODE NOT VALID
       64   0040 8F   A8   03A1   924             BISW    #LA_CISR_M_IE,LA_CISR(R4)  ; ENABLE READY IN INTERRUPTS
    02 A4   0040 8F   A8   03A6   925             BISW    #LA_COSR_M_IE,LA_COSR(R4)  ; ENABLE READY OUT INTERRUPTS
              59   11   03AC   926             BRB     WAIT                           ; WAIT FOR INTERRUPT
                          03AE   927
                          03AE   928 ;
                          03AE   929 ;       S E T   C L O C K
                          03AE   930 ;
                          03AE   931 SET_CLOCK:
    0164 C5    38 A3   7D   03AE   932             MOVQ    IRP$L_MEDIA(R3),UCB$W_RDA(R5)   ; BUILD RDA IN UCB
              0E   11   03B4   933             BRB     RDA_IN_UCB
                          03B6   934
                          03B6   935 ;
                          03B6   936 ;       S T A R T   D A T A
                          03B6   937 ;
                          03B6   938 START_DATA:
             00C1   30   03B6   939             BSBW    SDATA                          ; PREPARE FOR START DATA
           70 50   E9   03B9   940             BLBC    R0,DONE                        ; ERROR
              06   11   03BC   941             BRB     RDA_IN_UCB
                          03BE   942
                          03BE   943 ;
                          03BE   944 ;       S T O P
                          03BE   945 ;
                          03BE   946 STOP:
                          03BE   947             ; RDA IS IN SIP (FROM WHEN REQUEST WAS STARTED)
                          03BE   948             ASSUME  SIP$W_MODE EQ 0
    0164 C5    48 B3   B0   03BE   949             MOVW    @IRP$L_SIP(R3),UCB$W_RDA(R5)  ; COPY RDA INTO UCB
                          03C4   950
                          03C4   951 RDA_IN_UCB:
                          03C4   952             ; SET CLOCK, START DATA, AND STOP COME HERE.  THE RDA IS IN UCB$W_RDA.
                          03C4   953             ; GET 18 BIT UNIBUS ADDRESS OF RDA
       52   00A0 C5   D0   03C4   954             MOVL    UCB$L_RDABA(R5),R2
              13   11   03C9   955             BRB     COMMON
                          03CB   956
                          03CB   957 ;
                          03CB   958 ;       I N I T I A L I Z E
                          03CB   959 ;
                          03CB   960 INITIALIZE:
                          03CB   961             ; INITIALIZE IS THE ONLY FUNCTION WHERE THE RDA IS IN THE PROCESS
                          03CB   962             ; ADDRESS SPACE.  MOVE RDA DESCRIPTOR FROM IRP TO UCB.
    78 A5    2C A3   7D   03CB   963             MOVQ    IRP$L_SVAPTE(R3),UCB$L_SVAPTE(R5)
                          03D0   964
                          03D0   965             ; SET UP MAP REGISTERS
             37 A1   94   03D0   966             CLRB    CRB$L_INTD+VEC$B_DATAPATH(R1)  ; USE DIRECT DATAPATH
              01DC   30   03D3   967             BSBW    SETMAPREG                      ; REQUEST AND LOAD UBA MAP REGISTERS
             53 50   E9   03D6   968             BLBC    R0,DONE                        ; ALLOCATION FAILURE
             34 A1   D0   03D9   969             MOVL    CRB$L_INTD+VEC$W_MAPREG(R1),-  ; SAVE ALLOCATED MAP REGISTER
             40 A3                                       IRP$L_RDAMAPREG(R3)            ; INFO. IN IRP.
                          03DC   970
                          03DE   971
                          03DE   972 COMMON: ; COMMON FUNCTION PROCESSING.  INITIALIZE, SET CLOCK, START
```

```
                        03DE   973          ; DATA, AND STOP ALL COME HERE.  R2 CONTAINS 18 BIT UNIBUS ADDRESS
                        03DE   974          ; OF RDA.
                        03DE   975
                        03DE   976          ; GET POINTER TO LPA-11 DEVICE REGISTERS
                        03DE   977          ASSUME  IDB$L_CSR EQ 0
              54  2C B1  D0    03DE   978          MOVL    @CRB$L_INTD+VEC$L_IDB(R1),R4    ; GET PTR TO 1ST DEVICE REGISTER
                        03E2   979
                        03E2   980          ; BUILD WORD TO LOAD INTO LA_CISR IN R1
        51  52  F2 8F  78    03E2   981          ASHL    #-14,R2,R1                      ; PUT HIGH TWO BITS INTO POSITION IN R1
              51  03  AA    03E7   982          BICW    #3,R1                           ; CLEAR LOW TWO BITS
                  51  B6    03EA   983          INCW    R1                              ; SET GO BIT
                        03EC   984
                        03EC   985          ; CHECK FOR VALID MICROCODE, LOAD LPA-11 REGISTERS, AND THEN WAIT
                        03EC   986          ; FOR INTERRUPT (THIS ALSO CHECKS FOR POWERFAIL)
                        03EC   987          DSBINT  #31                             ; DON'T ALLOW INTERRUPTS (LIKE PWRFAIL)
                0A 44 A5  E8    03F2   988          BLBS    UCB$L_DEVDEPEND(R5),LOAD  ; BRANCH IF MICROCODE IS VALID
                        03F6   989
                        03F6   990  MCNVALID:   ; MICROCODE IS NOT VALID - COMPLETE REQUEST WITH ERROR
                        03F6   991          ENBINT                                  ; ALLOW INTERRUPTS
           50   035C 8F  3C    03F9   992          MOVZWL  #SS$_MCNOTVALID,R0      ; ERROR CODE
                  2C  11    03FE   993          BRB     DONE                            ; COMPLETE REQUEST
                        0400   994
                        0400   995  LOAD:    ; LOAD LPA-11 REGISTERS
              04 A4  52  B0    0400   996          MOVW    R2,LA_RDA(R4)                   ; LOAD UNIBUS ADDRESS OF RDA
                 64  51  A8    0404   997          BISW2   R1,LA_CISR(R4)                  ; GO!
                        0407   998
                        0407   999  WAIT:    ; WAIT FOR INTERRUPT
                        0407  1000          WFIKPCH TIMEOUT,#2                      ; WAIT FOR READY IN INTERRUPT.
                        0411  1001                                                  ; READY OUT INTERRUPTS DON'T COME HERE.
                        0411  1002                                                  ; (GO TO 'TIMEOUT' ON TIMEOUT OR
                        0411  1003                                                  ; POWERFAIL)
                        0411  1004          IOFORK                                  ; FORK TO DRIVER LEVEL
           53   58 A5  D0    0417  1005          MOVL    UCB$L_IRP(R5),R3        ; GET ADDRESS OF CURRENT I/O PACKET
                  14  13    041B  1006          BEQL    STRT_NXT_REQ            ; THERE IS NONE - ALREADY HANDLED
                 58 A5  D4    041D  1007          CLRL    UCB$L_IRP(R5)           ; CLEAR CURRENT I/O PACKET
                     23  10    0420  1008          BSBB    SETCHAR                 ; SET CHARACTERISTICS IF APPROPRIATE
                        0422  1009
                        0422  1010          ; COPY LPA REGISTERS FROM INTERRUPT SAVE AREA TO COMMON SAVE AREA
    00E4 C5   00F4 C5   7D    0422  1011          MOVQ    UCB$W_RISAVE(R5),UCB$L_REGSAVE(R5)
                        0429  1012
              50   01   3C    0429  1013          MOVZWL  S^#SS$_NORMAL,R0        ; SUCCESS STATUS
                        042C  1014
                        042C  1015  DONE:    ; REQUESTS COME HERE WHEN DONE WITH STATUS IN R0
                     51  D4    042C  1016          CLRL    R1
                   00D9  30    042E  1017          BSBW    REQ_COMPLETE
                        0431  1018
                        0431  1019  STRT_NXT_REQ:    ; START NEXT REQUEST
           53   00AC D5  0F    0431  1020          REMQUE  @UCB$L_INQFL(R5),R3     ; GET NEXT I/O PACKET IN QUEUE
                     06  1D    0436  1021          BVS     60$                     ; THERE ISN'T ONE
         00000000'GF   17    0438  1022          JMP     G^IOC$INITIATE
     64 A5   0100 8F  AA    043E  1023  60$:     BICW    #UCB$M_BSY,UCB$W_STS(R5)  ; CLEAR UNIT BUSY
                     05  0444  1024          RSB
                        0445  1025
                        0445  1026          .DSABL  LSB
```

```
                     0445  1028          .SBTTL  SETCHAR - SET CHARACTERISTICS
                     0445  1029
                     0445  1030 ;++
                     0445  1031 ; FUNCTIONAL DESCRIPTION:
                     0445  1032 ;
                     0445  1033 ;       THIS ROUTINE SETS DEVICE DEPENDENT CHARACTERISTICS AFTER THE
                     0445  1034 ;       SUCCESSFUL COMPLETION OF AN INITIALIZE OR SET CLOCK QIO.
                     0445  1035 ;       FOR INITIALIZE, THE CONFIGURATION BITS ARE SET.  FOR SET CLOCK
                     0445  1036 ;       THE CLOCK RATE AND PRESET ARE STORED IF CLOCK A WAS SET.
                     0445  1037 ;
                     0445  1038 ; CALLING SEQUENCE:
                     0445  1039 ;
                     0445  1040 ;       BSBW/B
                     0445  1041 ;
                     0445  1042 ; INPUT PARAMETERS:
                     0445  1043 ;
                     0445  1044 ;       R3      ADDRESS OF IRP
                     0445  1045 ;       R5      ADDRESS OF UCB
                     0445  1046 ;
                     0445  1047 ; IMPLICIT INPUTS:
                     0445  1048 ;
                     0445  1049 ;       THE CHARACTERISTICS ARE IN OFFSETS IRP$L_MEDIA THROUGH
                     0445  1050 ;       IRP$L_MEDIA+5 OF THE I/O PACKET
                     0445  1051 ;
                     0445  1052 ; OUTPUT PARAMETERS:
                     0445  1053 ;
                     0445  1054 ;       NONE
                     0445  1055 ;
                     0445  1056 ; SIDE EFFECTS:
                     0445  1057 ;
                     0445  1058 ;       R0,R2 ARE NOT PRESERVED
                     0445  1059 ;--
                     0445  1060
                     0445  1061 SETCHAR:
                     0445  1062          ASSUME  IRP$S_FCODE EQ 6
52   20 A3   C0 8F   8B 0445  1063          BICB3   #^XC0,IRP$W_FUNC(R3),R2  ; GET I/O FUNCTION CODE
                     044B  1064
                     044B  1065          ; IS IT INITIALIZE?
        04   52   91 044B  1066          CMPB    R2,#IO$_INITIALIZE
             09   12 044E  1067          BNEQ    10$                      ; NO
     03   38 A3   F0 0450  1068          INSV    IRP$L_MEDIA(R3),#LA$V_CONFIG,-   ; YES, STORE CONFIGURATION
     44 A5        0A 0454  1069                  #LA$S_CONFIG,UCB$L_DEVDEPEND(R5) ; BITS
             20   11 0457  1070          BRB     30$
                     0459  1071
                     0459  1072 10$:     ; IS IT A SET CLOCK (EITHER ONE)
        37   52   91 0459  1073          CMPB    R2,#IO$_SETCLOCK
             05   13 045C  1074          BEQL    20$                      ; YES
        05   52   91 045E  1075          CMPB    R2,#IO$_SETCLOCKP
             16   12 0461  1076          BNEQ    30$                      ; NO
                     0463  1077
                     0463  1078 20$:     ; IT'S A SET CLOCK.  ONLY SET CHARACTERISTICS IF CLOCK A WAS SET
 11 38 A3    04   E0 0463  1079          BBS     #4,IRP$L_MEDIA(R3),30$   ; BRANCH IF CLOCK B IS BEING SET
50   3A A3   FF 8F   78 0468  1080          ASHL    #-1,IRP$L_MEDIA+2(R3),R0  ; GET CLOCK A RATE IN LOW BITS OF R0
        0D   50   F0 046E  1081          INSV    R0,#LA$V_RATE,-          ; STORE RATE IN CHARACTERISTICS
     44 A5   03        0471  1082                  #LA$S_RATE,UCB$L_DEVDEPEND(R5)
                     0474  1083
                     0474  1084          ASSUME LA$V_PRESET EQ 16
```

```
3C A3   B0   0474  1085          MOVW    IRP$L_MEDIA+4(R3),-     ; STORE PRESET
46 A5        0477  1086                  UCB$L_DEVDEPEND+2(R5)
             0479  1087
        05   0479  1088 30$:     RSB
```

```
                                        .SBTTL  SDATA - START DATA PROCESSING
                                ;++
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ;       THIS ROUTINE PERFORMS THE PROCESSING NECESSARY FOR START DATA.
                                ;       IT ALLOCATES A BUFFERED DATAPATH (IF THE REQUEST IS A DEDICATED
                                ;       MODE REQUEST), ALLOCATES AND LOADS MAP REGISTERS FOR THE USW,
                                ;       BUFFERS, AND RCL AND BUILDS THE RDA FROM INFORMATION IN THE SIP.
                                ;
                                ; CALLING SEQUENCE:
                                ;
                                ;       BSBW    SDATA
                                ;
                                ; INPUT PARAMETERS:
                                ;
                                ;       R1      ADDRESS OF CRB
                                ;       R3      ADDRESS OF IRP
                                ;       R5      ADDRESS OF UCB
                                ;
                                ; OUTPUT PARAMETERS:
                                ;
                                ;       R0      COMPLETION CODE
                                ;
                                ; COMPLETION CODES:
                                ;
                                ;       SS$_NORMAL      NORMAL
                                ;       SS$_INSFMAPREG  INSUFFICIENT MAP REGISTERS
                                ;       SS$_INSFBUFDP   NO DATAPATHS AVAILABLE
                                ;
                                ; SIDE EFFECTS:
                                ;
                                ;       R2,R4 ARE DESTROYED
                                ;--
                                
54   48 A3   D0   047A 1125 SDATA:  MOVL    IRP$L_SIP(R3),R4        ; GET PTR TO SECONDARY I/O PACKET
                                
                                        ; IF A DEDICATED MODE TRANSFER, REQUEST A BUFFERED DATAPATH
13 64   03   E0   047E 1128         BBS     #3,SIP$W_MODE(R4),10$   ; BRANCH IF MULTI-REQUEST MODE
                                
00000000'GF   16   0482 1130         JSB     G^IOC$REQDATAPNW        ; DEDICATED MODE - GET A BDP
51   24 A5   D0   0488 1131         MOVL    UCB$L_CRB(R5),R1        ; RESTORE POINTER TO CRB
       75 50   E9   048C 1132         BLBC    R0,60$                  ; ALLOCATION FAILURE
       37 A1   89   048F 1133         BISB3   CRB$L_INTD+VEC$B_DATAPATH(R1),-  ; SAVE DATAPATH NUMBER AND
33 A4   20         0492 1134                 #VEC$M_LWAE,SIP$B_BFR_DATAP(R4)  ; SET LONGWORD ACCESS BIT
                                
                                        ; ALLOCATE AND LOAD MAP REGISTERS FOR BUFFERS, USW, AND RCL
                                        ASSUME  SIP$L_BFR_SVAPT EQ SIP$L_USW_SVAPT+12  ; USW MUST BE FIRST!
                                        ASSUME  SIP$L_RCL_SVAPT EQ SIP$L_BFR_SVAPT+12  ; RCL MUST BE LAST!
   54   1C   C0   0495 1139         ADDL    #SIP$C_USW_SVAPT,R4     ; POINT TO FIRST SVAPTE
        03   DD   0498 1140         PUSHL   #3
                                
78 A5   84   7D   049A 1142 15$:    MOVQ    (R4)+,UCB$L_SVAPTE(R5)  ; LOAD SVAPTE, BOFF, BCNT
        0F   13   049E 1143         BEQL    20$                     ; (ONLY IN CASE OF NO RCL - THIS
                                                                    ; WORKS ONLY IF RCL INFO. IS LAST)
37 A1   03 A4   90   04A0 1145         MOVB    3(R4),CRB$L_INTD+VEC$B_DATAPATH(R1)  ; LOAD DATAPATH #
         010A   30   04A5 1146         BSBW    SETMAPREG               ; ALLOCATE AND LOAD MAP REGISTERS
```

```
                55 50    E9  04AB  1147          BLBC    R0,50$                          ; ALLOCATION FAILURE
             84 34 A1    D0  04AB  1148          MOVL    CRB$L_INTD+VEC$W_MAPREG(R1),(R4)+  ; SAVE MAPREG, NUMREG
                E8 6E    F5  04AF  1149 20$:     SOBGTR  (SP),T5$
                         04B2  1150
                         04B2  1151          ; NOW BUILD THE RDA
             54 48 A3    D0  04B2  1152          MOVL    IRP$L_SIP(R3),R4        ; RESTORE POINTER TO BEGINNING OF SIP
          50 0164 C5     3E  04B6  1153          MOVAW   UCB$W_RDA(R5),R0        ; POINT TO RDA IN UCB
             80 64       7D  04BB  1154          MOVQ    SIP$W_MODE(R4),(R0)+    ; STORE MODE, BYTE COUNT, AND VALID
                         04BE  1155                                              ; BUFFER MASK IN RDA
             FA A0 02    A6  04BE  1156          DIVW2   #2,-6(R0)               ; CONVERT BYTE TO WORD COUNT IN RDA
                         04C2  1157
                         04C2  1158          ; INSERT USW ADDRESS
          FC A0 20 A4    B0  04C2  1159          MOVW    SIP$W_USW_BOFF(R4),-4(R0)  ; BYTE OFFSET
FC A0 09 09 24 A4        F0  04C7  1160          INSV    SIP$W_USW_MAPRE(R4),#9,#9,-4(R0)  ; PAGE NUMBER
                         04CE  1161
                         04CE  1162          ; NOW INSERT BUFFER ADDRESSES
                6E 07    D0  04CE  1163          MOVL    #7,(SP)
             52 02 A4    3C  04D1  1164          MOVZWL  SIP$W_BCNT(R4),R2       ; BUFFER LENGTH
             60 2C A4    3C  04D5  1165          MOVZWL  SIP$W_BFR_BOFF(R4),(R0) ; BYTE OFFSET
60 09 09 30 A4           F0  04D9  1166          INSV    SIP$W_BFR_MAPRE(R4),#9,#9,(R0)  ; FIRST BUFFER ADDRESS
                50 04    C0  04DF  1167          ADDL    #4,R0                   ; POINT TO SECOND BUFFER
             80 FC A0 52 C1  04E2  1168 40$:     ADDL3   R2,-4(R0),(R0)+         ; DO REMAINING 7 BUFFERS (ALWAYS CALC.
                F8 6E    F5  04E7  1169          SOBGTR  (SP),40$                ; ALL 8 BUFFERS EVEN IF THERE AREN'T
                         04EA  1170                                              ; THAT MANY).
                         04EA  1171
                         04EA  1172          ; NOW STORE RCL ADDRESS IF THERE IS ONE
             80 38 A4    3C  04EA  1173          MOVZWL  SIP$W_RCL_BOFF(R4),(R0)+  ; IF THERE IS NO RCL,
FC A0 09 09 3C A4        F0  04EE  1174          INSV    SIP$W_RCL_MAPRE(R4),#9,#9,-4(R0)  ; THIS STORES A ZERO
             80 0C A4    7D  04F5  1175          MOVQ    SIP$L_SLVDATA(R4),(R0)+ ; COPY REST OF RDA
             80 14 A4    7D  04F9  1176          MOVQ    SIP$L_SLVDATA+8(R4),(R0)+
                50 01    3C  04FD  1177          MOVZWL  S^#SS$_NORMAL,R0
                         0500  1178
                5E 04    C0  0500  1179 50$:     ADDL    #4,SP
                   05    05  0503  1180          RSB
                         0504  1181
                         0504  1182 60$:     ; NO DATAPATH
          50 033C 8F     3C  0504  1183          MOVZWL  #SS$_INSFBUFDP,R0
                   05    05  0509  1184          RSB
```

```
                                .SBTTL  REQUEST COMPLETE PROCESSING

                        ;++
                        ; FUNCTIONAL DESCRIPTION:
                        ;
                        ;       THIS ROUTINE RELEASES VARIOUS RESOURCES (UNLOCKS PAGES, RELEASES
                        ;       MAP REGISTERS AND DATAPATH, AND DEALLOCATES SIP) BEFORE SENDING
                        ;       AN I/O PACKET TO I/O POST PROCESSING.
                        ;       THIS ROUTINE ALSO DOES SOME STUFF FOR ERROR LOGGING AND DIAGNOSTICS
                        ;
                        ; CALLING SEQUENCE:
                        ;
                        ;       BSBW    REQ_COMPLETE
                        ;       BRW     REQ_COMPLETE
                        ;
                        ; INPUT PARAMETERS:
                        ;
                        ;       R0      FIRST LONGWORD OF I/O STATUS BLOCK
                        ;       R1      SECOND LONGWORD OF I/O STATUS BLOCK
                        ;               NOTE:   IF QIO IS A STOP, THEN STATUS IS ALREADY IN I/O PACKET
                        ;       R3      ADDRESS OF I/O PACKET
                        ;       R5      ADDRESS OF UCB
                        ;
                        ; OUTPUT PARAMETERS:
                        ;
                        ;       NONE
                        ;--

                        REQ_COMPLETE:
                3F  BB  PUSHR   #^M<R0,R1,R2,R3,R4,R5>
54  20 A3   C0 8F  8B  BICB3   #^XC0,IRP$W_FUNC(R3),R4 ; GET FUNCTION CODE
            00EC C5 7C  CLRQ    UCB$L_REGSAVE+8(R5)     ; CLEAR DATAPATH # AND REGISTER IN
                                                        ; REGISTER SAVE AREA

                        ; IF THIS IS A STOP REQUEST, THEN DON'T LOAD I/O STATUS
            54  03  91  CMPB    #IO$_STOP,R4            ; STOP REQUEST?
                04  13  BEQL    5$                      ; YES, DON'T LOAD STATUS
        38 A3   50  7D  MOVQ    R0,IRP$L_IOST1(R3)      ; NO, LOAD IOSB

                        5$:     ; GET POINTER TO CRB
        51  24 A5   D0  MOVL    UCB$L_CRB(R5),R1

                        ; IF THIS IS AN INITIALIZE QIO, RELEASE MAP REGISTERS POINTING TO RDA
            54  04  91  CMPB    #IO$_INITIALIZE,R4      ; INITIALIZE?
                08  12  BNEQ    10$                     ; NO
              40 A3 D0  MOVL    IRP$L_RDAMAPREG(R3),-   ; GET STARTING MAP # AND NUMBER OF
              34 A1                 CRB$L_INTD+VEC$W_MAPREG(R1)  ; REGISTERS AND MOVE INTO CRB
               0135 30  BSBW    REL_MRDP                ; RELEASE THEM

                        10$:    ; IF THIS WAS A START DATA OR STOP, GET POINTER TO SEC. I/O PACKET (SIP)
            54  38  91  CMPB    #IO$_STARTDATA,R4       ; START DATA?
                0A  13  BEQL    15$                     ; YES
            54  06  91  CMPB    #IO$_STARTDATAP,R4      ; START DATA PHYSICAL?
                05  13  BEQL    15$                     ; YES
            54  03  91  CMPB    #IO$_STOP,R4            ; STOP?
                27  12  BNEQ    30$                     ; NO
```

```
                54   48 A3    D0  053F  1244 15$:    MOVL    IRP$L_SIP(R3),R4         ; GET POINTER TO SIP
                                  0543  1245
                                  0543  1246         ; RELEASE MAP REGISTERS FOR USW, DATA BUFFERS, AND RCL.
                     24 A4    D0  0543  1247         MOVL    SIP$W_USW_MAPRE(R4),-    ; STARTING MAP REGISTER # AND NUMBER
                     34 A1        0546  1248                 CRB$L_INTD+VEC$W_MAPREG(R1)  ; OF REGISTERS FOR USW.
                        03    13  0548  1249         BEQL    16$                      ; NONE
                      0118    30  054A  1250         BSBW    REL_MRDP                 ; RELEASE USW MAP REGISTERS
                     30 A4    D0  054D  1251 16$:    MOVL    SIP$W_BFR_MAPRE(R4),-    ; SAME FOR DATA BUFFERS, BUT ALSO
                     34 A1        0550  1252                 CRB$L_INTD+VEC$W_MAPREG(R1)  ; INCLUDE BUFFERED DP #, IF ANY
                        03    13  0552  1253         BEQL    18$                      ; NONE
                      010E    30  0554  1254         BSBW    REL_MRDP                 ; RELEASE MAP REGISTERS AND DATAPATH
                     3C A4    D0  0557  1255 18$:    MOVL    SIP$W_RCL_MAPRE(R4),-    ; SAME FOR RCL, IF THERE IS ONE
                     34 A1        055A  1256                 CRB$L_INTD+VEC$W_MAPREG(R1)
                        03    13  055C  1257         BEQL    20$                      ; NONE
                      0104    30  055E  1258         BSBW    REL_MRDP                 ; RELEASE RCL MAP REGISTERS
                                  0561  1259
                                  0561  1260 20$:    ; NOW UNLOCK PAGES FOR USW, DATA BUFFERS, AND RCL AND DEALLOCATE SIP.
                55   54    D0  0561  1261         MOVL    R4,R5
                        0C    10  0564  1262         BSBB    UNLOCKF
                                  0566  1263
                                  0566  1264 30$:    ; DO ERROR LOGGING AND DIAGNOSTIC STUFF
                        3F    BA  0566  1265         POPR    #^M<R0,R1,R2,R3,R4,R5>
                      037C    30  0568  1266         BSBW    DODIAGERL
                                  056B  1267
                                  056B  1268         ; NOW QUEUE I/O PACKET FOR I/O POST PROCESSING
00000000'GF             16    056B  1269         JSB     G^COM$POST
                        05    0571  1270         RSB
```

```
                        0572  1272          .SBTTL  UNLOCK - UNLOCK PAGES AND DEALLOCATE SIP
                        0572  1273
                        0572  1274 ;++
                        0572  1275 ; FUNCTIONAL DESCRIPTION:
                        0572  1276 ;
                        0572  1277 ;       THIS ROUTINE UNLOCKS PAGES WHICH WERE LOCKED FOR A DATA TRANSFER
                        0572  1278 ;       AND DEALLOCATES THE SIP.  IT HAS TWO ENTRY POINTS: ONE SIMPLY
                        0572  1279 ;       UNLOCKS THE PAGES; THE OTHER FORKS (USING THE SIP AS A FORK BLOCK)
                        0572  1280 ;       BEFORE UNLOCKING THE PAGES.  PAGES ARE UNLOCKED FOR THE USW, THE
                        0572  1281 ;       DATA BUFFERS, AND THE RCL.
                        0572  1282 ;
                        0572  1283 ; CALLING SEQUENCE:
                        0572  1284 ;
                        0572  1285 ;       BSBW    UNLOCK          (DOESN'T FORK)
                        0572  1286 ;       BSBW    UNLOCKF         (FORKS)
                        0572  1287 ;
                        0572  1288 ; INPUT PARAMETERS:
                        0572  1289 ;
                        0572  1290 ;       R5      ADDRESS OF SIP
                        0572  1291 ;
                        0572  1292 ; OUTPUT PARAMEMTERS:
                        0572  1293 ;
                        0572  1294 ;       NONE
                        0572  1295 ;
                        0572  1296 ; SIDE EFFECTS:
                        0572  1297 ;
                        0572  1298 ;       R0 - R5 ARE NOT PRESERVED
                        0572  1299 ;--
                        0572  1300
                        0572  1301 UNLOCKF:  ; FORK ENTRY POINT
0B A5   06   90         0572  1302          MOVB    #IPL$_QUEUEAST,FKB$B_FIPL(R5)  ; LOAD FORK IPL
                        0576  1303          FORK
                        057C  1304
                        057C  1305 UNLOCK:  ; NO FORK ENTRY POINT
                        057C  1306
                        057C  1307          ; UNLOCK PAGES
          55   DD       057C  1308          PUSHL   R5                             ; SAVE POINTER TO BEGINNING OF SIP
     55   1C   C0       057E  1309          ADDL    #SIP$L_USW_SVAPT,R5            ; POINT TO FIRST SVAPTE
     54   03   D0       0581  1310          MOVL    #3,R4                          ; LOOP 3 TIMES (USW, DATA BUFFERS, RCL)
                        0584  1311
                        0584  1312 10$:     ; UNLOCK NEXT AREA
     53   65   D0       0584  1313          MOVL    (R5),R3                        ; GET SVAPTE
          19   13       0587  1314          BEQL    20$                            ; NOTHING THERE
   51   04 A5   3C       0589  1315          MOVZWL  4(R5),R1                       ; GET BOFF
   52   06 A5   3C       058D  1316          MOVZWL  6(R5),R2                       ; GET BCNT
51  01FF C142   9E       0591  1317          MOVAB   511(R1)[R2],R1                 ; COMBINE OFFSET AND COUNT AND ROUND
51  51   F7 8F   78       0597  1318          ASHL    #-VA$S_BYTE,R1,R1              ; CONVERT TO # OF PAGES (TO UNLOCK)
   00000000'GF   16       059C  1319          JSB     G^MMG$UNLOCK                   ; UNLOCK THEM
     55   0C   C0       05A2  1320 20$:     ADDL    #12,R5                         ; POINT TO NEXT SET OF INFO.
        DC 54   F5       05A5  1321          SOBGTR  R4,10$
                        05A8  1322
                        05A8  1323          ; NOW DEALLOCATE SIP
     50   8E   D0       05A8  1324          MOVL    (SP)+,R0                       ; GET POINTER TO BEGINNING OF SIP
   00000000'GF   16       05AB  1325          JSB     G^EXE$DEANONPAGED
                  05       05B1  1326          RSB
```

```
                        .SBTTL  SETMAPREG - ALLOCATE AND LOAD UBA MAP REGISTERS

                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       THIS ROUTINE ALLOCATES AND LOADS UBA MAPPING REGISTERS.
                ;       IF MAPPING REGISTERS WERE PREALLOCATED THEN THE ALLOCATION IS FROM
                ;       THE BITMAP IN THE UCB.  OTHERWISE THE ALLOCATION IS FROM THE BITMAP
                ;       IN THE ADP.
                ;
                ; CALLING SEQUENCE:
                ;
                ;       BSBW    SETMAPREG
                ;
                ; INPUT PARAMETERS:
                ;
                ;       R1      POINTS TO CRB
                ;       R5      POINTS TO UCB
                ;
                ; IMPLICIT INPUTS:
                ;
                ;       UCB$L_SVAPTE, UCB$W_BCNT, UCB$W_BOFF DESCRIBE THE AREA TO BE MAPPED
                ;       UCB$L_PREALLOC IS NON-ZERO IF MAP REGISTERS WERE PREALLOCATED
                ;       CRB$L_INTD+VEC$B_DATAPATH CONTAINS THE DATAPATH NUMBER TO USE
                ;
                ; OUTPUT PARAMETERS:
                ;
                ;       R0      CONTAINS A COMPLETION CODE (SEE BELOW)
                ;       R2      CONTAINS 18 BIT STARTING UNIBUS ADDRESS OF AREA MAPPED
                ;
                ; IMPLICIT OUTPUTS:
                ;
                ;       CRB$L_INTD+VEC$W_MAPREG CONTAINS STARTING MAP REGISTER NUMBER
                ;       CRB$L_INTD+VEC$B_NUMREG CONTAINS NUMBER OF MAPPING REGISTERS ALLOCATED
                ;
                ; COMPLETION CODES:
                ;
                ;       SS$_NORMAL      ALLOCATION WAS SUCCESSFUL
                ;       SS$_INSFMAPREG  ALLOCATION FAILED (INSUFFICIENT MAP REGISTERS)
                ;
                ; SIDE EFFECTS:
                ;
                ;       NONE
                ;
                ;--
                SETMAPREG:

                ; If map registers were preallocated, then we call local subroutine
                ;       ALLOC_LOCALMR to use some of preallocated registers.  Else we
                ;       use normal system subroutine to allocate from central pool.

00A8 C5   D5    TSTL    UCB$L_PREALLOC(R5)      ; ANY REGISTERS PREALLOCATED?
     04   13    BEQL    10$                     ; NO, PROCEED NORMALLY
     2D   10    BSBB    ALLOC_LOCALMR           ; Allocate from local pool.
     0A   11    BRB     20$                     ;  and branch around normal path.

                10$:    ; ALLOCATE MAPPING REGISTERS
```

```
                00000000'GF   16  05BC  1385          JSB     G^IOC$ALOUBAMAP
                 51    24 A5   D0  05C2  1386          MOVL    UCB$L_CRB(R5),R1        ; REFRESH R1 => CRB.
                                   05C6  1387 20$:
                       18 50   E9  05C6  1388          BLBC    R0,50$                  ; ALLOCATION FAILURE
                                   05C9  1389
                                   05C9  1390          ; LOAD UNIBUS MAPPING REGISTERS
                          12   BB  05C9  1391          PUSHR   #^M<R1,R4>
                00000000'GF   16  05CB  1392          JSB     G^IOC$LOADUBAMAP
                          12   BA  05D1  1393          POPR    #^M<R1,R4>
                                   05D3  1394
                                   05D3  1395          ; SET UP STARTING UNIBUS ADDRESS OF AREA MAPPED
                 52    7C A5   3C  05D3  1396          MOVZWL  UCB$W_BOFF(R5),R2       ; BYTE OFFSET IN PAGE (LOW 9 BITS)
          52  09   09    34 A1   F0  05D7  1397          INSV    CRB$L_INTD+VEC$W_MAPREG(R1),#9,#9,R2  ; HIGH 9 BITS
                                   05DD  1398
                    50    01   3C  05DD  1399          MOVZWL  S^#SS$_NORMAL,R0        ; SUCCESSFUL ALLOCATION
                                   05E0  1400          RSB
                                   05E1  1401
                                   05E1  1402
                                   05E1  1403 50$:     ; ALLOCATION FAILED
                 50    0344 8F   3C  05E1  1404          MOVZWL  #SS$_INSFMAPREG,R0      ; INSUFFICIENT MAP REGISTERS
                                   05E6  1405          RSB
```

```
                                05E7  1407          .SBTTL  ALLOCATE UBA MAP REGISTERS FROM LOCAL POOL
                                05E7  1408 ;+
                                05E7  1409 ; ALLOC_LOCALMR
                                05E7  1410 ;
                                05E7  1411 ; THIS ROUTINE IS CALLED TO ALLOCATE UBA MAP REGISTERS AND TO MARK THE ALLOCATION
                                05E7  1412 ; IN THE UBA MAP REGISTER ALLOCATION BITMAP MAINTAINED LOCALLY.
                                05E7  1413 ;
                                05E7  1414 ; INPUTS:
                                05E7  1415 ;
                                05E7  1416 ;       R5 = DEVICE UNIT UCB ADDRESS.
                                05E7  1417 ;
                                05E7  1418 ; OUTPUTS:
                                05E7  1419 ;
                                05E7  1420 ;       R0 = SUCCESS INDICATION.
                                05E7  1421 ;-
                                05E7  1422
                                05E7  1423 ALLOC_LOCALMR:                          ;ALLOCATE UBA MAP REGISTERS CRB SPECIFIED
                      7E    53  7D  05E7  1424          MOVQ    R3,-(SP)              ; Save R3 and R4.
                  53    7E A5  3C  05EA  1425          MOVZWL  UCB$W_BCNT(R5),R3     ;GET TRANSFER BYTE COUNT
                  54    7C A5  3C  05EE  1426          MOVZWL  UCB$W_BOFF(R5),R4     ;GET BYTE OFFSET IN PAGE
              53    03FF C344  9E  05F2  1427          MOVAB   ^X3FF(R3)[R4],R3      ;CALCULATE HIGHEST RELATIVE BYTE AND ROUND
          53    53    F7 8F  78  05F8  1428          ASHL    #-9,R3,R3             ;CALCULATE NUMBER OF MAP REGISTERS REQUIRED
                      50     D4  05FD  1429 5$:       CLRL    R0                    ;ASSUME ALLOCATION FAILURE
                  51    24 A5  D0  05FF  1430          MOVL    UCB$L_CRB(R5),R1      ;GET ADDRESS OF CRB
                  36 A1   53   90  0603  1431          MOVB    R3,CRB$L_INTD+VEC$B_NUMREG(R1) ;SET NUMBER OF MAP REGISTERS ALLOCATE
                      54     D4  0607  1432          CLRL    R4                    ;CLEAR STARTING BIT POSITION
              52    54    53   C1  0609  1433 10$:     ADDL3   R3,R4,R2              ;CALCULATE HIGHEST BIT IN REQUIRED SCAN
                01F0 8F    52   B1  060D  1434          CMPW    R2,#496               ;BEYOND END OF ALLOCATION BITMAP?
                           2D   14  0612  1435          BGTR    50$                   ;IF GTR YES
54    0124 C5    20        54   EA  0614  1436          FFS     R4,#32,UCB$W_MRBITMAP(R5),R4 ;FIND A SET BIT
                           EC   13  061B  1437          BEQL    10$                   ;IF EQL BIT NOT FOUND
                52    54    53   C1  061D  1438          ADDL3   R3,R4,R2              ;CALCULATE HIGH BIT FOR SUCCESSFUL ALLOCATIO
                   34 A1    54   B0  0621  1439          MOVW    R4,CRB$L_INTD+VEC$W_MAPREG(R1) ;SAVE STARTING BIT NUMBER
54    0124 C5    20        54   EB  0625  1440 20$:     FFC     R4,#32,UCB$W_MRBITMAP(R5),R4 ;FIND A CLEAR BIT
                           52   54  D1  062C  1441          CMPL    R4,R2                 ;ENOUGH SET BITS SCANNED OVER?
                           08   18  062F  1442          BGEQ    30$                   ;IF GEQ YES
          EE 0124 C5       54   E0  0631  1443          BBS     R4,UCB$W_MRBITMAP(R5),20$ ;IF SET, CONTINUE SCAN
                           D0   11  0637  1444          BRB     10$                   ;
                54    34 A1   3C  0639  1445 30$:     MOVZWL  CRB$L_INTD+VEC$W_MAPREG(R1),R4 ;RETRIEVE STARTING MAP REGISTER
                           06   10  063D  1446          BSBB    ALT_LOCALBITMAP       ;ALTER MAP REGISTER BITMAP
                           50   D6  063F  1447 40$:     INCL    R0                    ;SET SUCCESS INDICATOR
                53    8E   7D  0641  1448 50$:     MOVQ    (SP)+,R3              ;RESTORE REGISTERS
                           05   0644  1449          RSB                           ;
```

```
                                0645  1451          .SBTTL  ALTER LOCAL UBA MAP REGISTER BITMAP
                                0645  1452 ;+
                                0645  1453 ; ALT_LOCALBITMAP
                                0645  1454 ;
                                0645  1455 ; THIS ROUTINE IS CALLED TO EITHER CLEAR OR SET A FIELD OF BITS IN THE UBA MAP
                                0645  1456 ; REGISTER ALLOCATION BITMAP MAINTAINED LOCALLY IN THE UCB.
                                0645  1457 ;
                                0645  1458 ; INPUTS:
                                0645  1459 ;
                                0645  1460 ;       R0 = ALTERATION BIT MASK.
                                0645  1461 ;       R1 = ADDRESS OF CRB.
                                0645  1462 ;       R4 = STARTING MAP REGISTER NUMBER.
                                0645  1463 ;       R5 => UCB
                                0645  1464 ;
                                0645  1465 ; OUTPUTS:
                                0645  1466 ;
                                0645  1467 ;       THE SPECIFIED BIT FIELD IN THE UBA MAP ALLOCATION BIT MAP IS EITHER SET
                                0645  1468 ;       OR CLEARED DEPENDING ON THE STATE OF THE ALTERATION MASK.
                                0645  1469 ;
                                0645  1470 ;       R3 AND R4 ARE DESTROYED.
                                0645  1471 ;-
                                0645  1472
                                0645  1473 ALT_LOCALBITMAP:
          53    36 A1    9A     0645  1474          MOVZBL  CRB$L_INTD+VEC$B_NUMREG(R1),R3 ;GET NUMBER OF BITS TO ALTER
              53    20    D1    0649  1475 10$:     CMPL    #32,R3                  ;MORE THAN LONGWORD LEFT?
                    0F    18    064C  1476          BGEQ    20$                     ;IF GEQ NO
 0124 C5  20  54    50    F0    064E  1477          INSV    R0,R4,#32,UCB$W_MRBITMAP(R5) ;ALTER BITMAP WITH SUPPLIED PATTERN
              54    20    C0    0655  1478          ADDL    #32,R4                  ;UPDATE STARTING BIT POSITION
              53    20    C2    0658  1479          SUBL    #32,R3                  ;REDUCE NUMBER OF BITS TO ALTER
                    EC    11    065B  1480          BRB     10$                     ;
 0124 C5  53  54    50    F0    065D  1481 20$:     INSV    R0,R4,R3,UCB$W_MRBITMAP(R5) ;ALTER BITMAP WITH SUPPLIED PATTERN
                    05    0664  1482          RSB                             ;
```

```
                                                H 3
0665  1484          .SBTTL  REL_MRDP - RELEASE UBA MAP REGISTERS AND DATAPATH
0665  1485
0665  1486 ;++
0665  1487 ; FUNCTIONAL DESCRIPTION:
0665  1488 ;
0665  1489 ;       THIS ROUTINE RELEASES UBA MAP REGISTERS AND A BUFFERED
0665  1490 ;       DATAPATH IF ONE WAS ASSIGNED.  IF MAPPING REGISTERS
0665  1491 ;       WERE PREALLOCATED, THEN THEY ARE RELEASED INTO THE BITMAP IN THE
0665  1492 ;       UCB.  OTHERWISE, THEY ARE RELEASED INTO THE BITMAP IN THE ADP.
0665  1493 ;       IN THE LATTER CASE AN ATTEMPT IS MADE TO CALL ANY DRIVERS WAITING
0665  1494 ;       FOR MAP REGISTERS (ON THE ADP QUEUE).  BUFFERED DATAPATHS ARE
0665  1495 ;       ALWAYS RELEASED INTO THE ADP BITMAP BECAUSE THEY ARE NOT PREALLOCATED.
0665  1496 ;       ALSO, THE DATAPATH IS PURGED BEFORE IT IS RELEASED.
0665  1497 ;       ALSO, THE DATAPATH NUMBER AND DATAPATH REGISTER ARE COPIED INTO
0665  1498 ;       THE REGISTER SAVE AREA FOR DIAGNOSTICS AND ERROR LOGGING USE.
0665  1499 ;
0665  1500 ; CALLING SEQUENCE:
0665  1501 ;
0665  1502 ;       BSBW    REL_MRDP
0665  1503 ;
0665  1504 ; INPUT PARAMETERS:
0665  1505 ;
0665  1506 ;       R1      POINTS TO CRB
0665  1507 ;       R3      POINTS TO IRP
0665  1508 ;       R5      POINTS TO UCB
0665  1509 ;
0665  1510 ; IMPLICIT INPUTS:
0665  1511 ;
0665  1512 ;       UCB$L_PREALLOC IS NON-ZERO IF MAP REGISTERS WERE PREALLOCATED
0665  1513 ;       CRB$L_INTD+VEC$W_MAPREG CONTAINS THE STARTING MAP REGISTER NUMBER
0665  1514 ;       CRB$L_INTD+VEC$B_NUMREG CONTAINS NUMBER OF MAP REGISTERS TO RELEASE
0665  1515 ;       CRB$L_INTD+VEC$B_DATAPATH CONTAINS THE DATAPATH NUMBER (ZERO MEANS
0665  1516 ;               A BUFFERED DATAPATH WASN'T ALLOCATED).
0665  1517 ;
0665  1518 ; OUTPUT PARAMETERS:
0665  1519 ;
0665  1520 ;       NONE
0665  1521 ;
0665  1522 ; SIDE EFFECTS:
0665  1523 ;
0665  1524 ;       IF THERE IS A DATAPATH ERROR, THEN THE STATUS SS$_PARITY IS STORED
0665  1525 ;       IN THE I/O PACKET.
0665  1526 ;--
0665  1527
0665  1528 REL_MRDP:
      17   BB 0665  1529          PUSHR   #^M<R0,R1,R2,R4>
      53   DD 0667  1530          PUSHL   R3                      ; SAVE R3 SEPARATELY
  00A8 C5   D5 0669  1531          TSTL    UCB$L_PREALLOC(R5)      ; REGISTERS PREALLOCATED?
      0B   13 066D  1532          BEQL    10$                     ; NO
066F  1533
066F  1534          ; REGISTERS WERE PREALLOCATED SO SET UP TO ALTER BITMAP IN UCB.
54   34 A1   3C 066F  1535          MOVZWL  CRB$L_INTD+VEC$W_MAPREG(R1),R4  ; STARTING MAP REGISTER #
50   00   D2 0673  1536          MCOML   #0,R0                   ; ALTER PATTERN
      CD   10 0676  1537          BSBB    ALT_LOCALBITMAP         ; Alter local bit map.
      0A   11 0678  1538          BRB     20$
067A  1539
067A  1540 10$:     ; REGISTERS WERE NOT PREALLOCATED SO RETURN THEM TO ADP BITMAP
```

```
                00000000'GF   16   067A  1541          JSB     G^IOC$RELMAPREG
                 51    24 A5   D0   0680  1542          MOVL    UCB$L_CRB(R5),R1        ; RESTORE POINTER TO CRB
                                  0684  1543
                                  0684  1544 20$:       ; RELEASE DATAPATH IF ONE WAS ALLOCATED
                     53    8E   D0   0684  1545          MOVL    (SP)+,R3                ; RESTORE R3 (POINTER TO IRP)
                     05    00   EF   0687  1546          EXTZV   #VEC$V_DATAPATH,#VEC$S_DATAPATH,- ; EXTRACT DATAPATH NUMBER
                 52    37 A1        068A  1547                  CRB$L_INTD+VEC$B_DATAPATH(R1),R2  ; INTO R2
                           23   13   068D  1548          BEQL    30$                     ; NONE ALLOCATED
                                  068F  1549
                                  068F  1550          ; PURGE DATAPATH
                           0C   BB   068F  1551          PUSHR   #^M<R2,R3>              ; SAVE D.P. NUMBER AND IRP POINTER
                00000000'GF   16   0691  1552          JSB     G^IOC$PURGDATAP         ; RETURNS STATUS IN R0, D.P. REG. IN R1
                           0C   BA   0697  1553          POPR    #^M<R2,R3>
                         06 50  E8   0699  1554          BLBS    R0,25$                  ; NO TRANSMISSION ERROR
            38 A3    01F4 8F   3C   069C  1555          MOVZWL  #SS$_PARITY,IRP$L_IOST1(R3)     ; YES, RETURN ERROR STATUS
                                  06A2  1556
                                  06A2  1557 25$:       ; SAVE DATAPATH NUMBER AND CONTENTS OF DATAPATH REGISTER IN REGISTER
                                  06A2  1558          ; SAVE AREA
                00EC C5    52   D0   06A2  1559          MOVL    R2,UCB$L_REGSAVE+8(R5)  ; SAVE DATAPATH NUMBER
                00F0 C5    51   D0   06A7  1560          MOVL    R1,UCB$L_REGSAVE+12(R5) ; SAVE DATAPATH REGISTER
                                  06AC  1561
                00000000'GF   16   06AC  1562          JSB     G^IOC$RELDATAP          ; RELEASE DATAPATH
                                  06B2  1563
                           17   BA   06B2  1564 30$:       POPR    #^M<R0,R1,R2,R4>
                                05   06B4  1565          RSB
```

```
                     06B5  1567          .SBTTL  READY IN INTERRUPT SERVICE
                     06B5  1568
                     06B5  1569 ;++
                     06B5  1570 ; FUNCTIONAL DESCRIPTION:
                     06B5  1571 ;
                     06B5  1572 ;       THIS ROUTINE IS THE READY-IN INTERRUPT SERVICE ROUTINE.
                     06B5  1573 ;       ASSUMING THE INTERRUPT WAS EXPECTED, IT CALLS THE DRIVER AT
                     06B5  1574 ;       THE INTERRUPT WAIT ADDRESS AND THEN RETURNS.  UNEXPECTED
                     06B5  1575 ;       INTERRUPTS ARE IGNORED BY RETURNING IMMEDIATELY.
                     06B5  1576 ;
                     06B5  1577 ; CALLING SEQUENCE:
                     06B5  1578 ;
                     06B5  1579 ;       JSB FROM INTERRUPT VECTOR IN CRB
                     06B5  1580 ;
                     06B5  1581 ; INPUT PARAMETERS:
                     06B5  1582 ;
                     06B5  1583 ;       NONE
                     06B5  1584 ;
                     06B5  1585 ; IMPLICIT INPUTS:
                     06B5  1586 ;
                     06B5  1587 ;       THE STACK ON ENTRY IS AS FOLLOWS:
                     06B5  1588 ;
                     06B5  1589 ;                 0(SP)          ADDRESS OF IDB ADDRESS
                     06B5  1590 ;       4(SP) - 24(SP)          SAVED R0 - R5
                     06B5  1591 ;                28(SP)          INTERRUPT PC
                     06B5  1592 ;                32(SP)          INTERRUPT PSL
                     06B5  1593 ;
                     06B5  1594 ; OUTPUT PARAMETERS:
                     06B5  1595 ;
                     06B5  1596 ;       NONE
                     06B5  1597 ;--
                     06B5  1598
                     06B5  1599
                     06B5  1600 LA$RDYININTSV::
         53   9E   D0  06B5  1601          MOVL    @(SP)+,R3               ; GET ADDRESS OF IDB
                     06B8  1602          ASSUME  IDB$L_CSR+4 EQ IDB$L_OWNER
         54   63   7D  06B8  1603          MOVQ    IDB$L_CSR(R3),R4        ; CSR -> R4;   UCB -> R5
                     06BB  1604
   1E 64 A5   01   E5  06BB  1605          BBCC    #UCB$V_INT,UCB$W_STS(R5),INTEXIT  ; IF CLR, INT. NOT EXPECTED
                     06C0  1606
                     06C0  1607          ; COPY LPA-11 REGISTERS INTO READY-IN INTERRUPT SAVE AREA
    00F4 C5    64   B0  06C0  1608          MOVW    LA_CISR(R4),UCB$W_RISAVE(R5)
00F6 C5    02 A4   B0  06C5  1609          MOVW    LA_COSR(R4),UCB$W_RISAVE+2(R5)
00F8 C5    04 A4   B0  06CB  1610          MOVW    LA_RDA(R4),UCB$W_RISAVE+4(R5)
00FA C5    06 A4   B0  06D1  1611          MOVW    LA_MAINT(R4),UCB$W_RISAVE+6(R5)
                     06D7  1612
         53   10 A5   7D  06D7  1613          MOVQ    UCB$L_FR3(R5),R3        ; RESTORE DRIVER CONTEXT
              0C B5   16  06DB  1614          JSB     @UCB$L_FPC(R5)          ; CALL DRIVER AT INTERRUPT WAIT ADDRESS
                     06DE  1615
                     06DE  1616 INTEXIT:
         50   8E   7D  06DE  1617          MOVQ    (SP)+,R0                ; RESTORE REGISTERS
         52   8E   7D  06E1  1618          MOVQ    (SP)+,R2
         54   8E   7D  06E4  1619          MOVQ    (SP)+,R4
                   02  06E7  1620          REI
```

```
                     06E8  1622          .SBTTL  READY OUT INTERRUPT SERVICE
                     06E8  1623
                     06E8  1624 ;++
                     06E8  1625 ; FUNCTIONAL DESCRIPTION:
                     06E8  1626 ;
                     06E8  1627 ;       THIS ROUTINE IS THE READY-OUT INTERRUPT SERVICE ROUTINE.
                     06E8  1628 ;       AFTER RECEIVING THE INTERRUPT, THIS ROUTINE FORKS, DETERMINES
                     06E8  1629 ;       THE CAUSE OF THE INTERRUPT, AND DISPATCHES TO AN APPROPRIATE
                     06E8  1630 ;       ROUTINE.  THERE ARE BASICALLY FOUR CASES:
                     06E8  1631 ;               1)  NO ERROR
                     06E8  1632 ;                       A)  START REQUEST PROCESSED
                     06E8  1633 ;                       B)  BUFFER FULL OR EMPTY
                     06E8  1634 ;                       C)  BUFFER OVER/UNDERRUN
                     06E8  1635 ;               2)  COMMAND ERROR
                     06E8  1636 ;               3)  USER REQUEST ERROR (DURING A DATA TRANSFER)
                     06E8  1637 ;               4)  FATAL HARDWARE ERROR
                     06E8  1638 ;
                     06E8  1639 ; CALLING SEQUENCE:
                     06E8  1640 ;
                     06E8  1641 ;       JSB FROM INTERRUPT VECTOR IN CRB
                     06E8  1642 ;
                     06E8  1643 ; INPUT PARAMETERS:
                     06E8  1644 ;
                     06E8  1645 ;       NONE
                     06E8  1646 ;
                     06E8  1647 ; IMPLICIT INPUTS:
                     06E8  1648 ;
                     06E8  1649 ;       THE STACK ON ENTRY IS AS FOLLOWS:
                     06E8  1650 ;
                     06E8  1651 ;                 0(SP)          ADDRESS OF IDB ADDRESS
                     06E8  1652 ;       4(SP) - 24(SP)          SAVED R0 - R5
                     06E8  1653 ;                28(SP)          INTERRUPT PC
                     06E8  1654 ;                32(SP)          INTERRUPT PSL
                     06E8  1655 ;
                     06E8  1656 ; OUTPUT PARAMETERS:
                     06E8  1657 ;
                     06E8  1658 ;       NONE
                     06E8  1659 ;--
                     06E8  1660
                     06E8  1661 LA$RDYOUTINTSV::
          53   9E    D0  06E8  1662          MOVL    @(SP)+,R3               ; GET ADDRESS OF IDB
                     06EB  1663          ASSUME  IDB$L_CSR+4 EQ IDB$L_OWNER
          54   63    7D  06EB  1664          MOVQ    IDB$L_CSR(R3),R4        ; CSR -> R4;   UCB -> R5
                     06EE  1665
                     06EE  1666          ; COPY LPA-11 REGISTERS INTO READY-OUT INTERRUPT SAVE AREA
    00FC C5    64    B0  06EE  1667          MOVW    LA_CISR(R4),UCB$W_ROSAVE(R5)
00FE C5    02 A4    B0  06F3  1668          MOVW    LA_COSR(R4),UCB$W_ROSAVE+2(R5)
0100 C5    04 A4    B0  06F9  1669          MOVW    LA_RDA(R4),UCB$W_ROSAVE+4(R5)
0102 C5    06 A4    B0  06FF  1670          MOVW    LA_MAINT(R4),UCB$W_ROSAVE+6(R5)
                     0705  1671
        D6 AF    9F  0705  1672          PUSHAB  INTEXIT                 ; ADDRESS TO RETURN TO AFTER FORK
   55   00B4 C5    DE  0708  1673          MOVAL   UCB$L_FORKO(R5),R5      ; HAVE TO USE DIFFERENT FORK BLOCK THAN
                     070D  1674          FORK                            ; READY IN INTERRUPTS USE.
                     0713  1675
   55   FF4C C5    DE  0713  1676          MOVAL   -UCB$L_FORKO(R5),R5     ; RESTORE POINTER TO UCB
                     0718  1677
                     0718  1678          ; COPY LPA-11 REGISTERS FROM INTERRUPT SAVE AREA TO COMMON SAVE AREA
```

```
                00E4 C5    00FC C5    7D   0718  1679          MOVQ    UCB$W_ROSAVE(R5),UCB$L_REGSAVE(R5)
                                         071F  1680
                                         071F  1681          ; GET CONTENTS OF CONTROL OUT STATUS REGISTER, AND MAINTENANCE REGISTER
                                         071F  1682          ; AND THEN ACKNOWLEGE INTERRUPT (WHICH ALLOWS THE NEXT READY OUT
                                         071F  1683          ; INTERRUPT TO OCCUR)
                      50    02 A4    3C   071F  1684          MOVZWL  LA_COSR(R4),R0          ; CONTROL OUT STATUS
                      51    06 A4    3C   0723  1685          MOVZWL  LA_MAINT(R4),R1         ; MAINTENANCE REGISTER
                   02 A4    0080 8F  AA   0727  1686          BICW2   #LA_COSR_M_RDY,LA_COSR(R4)  ; ACKNOWLEGE INTERRUPT
                                         072D  1687
                                         072D  1688          ; PUT BOTH LPA-11 REGISTERS INTO R1 TO BE USED AS SECOND
                                         072D  1689          ; LONGWORD OF IOSB IN CASE OF ERROR.
                      51    51    10   78   072D  1690          ASHL    #16,R1,R1               ; PUT MAINT. REGISTER IN HIGH WORD
                            51    50   B0   0731  1691          MOVW    R0,R1                   ; PUT CONTROL OUT STATUS IN LOW WORD
                                         0734  1692
                                         0734  1693          ; GET USER # IN R2 AND DETERMINE IF THIS IS AN ERROR
                52  50    FFFFFFF8 8F   CB   0734  1694          BICL3   #^XFFFFFFF8,R0,R2       ; GET USER INDEX IN R2
                     50    50    F8 8F   78   073C  1695          ASHL    #-8,R0,R0               ; PUT STATUS ON LOW BYTE
                                  50   95   0741  1696          TSTB    R0                      ; ERROR?
                                  03   19   0743  1697          BLSS    ERROR                   ; YES
                                0079   31   0745  1698          BRW     NO_ERROR                ; NO
                                         0748  1699
                                         0748  1700 ;
                                         0748  1701 ;       E R R O R
                                         0748  1702 ;
                                         0748  1703 ERROR:  ; SOME SORT OF ERROR -  DETERMINE WHAT TYPE AND DISPATCH TO
                                         0748  1704          ; APPROPRIATE ROUTINE.  ERROR TYPE IS SPECIFIED BY FIELD
                                         0748  1705          ; LA_COSR_V_ERRTP WHICH HAS BEEN SHIFTED 8 BITS TO THE RIGHT IN R0
                   02   50    02    05   ED   0748  1706          CMPZV   #LA_COSR_V_ERRTP-8,#LA_COSR_S_ERRTP,R0,#2
                                  3F   19   074D  1707          BLSS    REQERR                  ; USER REQUEST ERROR
                                  60   13   074F  1708          BEQL    CMDERR                  ; COMMAND ERROR
                                         0751  1709
                                         0751  1710          ; FALL THROUGH TO ...
                                         0751  1711
                                         0751  1712 ;
                                         0751  1713 ;       F A T A L   H A R D W A R E   E R R O R
                                         0751  1714 ;
           50   0054 8F    3C   0751  1715          MOVZWL  #SS$_CTRLERR,R0         ; STATUS
                     15    11   0756  1716          BRB     COMPL_ALL_REQS
                                         0758  1717
                                         0758  1718 ;
                                         0758  1719 ;       T I M E O U T   O R   P O W E R F A I L
                                         0758  1720 ;
                                         0758  1721 TIMEOUT:  ; DEVICE TIMEOUT AND POWERFAIL COME HERE (AT DEVICE IPL).
                                         0758  1722          SETIPL  UCB$B_FIPL(R5)          ; LOWER TO FORK IPL
           50   0364 8F    3C   075C  1723          MOVZWL  #SS$_POWERFAIL,R0       ; ASSUME POWERFAIL
                     51    D4   0761  1724          CLRL    R1                      ; CLEAR SECOND LONGWORD OF IOSB
                     05    E0   0763  1725          BBS     #UCB$V_POWER,-          ; BRANCH IF POWERFAIL
                05 64 A5         0765  1726                  UCB$W_STS(R5),COMPL_ALL_REQS
           50   022C 8F    3C   0768  1727          MOVZWL  #SS$_TIMEOUT,R0         ; MUST BE TIMEOUT
                                         076D  1728
                                         076D  1729 COMPL_ALL_REQS:  ; COMPLETE ALL OUTSTANDING I/O REQUESTS
             53    58 A5    D0   076D  1730          MOVL    UCB$L_IRP(R5),R3        ; GET CURRENT I/O REQUEST PACKET
                     06    13   0771  1731          BEQL    10$                     ; THERE ISN'T ONE
                   58 A5    D4   0773  1732          CLRL    UCB$L_IRP(R5)           ; CLEAR CURRENT I/O PACKET
                    FD91    30   0776  1733          BSBW    REQ_COMPLETE            ; SEND IT TO REQUEST COMPLETE
                                         0779  1734
                                         0779  1735 10$:     ; NOW COMPLETE ALL OUTSTANDING DATA TRANSFER REQUESTS
```

```
                0259   30  0779  1736          BSBW    COMPLETE_ALL
                           077C  1737
                           077C  1738          ; DO A DEVICE RESET (MASTER CLEAR) TO STOP MICROPROCESSOR
                           077C  1739          DSBINT  UCB$B_DIPL(R5)              ; RAISE IPL TO DEVICE LEVEL
      64  4000 8F    B0  0783  1740          MOVW    #LA_CISR_M_RESET,LA_CISR(R4)  ; RESET
                           0788  1741          ENBINT                              ; LOWER IPL
                           078B  1742
                           078B  1743          ; REQUESTS ON THE INPUT QUEUE ARE STARTED IN THE NORMAL FASHION.
                           078B  1744          ; HOWEVER, THEY ARE EXPECTED TO TIMEOUT.
                FCA3   31  078B  1745          BRW     STRT_NXT_REQ                ; START NEXT REQUEST.
                           078E  1746
                           078E  1747
                           078E  1748 ;
                           078E  1749 ;        U S E R   R E Q U E S T   E R R O R
                           078E  1750 ;
                           078E  1751 REQERR: ; USER REQUEST ERROR
      53  0104 C542   D0  078E  1752          MOVL    UCB$L_RQLIST(R5)[R2],R3 ; GET POINTER TO I/O PACKET
                    1A   13  0794  1753          BEQL    30$                         ; CAN HAPPEN IF STOP HAS BEEN QUEUED
                           0796  1754                                              ; FOR THIS REQUEST
                0104 C542   D4  0796  1755          CLRL    UCB$L_RQLIST(R5)[R2]    ; CLEAR SLOT
             50   A8 8F    91  079B  1756          CMPB    #^O250,R0               ; STOPPED BY USW REQUEST?
                    07   13  079F  1757          BEQL    10$                         ; YES
           50   0334 8F    3C  07A1  1758          MOVZWL  #SS$_DEVREQERR,R0       ; NO - ERROR.  LOAD STATUS RETURN
                    05   11  07A6  1759          BRB     20$
                           07A8  1760
                           07A8  1761 10$:     ; STOPPED BY USW REQUEST
                50   01   3C  07A8  1762          MOVZWL  S^#SS$_NORMAL,R0        ; RETURN NORMAL STATUS
                    51   D4  07AB  1763          CLRL    R1                      ; CLEAR SECOND LONGWORD OF IOSB
                FD5A   30  07AD  1764 20$:     BSBW    REQ_COMPLETE
                    05   10  07B0  1765 30$:     RSB
                           07B1  1766
                           07B1  1767 ;
                           07B1  1768 ;        C O M M A N D   E R R O R
                           07B1  1769 ;
                           07B1  1770 CMDERR: ; COMMAND ERROR
        53   58 A5    D0  07B1  1771          MOVL    UCB$L_IRP(R5),R3        ; GET POINTER TO CURRENT PACKET
             58 A5    D4  07B5  1772          CLRL    UCB$L_IRP(R5)           ; CLEAR CURRENT PACKET ENTRY
      50   032C 8F    3C  07B8  1773          MOVZWL  #SS$_DEVCMDERR,R0       ; STATUS RETURN
                FD4A   30  07BD  1774          BSBW    REQ_COMPLETE
                    05   10  07C0  1775          RSB
                           07C1  1776
                           07C1  1777
                           07C1  1778 ;
                           07C1  1779 ;        N O   E R R O R
                           07C1  1780 ;
                           07C1  1781 NO_ERROR:  ; COME HERE IF THE INTERRUPT WAS NOT DUE TO AN ERROR.
                           07C1  1782          ; THERE ARE THREE CASES:
                           07C1  1783          ;       R0 = 0          START REQUEST PROCESSED
                           07C1  1784          ;       R0 = 1          NORMAL BUFFER FULL/EMPTY
                           07C1  1785          ;       R0 = 2          BUFFER OVER/UNDERRUN
                           07C1  1786          ; NOTE:  WHEN WE GET HERE R0 HAS JUST BEEN TESTED.
                           07C1  1787
                    32   12  07C1  1788          BNEQ    BFRFULL                 ; BUFFER FULL OR OVER/UNDERRUN
                           07C3  1789
                           07C3  1790 ;
                           07C3  1791 ;        S T A R T   R E Q U E S T   P R O C E S S E D
                           07C3  1792 ;
```

```
                        07C3  1793 STARTREQ:
                        07C3  1794          ; START REQUEST PROCESSED
          53   58 A5   D0  07C3  1795          MOVL    UCB$L_IRP(R5),R3          ; GET POINTER TO I/O PACKET
          54   48 A3   D0  07C7  1796          MOVL    IRP$L_SIP(R3),R4          ; GET POINTER TO SIP
             64   03   90  07CB  1797          MOVB    #STOP_MODE,SIP$W_MODE(R4)  ; BUILD STOP RDA IN SIP
          01 A4   52   90  07CE  1798          MOVB    R2,SIP$W_MODE+1(R4)       ; USER #
       0104 C542   53   D0  07D2  1799          MOVL    R3,UCB$L_RQLIST(R5)[R2] ; STORE ENTRY IN REQUEST LIST
             58 A5   D4  07D8  1800          CLRL    UCB$L_IRP(R5)             ; NO LONGER CURRENT PACKET
 6C A5  0000C000A'8F   C0  07DB  1801          ADDL    I^#10,UCB$L_DUETIM(R5)    ; ADD 10 SECONDS TO DUE TIME TO PREVENT
                        07E3  1802                                            ; TIMEOUTS IN DEDICATED MODE WITH
                        07E3  1803                                            ; SLOW TRANSFERS.
                        07E3  1804
                        07E3  1805          ; NOW CHECK TO SEE IF THIS REQUEST HAS BEEN CANCELED
          50   2C   3C  07E3  1806          MOVZWL  #SS$_ABORT,R0             ; ASSUME IT HAS
          64 A5   03   E0  07E6  1807          BBS     #UCB$V_CANCEL,UCB$W_STS(R5),-   ; BRANCH IF IT HAS BEEN CANCELED
                  25        07EA  1808                  QUEUE_STOP_REQ
                        07EB  1809
                        07EB  1810 10$:     ; NOW SIGNAL THAT REQUEST WAS STARTED
          54   3C A3   D0  07EB  1811          MOVL    IRP$L_BFR_AST(R3),R4      ; USE BUFFER FULL AST ADDRESS
                  4E   10  07EF  1812          BSBB    SIGNAL_BFR_FULL
                1C 50   E9  07F1  1813          BLBC    R0,QUEUE_STOP_REQ         ; ERROR
                        05  07F4  1814          RSB
                        07F5  1815
                        07F5  1816
                        07F5  1817
                        07F5  1818 ;
                        07F5  1819 ;       B U F F E R   F U L L   O R   O V E R / U N D E R R U N
                        07F5  1820 ;
                        07F5  1821 BFRFULL:
                        07F5  1822          ; BUFFER FULL OR EMPTY (AND POSSIBLY OVER/UNDERRUN)
 53   0104 C542   D0  07F5  1823          MOVL    UCB$L_RQLIST(R5)[R2],R3 ; GET POINTER TO I/O PACKET
             12   13  07FB  1824          BEQL    30$                       ; CAN HAPPEN IF STOP HAS BEEN QUEUED
       54   3C A3   D0  07FD  1825          MOVL    IRP$L_BFR_AST(R3),R4      ; GET BUFFER FULL AST ADDRESS
          01   50   91  0801  1826          CMPB    R0,#1                     ; BUFFER OVER/UNDERRUN?
             04   13  0804  1827          BEQL    20$                       ; NO
       54   40 A3   D0  0806  1828          MOVL    IRP$L_OVR_AST(R3),R4      ; YES, GET BFR OVER/UNDERRUN AST ADDRESS
             33   10  080A  1829 20$:     BSBB    SIGNAL_BFR_FULL
          01 50   E9  080C  1830          BLBC    R0,QUEUE_STOP_REQ         ; ERROR
                        05  080F  1831 30$:     RSB
```

```
                     0810  1833          .SBTTL  QUEUE_STOP_REQ - QUEUE A STOP REQUEST
                     0810  1834
                     0810  1835 ;++
                     0810  1836 ; FUNCTIONAL DESCRIPTION:
                     0810  1837 ;
                     0810  1838 ;       THIS ROUTINE TAKES AN I/O PACKET, CHANGES THE FUNCTION CODE TO
                     0810  1839 ;       STOP, AND QUEUES THE PACKET TO THE DRIVER (AT THE HEAD OF THE
                     0810  1840 ;       QUEUE).  IF THE DRIVER IS NOT BUSY, IT IS CALLED IMMEDIATELY.
                     0810  1841 ;       IT IS ASSUMED THAT THE STOP RDA HAS ALREADY BEEN BUILT IN THE PACKET.
                     0810  1842 ;       NOTE:  THIS ROUTINE MUST BE CALLED AT DRIVER FORK LEVEL.
                     0810  1843 ;
                     0810  1844 ; CALLING SEQUENCE:
                     0810  1845 ;
                     0810  1846 ;       BSBW    QUEUE_STOP_REQ   OR
                     0810  1847 ;       BRW     QUEUE_STOP_REQ
                     0810  1848 ;
                     0810  1849 ; INPUT PARAMETERS:
                     0810  1850 ;
                     0810  1851 ;       R0      FIRST LONGWORD OF I/O STATUS BLOCK
                     0810  1852 ;       R2      USER INDEX
                     0810  1853 ;       R5      POINTER TO UCB
                     0810  1854 ;
                     0810  1855 ; OUTPUT PARAMETERS:
                     0810  1856 ;
                     0810  1857 ;       NONE
                     0810  1858 ;--
                     0810  1859
                     0810  1860 QUEUE_STOP_REQ:
                3F   BB 0810  1861          PUSHR   #^M<R0,R1,R2,R3,R4,R5>
53   0104 C542      D0 0812  1862          MOVL    UCB$L_RQLIST(R5)[R2],R3 ; GET POINTER TO I/O PACKET
                22   13 0818  1863          BEQL    40$                     ; PACKET ALREADY WENT AWAY
     0104 C542      D4 081A  1864          CLRL    UCB$L_RQLIST(R5)[R2]    ; CLEAR SLOT
  20 A3    03      90 081F  1865          MOVB    #IO$_STOP,IRP$W_FUNC(R3)  ; STORE STOP FUNCTION CODE IN IRP
  38 A3    50      D0 0823  1866          MOVL    R0,IRP$L_IOST1(R3)      ; STORE STATUS CODE IN IOSB
        3C A3      D4 0827  1867          CLRL    IRP$L_IOST2(R3)         ; CLEAR SECOND LONGWORD
                     082A  1868
                     082A  1869          ; REQUEUE PACKET IN FRONT IF I/O QUEUE (OR IF NOT BUSY, HANDLE IT NOW)
08 64 A5     08      E2 082A  1870          BBSS    #UCB$V_BSY,UCB$W_STS(R5),30$  ; SET BUSY; WAS IT ALREADY SET?
  00000000'GF        16 082F  1871          JSB     G^IOC$INITIATE          ; NO, START DRIVER GOING
                05   11 0835  1872          BRB     40$
                     0837  1873
                     0837  1874 30$:       ; DRIVER IS BUSY.  QUEUE PACKET
 00AC C5     63      0E 0837  1875          INSQUE  IRP$L_IOQFL(R3),UCB$L_INQFL(R5)
                3F   BA 083C  1876 40$:       POPR    #^M<R0,R1,R2,R3,R4,R5>
                05   05 083E  1877          RSB
```

```
                        083F  1879          .SBTTL  SIGNAL_BFR_FULL - SIGNAL BUFFER FULL (OR EMPTY) TO USER
                        083F  1880
                        083F  1881 ;++
                        083F  1882 ; FUNCTIONAL DESCRIPTION:
                        083F  1883 ;
                        083F  1884 ;       THIS ROUTINE SIGNALS A USER PROCESS THAT A BUFFER HAS BEEN FILLED
                        083F  1885 ;       OR EMPTIED.  SIGNALING IS DONE BY SETTING AN EVENT FLAG OR
                        083F  1886 ;       ISSUING AN AST OR BOTH.  NOTE THAT THE SIGNALING IS DONE
                        083F  1887 ;       AFTER A FORK HAS BEEN PERFORMED.
                        083F  1888 ;
                        083F  1889 ; CALLING SEQUENCE:
                        083F  1890 ;
                        083F  1891 ;       BSBB    SIGNAL_BFR_FULL
                        083F  1892 ;
                        083F  1893 ; INPUT PARAMETERS:
                        083F  1894 ;
                        083F  1895 ;       R3      ADDRESS OF I/O PACKET
                        083F  1896 ;       R4      (USER) AST ADDRESS OR ZERO WHICH MEANS DON'T GIVE AN AST
                        083F  1897 ;       R5      ADDRESS OF UCB
                        083F  1898 ;
                        083F  1899 ; IMPLICIT INPUTS:
                        083F  1900 ;
                        083F  1901 ;       VARIOUS FIELDS IN THE I/O PACKET
                        083F  1902 ;
                        083F  1903 ; OUTPUT PARAMETERS:
                        083F  1904 ;
                        083F  1905 ;       R0      COMPLETION CODE
                        083F  1906 ;
                        083F  1907 ; COMPLETION CODES:
                        083F  1908 ;
                        083F  1909 ;       SS$_NORMAL      NORMAL SUCCESSFUL COMPLETION
                        083F  1910 ;       SS$_INSFMEM     INSUFFICIENT DYNAMIC MEMORY
                        083F  1911 ;       SS$_EXQUOTA     EXCEEDED AST QUOTA
                        083F  1912 ;
                        083F  1913 ; SIDE EFFECTS:
                        083F  1914 ;
                        083F  1915 ;       R1 IS NOT PRESERVED
                        083F  1916 ;--
                        083F  1917
                        083F  1918 SIGNAL_BFR_FULL:
                  24    BB  083F  1919          PUSHR   #^M<R2,R5>              ; SAVE REGISTERS HERE SO R5 CAN BE
                  03    10  0841  1920          BSBB    5$                      ; RESTORED AFTER FORK
                  24    BA  0843  1921          POPR    #^M<R2,R5>
                  05        0845  1922          RSB
                            0846  1923
                            0846  1924 5$:      ; MAKE SURE THERE IS ENOUGH AST QUOTA TO ALLOCATE A FORK/AST BLOCK
              55  0C A3    3C  0846  1925          MOVZWL  IRP$L_PID(R3),R5        ; GET PROCESS INDEX
      00000000'GF       DD  084A  1926          PUSHL   G^SCH$GL_PCBVEC         ; PUSH ADDRESS OF PCB TABLE
              55    9E45    D0  0850  1927          MOVL    @(SP)+[R5],R5           ; GET PCB ADDRESS
              50      1C    3C  0854  1928          MOVZWL  #SS$_EXQUOTA,R0         ; ASSUME ERROR
                38 A5       B5  0857  1929          TSTW    PCB$W_ASTCNT(R5)        ; ENOUGH AST QUOTA LEFT?
                    1E      15  085A  1930          BLEQ    10$                     ; NO
                38 A5       B7  085C  1931          DECW    PCB$W_ASTCNT(R5)        ; YES, TAKE ONE AWAY
                            085F  1932
                            085F  1933          ; ALLOCATE A PACKET TO BE USED AS A FORK BLOCK AND AST CONTROL BLOCK
          51  00C4 8F      3C  085F  1934          MOVZWL  #IRP$C_LENGTH,R1        ; LENGTH = AN I/O PACKET
                    53      DD  0864  1935          PUSHL   R3                      ; SAVE R3
```

```
                00000000'GF   16  0866  1936          JSB     G^EXE$ALONONPAGED               ; RETURNS POINTER TO PACKET IN R2
                     53   8E   D0  086C  1937          MOVL    (SP)+,R3                        ; RESTORE R3
                       09 50   E8  086F  1938          BLBS    R0,20$                          ; SUCCESSFUL ALLOCATION
                50   0124 8F   3C  0872  1939          MOVZWL  #SS$_INSFMEM,R0                 ; ERROR - INSUFFICIENT DYNAMIC MEMORY
                       38 A5   B6  0877  1940          INCW    PCB$W_ASTCNT(R5)                ; ADD 1 BACK TO AST QUOTA
                               05  087A  1941 10$:      RSB                                     ; ERROR RETURN
                                   087B  1942
                                   087B  1943 20$:      ; PUT SIZE AND TYPE INTO PACKET
                                   087B  1944          ASSUME  IRP$B_TYPE EQ IRP$W_SIZE+2
          08 A2     000200C4 8F   D0  087B  1945          MOVL    #<DYN$C_ACB@16>+IRP$C_LENGTH,IRP$W_SIZE(R2)
                                   0883  1946
                                   0883  1947          ; NOW FORK (AND RETURN STATUS TO CALLER)
                                   0883  1948          ASSUME FKB$B_FIPL EQ IRP$B_RMOD
                    0B A2   06     90  0883  1949          MOVB    #IPL$_QUEUEAST,FKB$B_FIPL(R2)  ; SET FORK IPL
                       55   52     D0  0887  1950          MOVL    R2,R5
                       50   01     3C  088A  1951          MOVZWL  S^#SS$_NORMAL,R0                ; RETURN NORMAL STATUS TO CALLER
                                   088D  1952          FORK
                                   0893  1953
                                   0893  1954          ; SET VARIOUS VALUES IN AST CONTROL BLOCK IN PREPARATION FOR
                                   0893  1955          ; QUEUING AST
                    51    0C A3    D0  0893  1956          MOVL    IRP$L_PID(R3),R1                ; SAVE PID FOR CALL TO SCH$POSTEF
                    0C A5   51     D0  0897  1957          MOVL    R1,ACB$L_PID(R5)                ; PID
                    10 A5   54     D0  089B  1958          MOVL    R4,ACB$L_AST(R5)                ; AST ADDRESS
                            0F     13  089F  1959          BEQL    30$                             ; NO AST
                 0B A5    0B A3    90  08A1  1960          MOVB    IRP$B_RMOD(R3),ACB$B_RMOD(R5)   ; ACCESS MODE
                 0B A5    40 8F    88  08A6  1961          BISB    #ACB$M_QUOTA,ACB$B_RMOD(R5)   ; SET AST QUOTA ACCOUNTING FLAG
                 14 A5    14 A3    D0  08AB  1962          MOVL    IRP$L_ASTPRM(R3),ACB$L_ASTPRM(R5)  ; COPY AST PARAMETER
                                   08B0  1963
                                   08B0  1964 30$:      ; NOW POST EVENT FLAG IF SUBFUNCTION CODE SPECIFIES IT
                       52   01     9A  08B0  1965          MOVZBL  #PRI$_IOCOM,R2                  ; PRIORITY INCREMENT CLASS
                0A 20 A3    06     E1  08B3  1966          BBC     #IO$V_SETEVF,IRP$W_FUNC(R3),35$  ; BRANCH IF DON'T POST E.F.
                   53   22 A3     9A  08B8  1967          MOVZBL  IRP$B_EFN(R3),R3                ; EVENT FLAG NUMBER
                00000000'GF      16  08BC  1968          JSB     G^SCH$POSTEF                    ; POST EVENT FLAG
                                   08C2  1969
                                   08C2  1970 35$:      ; NOW EITHER GIVE AST OR DEALLOCATE BLOCK
                         10 A5     D5  08C2  1971          TSTL    ACB$L_AST(R5)                   ; GIVE AST?
                               06  13  08C5  1972          BEQL    40$                             ; NO
                00000000'GF      17  08C7  1973          JMP     G^SCH$QAST                      ; YES
                                   08CD  1974
                                   08CD  1975 40$:      ; DON'T GIVE AST SO DEALLOCATE PACKET
                   52   0C A5     3C  08CD  1976          MOVZWL  ACB$L_PID(R5),R2                ; BUT FIRST INCR. AST QUOTA
                00000000'GF      DD  08D1  1977          PUSHL   G^SCH$GL_PCBVEC                 ; PUSH ADDRESS OF PCB TABLE
                   52   9E42     D0  08D7  1978          MOVL    @(SP)+[R2],R2                   ; GET PCB ADDRESS
                         38 A2     B6  08DB  1979          INCW    PCB$W_ASTCNT(R2)                ; INCR. QUOTA
                      50   55     D0  08DE  1980          MOVL    R5,R0
                00000000'GF      17  08E1  1981          JMP     G^EXE$DEANONPAGED
```

```
                                        08E7  1983          .SBTTL  DODIAGERL - DO DIAG. AND ERROR LOGGING STUFF
                                        08E7  1984 ;++
                                        08E7  1985 ; FUNCTIONAL DESCRIPTION:
                                        08E7  1986 ;
                                        08E7  1987 ;       THIS ROUTINE DOES THE FOLLOWING:
                                        08E7  1988 ;               1)  CALLS THE DIAGNOSTIC BUFFER FILL ROUTINE WHICH COPIES
                                        08E7  1989 ;                   THE REGISTER SAVE INFO. INTO A DIAGNOSTIC BUFFER IF ONE
                                        08E7  1990 ;                   WAS SUPPLIED WITH THE REQUEST.
                                        08E7  1991 ;               2)  IF THE I/O STATUS INDICATES A LOGGABLE ERROR, THEN
                                        08E7  1992 ;                   THE ERROR IS LOGGED.  NOTE THAT THIS ROUTINE DOES THE
                                        08E7  1993 ;                   PROCESSING NORMALLY DONE IN IOC$REQCOM SINCE THIS DRIVER
                                        08E7  1994 ;                   DOESN'T CALL IOC$REQCOM.
                                        08E7  1995 ;
                                        08E7  1996 ; CALLING SEQUENCE:
                                        08E7  1997 ;
                                        08E7  1998 ;       BSBW    DODIAGERL
                                        08E7  1999 ;
                                        08E7  2000 ; INPUT PARAMETERS:
                                        08E7  2001 ;
                                        08E7  2002 ;       R0      FIRST LONGWORD OF I/O STATUS
                                        08E7  2003 ;       R1      SECOND LONGWORD OF I/O STATUS
                                        08E7  2004 ;       R3      ADDRESS OF IRP
                                        08E7  2005 ;       R5      ADDRESS OF UCB
                                        08E7  2006 ;
                                        08E7  2007 ; IMPLICIT INPUTS:
                                        08E7  2008 ;
                                        08E7  2009 ;       VARIOUS FIELDS IN THE IRP AND UCB
                                        08E7  2010 ;
                                        08E7  2011 ; OUTPUT PARAMETERS:
                                        08E7  2012 ;
                                        08E7  2013 ;       NONE
                                        08E7  2014 ;
                                        08E7  2015 ; SIDE EFFECTS:
                                        08E7  2016 ;
                                        08E7  2017 ;       OFFSET UCB$W_FUNC IN THE UCB IS MODIFIED
                                        08E7  2018 ;--
                                        08E7  2019
                                        08E7  2020 DODIAGERL:
                                  07    BB    08E7  2021          PUSHR   #^M<R0,R1,R2>
                                58 A5   DD    08E9  2022          PUSHL   UCB$L_IRP(R5)           ; SAVE THIS 'CAUSE WE MODIFY IT
                                        08EC  2023
                           009A C5  20 A3   B0    08EC  2024          MOVW    IRP$W_FUNC(R3),UCB$W_FUNC(R5)   ; SAVE FUNCTION CODE
                                58 A5   53   D0    08F2  2025          MOVL    R3,UCB$L_IRP(R5)        ; MAKE THIS IRP THE 'CURRENT' ONE
                                        08F6  2026
                                        08F6  2027          ; CALL DIAGNOSTIC BUFFER FILL ROUTINE
                            00000000'GF   16    08F6  2028          JSB     G^IOC$DIAGBUFILL
                                        08FC  2029
                                        08FC  2030          ; CALL ERROR LOGGER IF WE HAVE A LOGGABLE ERROR
                           022C 8F  38 A3   B1    08FC  2031          CMPW    IRP$L_IOST1(R3),#SS$_TIMEOUT    ; IS IT A TIMEOUT?
                                    08   12    0902  2032          BNEQ    10$                             ; NO
                            00000000'GF   16    0904  2033          JSB     G^ERL$DEVICTMO                  ; YES, LOG TIMEOUT
                                    1E   11    090A  2034          BRB     40$
                                        090C  2035
                                        090C  2036 10$:     ; IS IT ANY OTHER LOGGABLE ERROR?
                           0054 8F  38 A3   B1    090C  2037          CMPW    IRP$L_IOST1(R3),#SS$_CTRLERR    ; IS IT A FATAL HRDWRE ERROR?
                                    10   13    0912  2038          BEQL    30$                             ; YES
                           0334 8F  38 A3   B1    0914  2039          CMPW    IRP$L_IOST1(R3),#SS$_DEVREQERR  ; IS IT A DEVICE REQUEST ERROR?
```

```
                 08   13  091A  2040          BEQL    30$                             ; YES
01F4 8F    38 A3   B1  091C  2041          CMPW    IRP$L_IOST1(R3),#SS$_PARITY     ; UBA PARITY ERROR?
              25   12  0922  2042          BNEQ    50$                             ; NO
                       0924  2043
    00000000'GF    16  0924  2044 30$:     JSB     G^ERL$DEVICERR                  ; LOG DEVICE ERROR
                       092A  2045
                       092A  2046 40$:     ; NOW FILL IN REST OF BUFFER LIKE IOC$REQCOM DOES
  1A 64 A5    02   E5  092A  2047          BBCC    #UCB$V_ERLOGIP,UCB$W_STS(R5),50$ ; CLEAR ERROR LOG IN PROGRESS
    52  0094 C5   D0  092F  2048          MOVL    UCB$L_EMB(R5),R2                ; GET ADDRESS OF ERROR MESSAGE BUFFER
  1A A2   64 A5   B0  0934  2049          MOVW    UCB$W_STS(R5),EMB$W_DV_STS(R2) ;INSERT FINAL DEVICE STATUS
10 A2  0080 C5    B0  0939  2050          MOVW    UCB$B_ERTCNT(R5),EMB$B_DV_ERTCNT(R2)    ; INSERT ERROR COUNTERS
    12 A2   50   7D  093F  2051          MOVQ    R0,EMB$Q_DV_IOSB(R2)            ; INSERT I/O STATUS
                       0943  2052
    00000000'GF    16  0943  2053          JSB     G^ERL$RELEASEMB                 ; RELEASE ERROR MESSAGE BUFFER
                       0949  2054
          58 A5 8ED0   0949  2055 50$:     POPL    UCB$L_IRP(R5)                   ; RESTORE THIS LOCATION
              07   BA  094D  2056          POPR    #^M<R0,R1,R2>
                   05  094F  2057          RSB
```

```
                     0950  2059          .SBTTL  LA_REGDUMP - REGISTER DUMP ROUTINE
                     0950  2060 ;++
                     0950  2061 ; FUNCTIONAL DESCRIPTION
                     0950  2062 ;
                     0950  2063 ;       THIS ROUTINE WRITES THE SAVED REGISTERS INTO A BUFFER.  IT IS
                     0950  2064 ;       CALLED FROM THE ERROR LOGGING ROUTINE AND THE DIAGNOSTIC BUFFER
                     0950  2065 ;       FILL ROUTINE.
                     0950  2066 ;
                     0950  2067 ; CALLING SEQUENCE:
                     0950  2068 ;
                     0950  2069 ;       BSBW    LA_REGDUMP
                     0950  2070 ;
                     0950  2071 ; INPUT PARAMETERS:
                     0950  2072 ;
                     0950  2073 ;       R0      ADDRESS OF REGISTER SAVE BUFFER
                     0950  2074 ;       R5      ADDRESS OF UCB
                     0950  2075 ;
                     0950  2076 ; OUTPUT PARAMETERS:
                     0950  2077 ;
                     0950  2078 ;       NONE
                     0950  2079 ;
                     0950  2080 ; SIDE EFFECTS:
                     0950  2081 ;
                     0950  2082 ;       R1,R2 ARE NOT PRESERVED
                     0950  2083 ;--
                     0950  2084
                     0950  2085 LA_REGDUMP:
     80    06   D0  0950  2086          MOVL    #6,(R0)+                        ; INSERT NUMBER OF REGISTERS INTO BFR
51   00E4  C5   DE  0953  2087          MOVAL   UCB$L_REGSAVE(R5),R1            ; GET ADDRESS OF SAVE AREA
     52    04   D0  0958  2088          MOVL    #4,R2                           ; NUMBER OF LPA-11 REGISTERS
     80    81   3C  095B  2089 10$:     MOVZWL  (R1)+,(R0)+                     ; COPY INTO BUFFER
       FA  52   F5  095E  2090          SOBGTR  R2,10$                          ; LOOP BACK
                     0961  2091
     60    61   7D  0961  2092          MOVQ    (R1),(R0)                       ; COPY DATAPATH NUMBER AND REGISTER
                  05  0964  2093          RSB
```

```
                     0965  2096          .SBTTL  CANCEL_IO - CANCEL I/O
                     0965  2097
                     0965  2098 ;++
                     0965  2099 ; FUNCTIONAL DESCRIPTION:
                     0965  2100 ;
                     0965  2101 ;       THIS ROUTINE PERFORMS THE CANCEL I/O FUNCTION.  ONLY PACKETS
                     0965  2102 ;       THAT HAVE A MATCHING CHANNEL INDEX AND PID ARE CANCELED.  FIRST, THE
                     0965  2103 ;       CURRENT PACKET (IF THERE IS ONE) IS CANCELED BY SETTING THE CANCEL I/O
                     0965  2104 ;       BIT IN THE UCB.  THEN ANY PACKETS ON THE INPUT QUEUE ARE CANCELED
                     0965  2105 ;       BY SENDING THEM TO REQ_COMPLETE WITH A STATUS OF SS$_CANCEL.  THE
                     0965  2106 ;       ONLY EXCEPTION IS THAT STOP QIO'S ARE NOT CANCELED.  FINALLY,
                     0965  2107 ;       ONGOING DATA TRANSFERS ARE CANCELED BY SENDING THEM TO QUEUE_STOP_REQ
                     0965  2108 ;       WITH A STATUS OF SS$_ABORT.
                     0965  2109 ;
                     0965  2110 ; CALLING SEQUENCE:
                     0965  2111 ;
                     0965  2112 ;       BSBW/B
                     0965  2113 ;
                     0965  2114 ; INPUT PARAMETERS:
                     0965  2115 ;
                     0965  2116 ;       R2      CHANNEL INDEX
                     0965  2117 ;       R3      POINTER TO CURRENT I/O PACKET
                     0965  2118 ;       R4      PCB ADDRESS
                     0965  2119 ;       R5      POINTER TO UCB
                     0965  2120 ;
                     0965  2121 ; OUTPUT PARAMETERS:
                     0965  2122 ;
                     0965  2123 ;       NONE
                     0965  2124 ;--
                     0965  2125
                     0965  2126 CANCEL_IO:
     00DC 8F    BB   0965  2127          PUSHR   #^M<R2,R3,R4,R6,R7>
     57   52    D0   0969  2128          MOVL    R2,R7                   ; CHANNEL INDEX
54   60   A4    D0   096C  2129          MOVL    PCB$L_PID(R4),R4        ; PUT PID IN R4
                     0970  2130
                     0970  2131          ; FIRST CANCEL CURRENT I/O PACKET IF THERE IS ONE
          53    D5   0970  2132          TSTL    R3                      ; POINTER TO CURRENT PACKET
          08    13   0972  2133          BEQL    10$                     ; NO CURRENT PACKET
          54    10   0974  2134          BSBB    CANCELCK                ; CHECK CHANNEL AND PID
          04    12   0976  2135          BNEQ    10$                     ; NOT A MATCH
64 A5     08    A8   0978  2136          BISW    #UCB$M_CANCEL,UCB$W_STS(R5)  ; SET CANCEL BIT
                     097C  2137
                     097C  2138 10$:     ; NOW CANCEL THE PACKETS ON THE INPUT QUEUE
50   0830 8F    3C   097C  2139          MOVZWL  #SS$_CANCEL,R0          ; STATUS
          51    D4   0981  2140          CLRL    R1
53   00AC C5    9E   0983  2141          MOVAB   UCB$L_INQFL(R5),R3      ; GET POINTER TO QUEUE HEAD
     56   53    D0   0988  2142          MOVL    R3,R6                   ; SAVE POINTER TO QUEUE HEAD
                     098B  2143
                     098B  2144 20$:     ; EXAMINE NEXT PACKET IN QUEUE
     53   63    D0   098B  2145          MOVL    IRP$L_IOQFL(R3),R3      ; GET POINTER TO NEXT PACKET
     56   53    D1   098E  2146          CMPL    R3,R6                   ; REACHED END OF QUEUE YET?
          1A    13   0991  2147          BEQL    30$                     ; YES, DONE WITH THIS PHASE
          35    10   0993  2148          BSBB    CANCELCK                ; CHECK CHANNEL AND PID
          F4    12   0995  2149          BNEQ    20$                     ; NOT A MATCH, GET NEXT PACKET
20 A3     03    91   0997  2150          CMPB    #IO$_STOP,IRP$W_FUNC(R3)  ; DON'T CANCEL STOP REQUESTS
          EE    13   099B  2151          BEQL    20$                     ; IT'S A STOP.  GET NEXT PACKET
52   04   A3    D0   099D  2152          MOVL    IRP$L_IOQBL(R3),R2      ; HAVE A PACKET TO REMOVE.  BACK UP
```

```
                53   00 B2    0F   09A1  2153          REMQUE  @IRP$L_IOQFL(R2),R3     ; REMOVE PACKET FROM QUEUE
                     FB62     30   09A5  2154          BSBW    REQ_COMPLETE            ; SEND PACKET TO REQUEST COMPLETE
                   53   52     D0   09A8  2155          MOVL    R2,R3
                        DE     11   09AB  2156          BRB     20$                     ; GET NEXT PACKET
                                    09AD  2157
                                    09AD  2158 30$:     ; NOW STOP ANY MATCHING DATA TRANSFER REQUESTS
                   50   2C     3C   09AD  2159          MOVZWL  #SS$_ABORT,R0           ; STATUS
                        52     D4   09B0  2160          CLRL    R2
                                    09B2  2161
                                    09B2  2162 40$:     ; EXAMINE NEXT ENTRY IN REQUEST LIST
             53   0104 C542     D0   09B2  2163          MOVL    UCB$L_RQLIST(R5)[R2],R3 ; GET POINTER TO PACKET
                        07     13   09B8  2164          BEQL    50$                     ; EMPTY SLOT
                        0E     10   09BA  2165          BSBB    CANCELCK                ; CHECK CHANNEL AND PID
                        03     12   09BC  2166          BNEQ    50$                     ; NOT A MATCH
                     FE4F     30   09BE  2167          BSBW    QUEUE_STOP_REQ          ; QUEUE A STOP REQUEST
                ED 52   08     F2   09C1  2168 50$:     AOBLSS  #8,R2,40$               ; REPEAT FOR ALL 8 REQUESTS
                                    09C5  2169
                   00DC 8F     BA   09C5  2170          POPR    #^M<R2,R3,R4,R6,R7>
                               05   09C9  2171          RSB
                                    09CA  2172
                                    09CA  2173
                                    09CA  2174
                                    09CA  2175
                                    09CA  2176 ; LOCAL SUBROUTINE TO CHECK FOR MATCHING CHANNEL AND PID
                                    09CA  2177 ;
                                    09CA  2178 ; INPUT:
                                    09CA  2179 ;       R3      POINTS TO I/O PACKET
                                    09CA  2180 ;       R4      CONTAINS PID
                                    09CA  2181 ;       R7      CONTAINS CHANNEL INDEX
                                    09CA  2182 ; OUTPUT:
                                    09CA  2183 ;       Z BIT   IS SET IF BOTH MATCH, CLEARED OTHERWISE
                                    09CA  2184
                                    09CA  2185 CANCELCK:
          54   0C A3     D1   09CA  2186          CMPL    IRP$L_PID(R3),R4        ; CHECK PID
                   04     12   09CE  2187          BNEQ    10$                     ; NO MATCH
          57   28 A3     B1   09D0  2188          CMPW    IRP$W_CHAN(R3),R7       ; CHECK CHANNEL AND SET OR CLEAR Z BIT
                               09D4  2189 10$:     RSB
```

```
                        09D5  2191          .SBTTL  COMPLETE_ALL - COMPLETE ALL DATA TRANSFER REQUESTS
                        09D5  2192
                        09D5  2193 ;++
                        09D5  2194 ; FUNCTIONAL DESCRIPTION:
                        09D5  2195 ;
                        09D5  2196 ;       THIS ROUTINE GOES THROUGH THE USER TABLE SENDING ALL CURRENT
                        09D5  2197 ;       DATA TRANSFER REQUESTS TO REQ_COMPLETE.
                        09D5  2198 ;
                        09D5  2199 ; CALLING SEQUENCE:
                        09D5  2200 ;
                        09D5  2201 ;       BSBW    COMPLETE_ALL
                        09D5  2202 ;
                        09D5  2203 ; INPUT PARAMETERS:
                        09D5  2204 ;
                        09D5  2205 ;       R0      FIRST LONGWORD OF I/O STATUS BLOCK
                        09D5  2206 ;       R1      SECOND LONGWORD OF I/O STATUS BLOCK
                        09D5  2207 ;       R5      ADDRESS OF UCB
                        09D5  2208 ;
                        09D5  2209 ; OUTPUT PARAMETERS:
                        09D5  2210 ;
                        09D5  2211 ;       NONE
                        09D5  2212 ;
                        09D5  2213 ; SIDE EFFECTS:
                        09D5  2214 ;
                        09D5  2215 ;       R2,R3 ARE NOT SAVED
                        09D5  2216 ;--
                        09D5  2217
                        09D5  2218 COMPLETE_ALL:
                        09D5  2219
                52   D4 09D5  2220          CLRL    R2                      ; INITIALIZE INDEX INTO REQUEST LIST
                        09D7  2221
                        09D7  2222 20$:     ; DO NEXT ONE IN REQUEST LIST
            53  0104 C542 D0  09D7  2223          MOVL    UCB$L_RQLIST(R5)[R2],R3 ; GET POINTER TO I/O PACKET
                     08 13  09DD  2224          BEQL    30$                     ; NO REQUEST IN THIS SLOT
                0104 C542 D4  09DF  2225          CLRL    UCB$L_RQLIST(R5)[R2]    ; CLEAR SLOT
                     FB23 30  09E4  2226          BSBW    REQ_COMPLETE            ; SEND IT TO REQUEST COMPLETE
                EC 52   08 F2  09E7  2227 30$:    AOBLSS  #8,R2,20$               ; GO BACK FOR NEXT
                        05  09EB  2228          RSB
```

```
                     09EC  2230          .SBTTL  UNIT_INIT - LPA-11 UNIT INITIALIZATION
                     09EC  2231 ;++
                     09EC  2232 ; FUNCTIONAL DESCRIPTION:
                     09EC  2233 ;
                     09EC  2234 ;       THIS ROUTINE IS ENTERED WHEN THE DRIVER IS LOADED AND ON POWER
                     09EC  2235 ;       RECOVERY.  ON DRIVER LOAD IT INITIALIZES THE UCB, OPTIONALLY
                     09EC  2236 ;       PREALLOCATES MAP REGISTERS, AND ALLOCATES AND LOADS MAP REGISTERS
                     09EC  2237 ;       TO PERMANENTLY MAP THE RDA IN THE UCB.  ON POWER RECOVERY, IT
                     09EC  2238 ;       CLEARS THE MICROCODE VALID BIT, RELOADS THE MAP REGISTERS THAT
                     09EC  2239 ;       MAP THE RDA IN THE UCB, AND THEN FORKS TO COMPLETE ALL ACTIVE
                     09EC  2240 ;       REQUESTS WITH A STATUS OF SS$_POWERFAIL.
                     09EC  2241 ;
                     09EC  2242 ; CALLING SEQUENCE:
                     09EC  2243 ;
                     09EC  2244 ;       JSB     UNIT_INIT
                     09EC  2245 ;
                     09EC  2246 ; INPUT PARAMETERS:
                     09EC  2247 ;
                     09EC  2248 ;       R5      ADDRESS OF UCB
                     09EC  2249 ;
                     09EC  2250 ; OUTPUT PARAMETERS:
                     09EC  2251 ;
                     09EC  2252 ;       NONE
                     09EC  2253 ;
                     09EC  2254 ; SIDE EFFECTS:
                     09EC  2255 ;
                     09EC  2256 ;       R0 - R4 ARE NOT PRESERVED
                     09EC  2257 ;--
                     09EC  2258
                     09EC  2259 UNIT_INIT:
           51  24 A5  D0  09EC  2260          MOVL    UCB$L_CRB(R5),R1        ; GET POINTER TO CRB
                     09F0  2261
                     09F0  2262          ; DETERMINE IF THIS IS INITIAL LOADING OR POWER RECOVERY
        67 64 A5  05  E0  09F0  2263          BBS     #UCB$V_POWER,UCB$W_STS(R5),60$  ; BRANCH IF POWER RECOVERY
                     09F5  2264 ;
                     09F5  2265 ;       D R I V E R   L O A D
                     09F5  2266 ;
                     09F5  2267          ; INITIALIZE INPUT QUEUE
00AC C5  00AC C5  DE 09F5  2268          MOVAL   UCB$L_INQFL(R5),UCB$L_INQFL(R5)
00B0 C5  00AC C5  DE 09FC  2269          MOVAL   UCB$L_INQFL(R5),UCB$L_INQBL(R5)
                     0A03  2270
                     0A03  2271          ; MAKE UCB OWNER OF IDB
           50  2C A1  D0  0A03  2272          MOVL    CRB$L_INTD+VEC$L_IDB(R1),R0  ; GET POINTER TO IDB
           04  A0  55  D0  0A07  2273          MOVL    R5,IDB$L_OWNER(R0)      ; MAKE UCB OWNER OF IDB
                     0A0B  2274
                     0A0B  2275          ; OPTIONALLY PREALLOCATE MAP REGISTERS
     53  00000000'GF  9A  0A0B  2276          MOVZBL  G^IOC$GW_LAMAPREG,R3    ; NUM. TO PREALLOCATE (SYSGEN PARAM.)
                 30  13  0A12  2277          BEQL    20$                     ; DON'T PREALLOCATE
         00FE 8F  53  B1  0A14  2278          CMPW    R3,#254                 ; Prevent allocating more than 254.
                 05  15  0A19  2279          BLEQ    10$                     ; LEQ implies we are OK.
         53  00FE 8F  3C  0A1B  2280          MOVZWL  #254,R3                 ; Else reduce request to 254 registers.
                     0A20  2281 10$:
          00000000'GF  16  0A20  2282          JSB     G^IOC$ALOUBMAPRMN       ; Permanently allocate specified number.
               32  50  E9  0A26  2283          BLBC    R0,50$                  ; ERROR - DIDN'T ALLOCATE
           51  24  A5  D0  0A29  2284          MOVL    UCB$L_CRB(R5),R1        ; Refresh R1 => CRB.
              8000 8F  AA  0A2D  2285          BICW    #VEC$M_MAPLOCK,-        ; Undo permanent bit set by IOC$ALOUBMAPRMN.
                 34  A1      0A31  2286                  CRB$L_INTD+VEC$W_MAPREG(R1)
```

```
                34 A1   D0  0A33  2287          MOVL    CRB$L_INTD+VEC$W_MAPREG(R1),-   ; SAVE INFO. ON MAP REGISTERS
            00A8 C5         0A36  2288                  UCB$L_PREALLOC(R5)              ; ALLOCATED
                            0A39  2289
                            0A39  2290          ; NOW MARK IN UCB BITMAP AS AVAILABLE, THE MAP REGISTERS ALLOCATED
             50   00   D2  0A39  2291          MCOML   #0,R0                           ; BITMAP PATTERN (1 MEANS AVAILABLE)
      54     00A8 C5   3C  0A3C  2292          MOVZWL  UCB$L_PREALLOC(R5),R4           ; R4 contains starting map register #
              FC01     30  0A41  2293          BSBW    ALT_LOCALBITMAP                 ; ALTER MAP
                            0A44  2294
                            0A44  2295  20$:    ; ALLOCATE AND LOAD MAP REGISTERS TO PERMANENTLY MAP RDA IN UCB
                  51    10  0A44  2296          BSBB    LOADUCB                         ; LOAD BOFF, BCNT, AND SVAPTE IN UCB
                FB69    30  0A46  2297          BSBW    SETMAPREG                       ; REQUEST AND LOAD UBA MAP REGISTERS
               OF 50    E9  0A49  2298          BLBC    R0,50$                          ; ALLOCATION FAILURE
                34 A1   D0  0A4C  2299          MOVL    CRB$L_INTD+VEC$W_MAPREG(R1),-   ; SAVE ALLOCATED MAP REGISTER
            00A4 C5         0A4F  2300                  UCB$L_RDAMR(R5)                 ; INFO. IN UCB
      00A0  C5    52    D0  0A52  2301          MOVL    R2,UCB$L_RDABA(R5)              ; UNIBUS ADDRESS OF RDA
        64  A5    10    A8  0A57  2302          BISW    #UCB$M_ONLINE,UCB$W_STS(R5)     ; SET UNIT ONLINE
                    05    0A5B  2303  50$:    RSB
                            0A5C  2304
                            0A5C  2305
                            0A5C  2306  ;
                            0A5C  2307  ;       P O W E R   R E C O V E R Y
                            0A5C  2308  ;
        44  A5    01    CA  0A5C  2309  60$:    BICL    #LA$M_MCVALID,UCB$L_DEVDEPEND(R5)  ; CLEAR MICROCODE VALID
        64  A5    10    A8  0A60  2310          BISW    #UCB$M_ONLINE,UCB$W_STS(R5)     ; SET UNIT ONLINE
                            0A64  2311
                            0A64  2312          ; RELOAD UBA MAP REGISTERS TO MAP RDA IN UCB
                  31    10  0A64  2313          BSBB    LOADUCB                         ; LOAD BCNT, BOFF, AND SVAPTE IN UCB
            00A4 C5         0A66  2314          MOVL    UCB$L_RDAMR(R5),-               ; LOAD MAPREG, NUMREG, AND DATAPATH
                34 A1       0A6A  2315                  CRB$L_INTD+VEC$W_MAPREG(R1)     ; IN CRB
      00000000'GF   16      0A6C  2316          JSB     G^IOC$LOADUBAMAP                ; LOAD MAP REGISTERS
                            0A72  2317
                            0A72  2318          ; FORK TO COMPLETE ALL ACTIVE REQUESTS
                00CC C5   D5  0A72  2319          TSTL    UCB$L_FORKP(R5)             ; INTERLOCK AGAINST MULTIPLE PWR FAILS
                      1E    12  0A76  2320          BNEQ    90$                         ; IT'S ALREADY QUEUED!
             55   00CC C5   DE  0A78  2321          MOVAL   UCB$L_FORKP(R5),R5          ; POINT TO FORK BLOCK
                            0A7D  2322          FORK
             55   FF34 C5   DE  0A83  2323          MOVAL   -UCB$L_FORKP(R5),R5         ; RESTORE POINTER TO UCB
                  00CC C5   D4  0A88  2324          CLRL    UCB$L_FORKP(R5)             ; INDICATE THAT FORK BLOCK IS AVAILABLE
             50   0364 8F   3C  0A8C  2325          MOVZWL  #SS$_POWERFAIL,R0           ; RETURN STATUS
                      51    D4  0A91  2326          CLRL    R1
                  FF3F      30  0A93  2327          BSBW    COMPLETE_ALL                ; COMPLETE ALL REQUESTS
                            05  0A96  2328  90$:    RSB
                            0A97  2329
                            0A97  2330
                            0A97  2331
                            0A97  2332          ; LOCAL SUBROUTINE TO LOAD BCNT, BOFF, AND SVAPTE FIELDS IN
                            0A97  2333          ; UCB WITH VALUES WHICH DESCRIBE UCB$W_RDA
                            0A97  2334
                            0A97  2335  LOADUCB:
          7E A5     3A    B0  0A97  2336          MOVW    #58,UCB$W_BCNT(R5)          ; SIZE OF RDA
        50   0164 C5     3E  0A9B  2337          MOVAW   UCB$W_RDA(R5),R0            ; GET ADDRESS OF RDA
                            0AA0  2338
                            0AA0  2339          ASSUME  VA$S_BYTE EQ 9
  7C A5   50    FE00 8F   AB  0AA0  2340          BICW3   #^XFE00,R0,UCB$W_BOFF(R5)   ;  INSERT BYTE OFFSET IN PAGE
    50    50      15    09  EF  0AA7  2341          EXTZV   #VA$V_VPN,#VA$S_VPN,R0,R0   ; GET VIRTUAL PAGE #
      52   00000000'GF   D0  0AAC  2342          MOVL    G^MMG$GL_SPTBASE,R2         ; GET ADDRESS OF SYSTEM PAGE TABLE
          78 A5   6240    DE  0AB3  2343          MOVAL   (R2)[R0],UCB$L_SVAPTE(R5)   ; STORE SVA OF PTE FOR RDA
```

```
05  0AB8  2344          RSB
```

```
                0AB9  2346
                0AB9  2347
                0AB9  2348 LA_END:                              ; ADDRESS OF LAST LOCATION IN DRIVER
                0AB9  2349
                0AB9  2350
                0AB9  2351
                0AB9  2352
                0AB9  2353          .END
```

```
$$$                     =  00000020  R      02
$$OP                    =  00000002
ABORT                      000002CE  R      03
ACB$B_RMOD              =  0000000B
ACB$L_AST               =  00000010
ACB$L_ASTPRM            =  00000014
ACB$L_PID               =  0000000C
ACB$M_QUOTA             =  00000040
ALIGNERR                   000002C0  R      03
ALLOC_LOCALMR              000005E7  R      03
ALT_LOCALBITMAP            00000645  R      03
AT$_UBA                 =  00000001
BFRFULL                    000007F5  R      03
CANCELCK                   000009CA  R      03
CANCEL_IO                  00000965  R      03
CLEANUP                    000002E5  R      03
CMDERR                     000007B1  R      03
COM$POST                   ********    X    03
COMMON                     000003DE  R      03
COMPLETE_ALL               000009D5  R      03
COMPL_ALL_REQS             0000076D  R      03
CRB$L_INTD              =  00000024
CRB$L_INTD2             =  00000048
DC$_REALTIME            =  00000060
DDB$L_DDT               =  0000000C
DEV$M_AVL               =  00040000
DEV$M_ELG               =  00400000
DEV$M_IDV               =  04000000
DEV$M_ODV               =  08000000
DEV$M_RTM               =  20000000
DEV$M_SHR               =  00010000
DEVADDR                 =  00000002
DODIAGERL                  000008E7  R      03
DONE                       0000042C  R      03
DPT$C_LENGTH            =  00000038
DPT$C_VERSION           =  00000004
DPT$INITAB                 00000038  R      02
DPT$M_NOUNLOAD          =  00000004
DPT$REINITAB               0000005D  R      02
DPT$TAB                    00000000  R      02
DT$_LPA11               =  00000001
DYN$C_ACB               =  00000002
DYN$C_CRB               =  00000005
DYN$C_DDB               =  00000006
DYN$C_DPT               =  0000001E
DYN$C_FRK               =  00000008
DYN$C_UCB               =  00000010
EMB$B_DV_ERTCNT         =  00000010
EMB$L_DV_REGSAV         =  0000004E
EMB$Q_DV_IOSB           =  00000012
EMB$W_DV_STS            =  0000001A
ERL$DEVICERR               ********    X    03
ERL$DEVICTMO               ********    X    03
ERL$RELEASEMB              ********    X    03
ERROR                      00000748  R      03
EXE$ALONONPAGED            ********    X    03
EXE$DEANONPAGED            ********    X    03
EXE$FINISHIOC              ********    X    03
EXE$FORK                   ********    X    03
EXE$GL_TENUSEC             ********    X    03
EXE$GL_UBDELAY             ********    X    03
EXE$INSERTIRP              ********    X    03
EXE$IOFORK                 ********    X    03
EXE$QIORETURN              ********    X    03
EXE$READLOCKR              ********    X    03
EXE$WRITECHK               ********    X    03
EXE$WRITELOCK              ********    X    03
EXE$WRITELOCKR             ********    X    03
FKB$B_FIPL              =  0000000B
FKB$K_LENGTH            =  00000018
FUNCTABLE                  00000038  R      03
FUNCTAB_LEN             =  00000058
IDB$L_CSR               =  00000000
IDB$L_OWNER             =  00000004
INITIALIZE                 000003CB  R      03
INIT_FDT                   0000016E  R      03
INTEXIT                    000006DE  R      03
IO$V_SETEVF             =  00000006
IO$_INITIALIZE          =  00000004
IO$_LOADMCODE           =  00000001
IO$_QSTOP               =  00000007
IO$_SETCLOCK            =  00000037
IO$_SETCLOCKP           =  00000005
IO$_STARTDATA           =  00000038
IO$_STARTDATAP          =  00000006
IO$_STARTMPROC          =  00000002
IO$_STOP                =  00000003
IO$_VIRTUAL             =  0000003F
IOC$ALOUBAMAP              ********    X    03
IOC$ALOUBMAPRMN            ********    X    03
IOC$DIAGBUFILL             ********    X    03
IOC$GW_LAMAPREG            ********    X    03
IOC$INITIATE               ********    X    03
IOC$LOADUBAMAP             ********    X    03
IOC$MNTVER                 ********    X    03
IOC$PURGDATAP              ********    X    03
IOC$RELDATAP               ********    X    03
IOC$RELMAPREG              ********    X    03
IOC$REQDATAPNW             ********    X    03
IOC$RETURN                 ********    X    03
IOC$WFIKPCH                ********    X    03
IOFCTBL                    00000090  R      03
IOFCTBLN                =  00000007
IPL$_QUEUEAST           =  00000006
IRP$B_CARCON            =  0000003C
IRP$B_EFN               =  00000022
IRP$B_RMOD              =  0000000B
IRP$B_TYPE              =  0000000A
IRP$C_LENGTH            =  000000C4
IRP$L_ASTPRM            =  00000014
IRP$L_BFR_AST           =  0000003C
IRP$L_IOQBL             =  00000004
IRP$L_IOQFL             =  00000000
IRP$L_IOST1             =  00000038
```

```
IRP$L_IOST2                     = 0000003C
IRP$L_MEDIA                     = 00000038
IRP$L_OVR_AST                   = 00000040
IRP$L_PID                       = 0000000C
IRP$L_RDAMAPREG                 = 00000040
IRP$L_SEGVBN                    = 00000048
IRP$L_SIP                       = 00000048
IRP$L_SVAPTE                    = 0000002C
IRP$S_FCODE                     = 00000006
IRP$W_ABCNT                     = 00000040
IRP$W_CHAN                      = 00000028
IRP$W_FUNC                      = 00000020
IRP$W_SIZE                      = 00000008
LA$DDT                            00000000 RG     03
LA$M_MCVALID                    = 00000001
LA$RDYININTSV                     000006B5 RG     03
LA$RDYOUTINTSV                    000006E8 RG     03
LA$S_CONFIG                     = 0000000A
LA$S_MCTYPE                     = 00000002
LA$S_RATE                       = 00000003
LA$V_CONFIG                     = 00000003
LA$V_MCTYPE                     = 00000001
LA$V_PRESET                     = 00000010
LA$V_RATE                       = 0000000D
LA_CISR                           00000000
LA_CISR_M_CRAM                  = 00002000
LA_CISR_M_ENA                   = 00000800
LA_CISR_M_IE                    = 00000040
LA_CISR_M_RESET                 = 00004000
LA_CISR_M_ROMO                  = 00000400
LA_CISR_M_RUN                   = 00008000
LA_COSR                           00000002
LA_COSR_M_IE                    = 00000040
LA_COSR_M_RDY                   = 00000080
LA_COSR_S_ERRTP                 = 00000002
LA_COSR_V_ERRTP                 = 0000000D
LA_END                            00000AB9 R      03
LA_MAINT                          00000006
LA_RDA                            00000004
LA_REGDUMP                        00000950 R      03
LENGTHERR                         000002C7 R      03
LOAD                              00000400 R      03
LOADUCB                           00000A97 R      03
LOAD_MICROCODE                    00000097 R      03
MASKA                           = 00000000
MASKL                           = 00000080
MCNVALID                          000003F6 R      03
MMG$GL_SPTBASE                    ******** X      03
MMG$UNLOCK                        ******** X      03
NO_ERROR                          000007C1 R      03
P1                              = 00000000
P2                              = 00000004
P3                              = 00000008
P4                              = 0000000C
PCB$L_PID                       = 00000060
PCB$W_ASTCNT                    = 00000038
PR$_IPL                         = 00000012
PRI$_IOCOM                                      = 00000001
QSTOP_FDT                                         000002F5 R      03
QUEUE_STOP_REQ                                    00000810 R      03
QUE_PRT                                           0000031A R      03
RDA_IN_UCB                                        000003C4 R      03
READLOCK                                          000002D4 R      03
REL_MRDP                                          00000665 R      03
REQERR                                            0000078E R      03
REQ_COMPLETE                                      0000050A R      03
RESET                                             00000130 R      03
SCH$GL_PCBVEC                                     ******** X      03
SCH$POSTEF                                        ******** X      03
SCH$QAST                                          ******** X      03
SDATA                                             0000047A R      03
SETCHAR                                           00000445 R      03
SETCLOCK_FDT                                      000001BE R      03
SETMAPREG                                         000005B2 R      03
SET_CLOCK                                         000003AE R      03
SIGNAL_BFR_FULL                                   0000083F R      03
SIP$B_BFR_DATAP                                   00000033
SIP$B_BFR_NUMRE                                   00000032
SIP$B_RCL_DATAP                                   0000003F
SIP$B_RCL_NUMRE                                   0000003E
SIP$B_TYPE                                        0000000A
SIP$B_USW_DATAP                                   00000027
SIP$B_USW_NUMRE                                   00000026
SIP$B_VBFRMASK                                    00000007
SIP$L_BFR_SVAPT                                   00000028
SIP$L_RCL_SVAPT                                   00000034
SIP$L_SLVDATA                                     0000000C
SIP$L_USW_SVAPT                                   0000001C
SIP$W_BCNT                                        00000002
SIP$W_BFR_BCNT                                    0000002E
SIP$W_BFR_BOFF                                    0000002C
SIP$W_BFR_MAPRE                                   00000030
SIP$W_MODE                                        00000000
SIP$W_RCL_BCNT                                    0000003A
SIP$W_RCL_BOFF                                    00000038
SIP$W_RCL_MAPRE                                   0000003C
SIP$W_SIZE                                        00000008
SIP$W_USW_BCNT                                    00000022
SIP$W_USW_BOFF                                    00000020
SIP$W_USW_MAPRE                                   00000024
SIZ...                                          = 00000001
SS$_ABORT                                       = 0000002C
SS$_BADPARAM                                    = 00000014
SS$_BUFNOTALIGN                                 = 00000324
SS$_CANCEL                                      = 00000830
SS$_CTRLERR                                     = 00000054
SS$_DATACHECK                                   = 0000005C
SS$_DEVACTIVE                                   = 000002C4
SS$_DEVCMDERR                                   = 0000032C
SS$_DEVREQERR                                   = 00000334
SS$_EXQUOTA                                     = 0000001C
SS$_INSFBUFDP                                   = 0000033C
SS$_INSFMAPREG                                  = 00000344
SS$_INSFMEM                                     = 00000124
```

```
SS$_IVBUFLEN                    = 0000034C
SS$_IVMODE                      = 00000354
SS$_MCNOTVALID                  = 0000035C
SS$_NORMAL                      = 00000001
SS$_PARITY                      = 000001F4
SS$_POWERFAIL                   = 00000364
SS$_TIMEOUT                     = 0000022C
STARTDATA_FDT                     000001D6 R      03
STARTIO                           00000342 R      03
STARTMP_FDT                       00000165 R      03
STARTREQ                          000007C3 R      03
START_DATA                        000003B6 R      03
STOP                              000003BE R      03
STOP_MODE                       = 00000003
STRT_NXT_REQ                      00000431 R      03
TIMEOUT                           00000758 R      03
UCB$B_DEVCLASS                  = 00000040
UCB$B_DEVTYPE                   = 00000041
UCB$B_DIPL                      = 0000005E
UCB$B_ERTCNT                    = 00000080
UCB$B_FIPL                      = 0000000B
UCB$K_SIZE                      = 000001A0
UCB$L_CRB                       = 00000024
UCB$L_DEVCHAR                   = 00000038
UCB$L_DEVDEPEND                 = 00000044
UCB$L_DPC                       = 0000009C
UCB$L_DUETIM                    = 0000006C
UCB$L_EMB                       = 00000094
UCB$L_FORKO                       000000B4
UCB$L_FORKP                       000000CC
UCB$L_FPC                       = 0000000C
UCB$L_FR3                       = 00000010
UCB$L_INQBL                       000000B0
UCB$L_INQFL                       000000AC
UCB$L_IRP                       = 00000058
UCB$L_PREALLOC                    000000A8
UCB$L_RDABA                       000000A0
UCB$L_RDAMR                       000000A4
UCB$L_REGSAVE                     000000E4
UCB$L_RQLIST                      00000104
UCB$L_SVAPTE                    = 00000078
UCB$M_BSY                       = 00000100
UCB$M_CANCEL                    = 00000008
UCB$M_ONLINE                    = 00000010
UCB$M_POWER                     = 00000020
UCB$V_BSY                       = 00000008
UCB$V_CANCEL                    = 00000003
UCB$V_ERLOGIP                   = 00000002
UCB$V_INT                       = 00000001
UCB$V_POWER                     = 00000005
UCB$W_BCNT                      = 0000007E
UCB$W_BOFF                      = 0000007C
UCB$W_FUNC                      = 0000009A
UCB$W_MRBITMAP                    00000124
UCB$W_RDA                         00000164
UCB$W_RISAVE                      000000F4
UCB$W_ROSAVE                      000000FC
UCB$W_STS                       = 00000064
UNIT_INIT                         000009EC R      03
UNLOCK                            0000057C R      03
UNLOCKF                           00000572 R      03
VA$S_BYTE                       = 00000009
VA$S_VPN                        = 00000015
VA$V_VPN                        = 00000009
VEC$B_DATAPATH                  = 00000013
VEC$B_NUMREG                    = 00000012
VEC$L_IDB                       = 00000008
VEC$L_UNITINIT                  = 00000018
VEC$M_LWAE                      = 00000020
VEC$M_MAPLOCK                   = 00008000
VEC$S_DATAPATH                  = 00000005
VEC$V_DATAPATH                  = 00000000
VEC$W_MAPREG                    = 00000010
WAIT                              00000407 R      03
WRITELOCK                         000002DC R      03
```

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	000001A0 (416.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
$$$105_PROLOGUE	00000072 (114.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
$$$115_DRIVER	00000AB9 (2745.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	30	00:00:00.07	00:00:01.07
Command processing	108	00:00:00.40	00:00:03.44
Pass 1	635	00:00:19.53	00:01:10.65
Symbol table sort	0	00:00:02.70	00:00:11.54
Pass 2	388	00:00:04.96	00:00:16.90
Symbol table output	17	00:00:00.19	00:00:01.19
Psect synopsis output	0	00:00:00.00	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1180	00:00:27.86	00:01:44.81

The working set limit was 2250 pages.
166852 bytes (326 pages) of virtual memory were used to buffer the intermediate code.
There were 130 pages of symbol table space allocated to hold 2487 non-local and 98 local symbols.
2353 source lines were read in Pass 1, producing 23 object records in Pass 2.
51 pages of virtual memory were used to define 48 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYS.OBJ]LIB.MLB;1	34
_255DUA28:[SYSLIB]STARLET.MLB;2	11
TOTALS (all libraries)	45

2717 GETS were required to define 45 macros.

There were no errors, warnings or information messages.

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

DYDRIVER
LIS

OUTUEND
LIS

DUTUSUBS
LIS

DXPDRIVER
LIS

DXUTILITY
LIS

LADRIVER
LIS

LCDRIVER
LIS

LPDRIVER
LIS

NODRIVER
LIS

MBXDRIVER
LIS