```
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
```

```
DDDDDDDD    YY      YY  DDDDDDDD    RRRRRRRR     IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
DDDDDDDD    YY      YY  DDDDDDDD    RRRRRRRR     IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
DD      DD  YY      YY  DD      DD  RR      RR     II      VV      VV  EE          RR      RR
DD      DD  YY      YY  DD      DD  RR      RR     II      VV      VV  EE          RR      RR
DD      DD    YY  YY    DD      DD  RR      RR     II      VV      VV  EE          RR      RR
DD      DD    YY  YY    DD      DD  RR      RR     II      VV      VV  EE          RR      RR
DD      DD     YY       DD      DD  RRRRRRRR     II      VV      VV  EEEEEEEE    RRRRRRRR
DD      DD     YY       DD      DD  RRRRRRRR     II      VV      VV  EEEEEEEE    RRRRRRRR
DD      DD     YY       DD      DD  RR  RR       II      VV      VV  EE          RR  RR
DD      DD     YY       DD      DD  RR  RR       II      VV      VV  EE          RR  RR
DD      DD     YY       DD      DD  RR    RR     II        VV  VV    EE          RR    RR        ....
DD      DD     YY       DD      DD  RR    RR     II        VV  VV    EE          RR    RR        ....
DDDDDDDD       YY       DDDDDDDD    RR      RR   IIIIII      VV      EEEEEEEEEE  RR      RR      ....
DDDDDDDD       YY       DDDDDDDD    RR      RR   IIIIII      VV      EEEEEEEEEE  RR      RR      ....

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

Table of contents

```
0000     1          .TITLE  DYDRIVER - VAX/VMS RX211/RX02 DISK DRIVER
0000     2          .IDENT  'V04-000'
0000     3 ;
0000     4 ;*****************************************************************************
0000     5 ;*                                                                           *
0000     6 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8 ;*  ALL RIGHTS RESERVED.                                                     *
0000     9 ;*                                                                           *
0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    11 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    15 ;*  TRANSFERRED.                                                             *
0000    16 ;*                                                                           *
0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    18 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    19 ;*  CORPORATION.                                                             *
0000    20 ;*                                                                           *
0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23 ;*                                                                           *
0000    24 ;*                                                                           *
0000    25 ;*****************************************************************************
0000    26 ;
0000    27 ;++
0000    28 ;
0000    29 ; FACILITY:
0000    30 ;
0000    31 ;       VAX/VMS RX211/RX02 DISK DRIVER
0000    32 ;
0000    33 ; AUTHOR:
0000    34 ;
0000    35 ; C. FRANKS      15-FEB-80
0000    36 ;
0000    37 ; MODIFIED BY:
0000    38 ;
0000    39 ;       V03-007 RAS0300         Ron Schaefer            27-Apr-1984
0000    40 ;               Add DEV$M_NNM characteristic to DECHAR2 so that these
0000    41 ;               devices will have the "node$" prefix.
0000    42 ;
0000    43 ;       V03-006 PRD0034         Paul R. DeStefano       09-Sep-1983
0000    44 ;               Added EXE$LCLDSKVALID to function decision table.
0000    45 ;
0000    46 ;       V03-005 ROW0211         Ralph O. Weber          16-AUG-1983
0000    47 ;               Change device-dependent UCB definition base from UCB$W_BCR+2
0000    48 ;               to UCB$K_LCL_DISK_LENGTH.
0000    49 ;
0000    50 ;       V03-004 KDM0059         Kathleen D. Morse       14-Jul-1983
0000    51 ;               Change WAIT_TR macro to new macro TIMEDWAIT.
0000    52 ;
0000    53 ;       V03-003 ROW53099        Ralph O. Weber          17-FEB-1983
0000    54 ;               Change timeout interval on WFIKPCH in RX211_REINIT from 2
0000    55 ;               seconds to 3 seconds to allow more time for RX211 to
0000    56 ;               initialize.  This corrects conditions which would sometimes
0000    57 ;               cause a transfer to successfully complete with the bytes
```

```
0000     58 ;                transfered count less than the bytes requested count.
0000     59 ;
0000     60 ;        V03-002 KDM0002         Kathleen D. Morse       28-Jun-1982
0000     61 ;                Added $DYNDEF and $VADEF.
0000     62 ;
0000     63 ;        V03-001 KTA0100         Kerbey T. Altmann       07-Jun-1982
0000     64 ;                Add code to set UCB$L_MEDIA_ID.
0000     65 ;
0000     66 ;
```

```
0000     68 ;
0000     69 ; ABSTRACT:
0000     70 ;
0000     71 ;       THIS MODULE CONTAINS THE TABLES AND ROUTINES NECESSARY TO
0000     72 ;       PERFORM ALL DEVICE-DEPENDENT PROCESSING OF AN I/O REQUEST
0000     73 ;       FOR RX211/RX02 AND RX411/RX04 DISK TYPES ON A VAX/VMS SYSTEM.
0000     74 ;
0000     75 ;       THE PYSICAL GEOMETRY OF THE DISKETTES ARE:
0000     76 ;
0000     77 ;                 TRACKS/    SECTORS/     BYTES/     MAXIMUM      DISKETTE
0000     78 ;       #CYL      CYLINDER   TRACK        SECTOR     BLOCKS       TYPE
0000     79 ;
0000     80 ;        77          1          26          128        494         RX02 (SINGLE DEN)
0000     81 ;        77          1          26          256        988         RX02 (DOUBLE DEN)
0000     82 ;        77          1          26          512        1976        RX04 (QUAD DEN)
0000     83 ;        77          2          26          256        1989          *
0000     84 ;
0000     85 ;       SINCE THE SECTOR SIZE IS NOT NECESSARILY ONE BLOCK, AND SINCE
0000     86 ;       SECTORS ARE INTERLEAVED FOR EFFICIENCY, LOGICAL TO PHYSICAL
0000     87 ;       CONVERSION OF THE DISK ADDRESS IS PERFORMED IN THIS DRIVER'S
0000     88 ;       STARTIO ROUTINE RATHER THAN THE IOC$CVTLOGPHY FDT ROUTINE.
0000     89 ;
0000     90 ;       IF VIRTUAL OR LOGICAL I/O IS BEING PERFORMED, SECTOR NUMBERS
0000     91 ;       ARE INTERLEAVED TO OPTIMIZE DATA TRANSFER, AND A SKEW OF SIX
0000     92 ;       SECTORS IS ADDED FOR EACH CYLINDER TO ALLOW FOR SWITCHING TIME.
0000     93 ;       ALSO, THE FIRST TRACK IS SKIPPED FOR INDUSTRY COMPATIBILITY.
0000     94 ;
0000     95 ;       SINGLE SIDED DISKETTES CAN BE RECORDED WITH SINGLE (RX01 COMPATIBLE)
0000     96 ;       OR DOUBLE DENSITY DATA.  DISKETTE DENSITY IS CHANGED VIA THE
0000     97 ;       IO$_FORMAT FUNCTION.  EXISTING DISKETTE DENSITY CAN BE DETERMINED BY
0000     98 ;       EXAMINING UCB$L_MAXBLOCK VIA THE $GETCHN OR $GETDEV SYSTEM SERVICES.
0000     99 ;
0000    100 ;       THE IO$_WRITEPBLK FUNCTION CAN BE ISSUED WITH A "DELETED DATA"
0000    101 ;       MODIFIER WHICH WILL CAUSE A DELETED DATA ADDRESS MARK TO BE
0000    102 ;       WRITTEN PRIOR TO WRITING THE DATA IN EACH SECTOR.  SUBSEQUENT
0000    103 ;       READING OF DATA FROM A SECTOR WITH A DELETED DATA ADDRESS MARK
0000    104 ;       WILL CAUSE THE DATA TO BE RETURNED WITH THE STATUS CODE
0000    105 ;       SS$_RDDELDATA IF SUCCESSFUL.
0000    106 ;
0000    107 ;       IO$_PACKACK MUST BE THE FIRST FUNCTION ISSUED TO A DISKETTE
0000    108 ;       AFTER IT HAS BEEN PLACED IN A DRIVE (TO UPDATE THE UCB
0000    109 ;       WITH THE DISKETTE'S DENSITY AND # SIDES).
0000    110 ;
0000    111 ;       THE RX211 DOES NOT PERFORM EXPLICIT SEEKS, SO OVERLAPPED SEEKS
0000    112 ;       ARE NOT SUPPORTED BY THIS DRIVER.
0000    113 ;
0000    114 ;       THIS DRIVER WILL ONLY SUPPORT RX211 CONTROLLERS WHOSE HARDWARE
0000    115 ;       SWITCH IS IN THE RX02 (NOT RX01) POSITION.
0000    116 ;
0000    117 ;       * NOTE: CODE HAS BEEN INCLUDED FOR A FUTURE DOUBLE SIDED, DOUBLE
0000    118 ;       DENSITY FLOPPY.  IF THIS PRODUCT BECOMES A REALITY, COMPATIBILITY
0000    119 ;       WITH OTHER DEC OPERATING SYSTEMS SHOULD BE CHECKED WITH REGARD TO
0000    120 ;       THE FOLLOWING ASSUMPTIONS MADE BY THIS DRIVER:
0000    121 ;           (1) THE SIX SECTOR SKEW IS APPLIED ONLY WHEN SWITCHING
0000    122 ;               CYLINDERS, NOT WHEN SWITCHING SURFACES.
0000    123 ;           (2) AS WITH OTHER DISKS, ADDRESSES ARE SPIRALLED.  THAT IS, UPON
0000    124 ;               REACHING THE END OF TRACK, THE NEXT SURFACE IS ADDRESSED.  ONLY
```

```
0000    125 ;
0000    126 ;--                 WHEN NO MORE SURFACES REMAIN IS THE NEXT CYLINDER ADDRESSED.
```

```
                        .SBTTL  EXTERNAL AND LOCAL DEFINITIONS

;
; EXTERNAL SYMBOLS
;

        $ADPDEF                         ;DEFINE ADAPTER CONTROL BLOCK
        $CRBDEF                         ;DEFINE CHANNEL REQUEST BLOCK
        $DCDEF                          ;DEFINE DEVICE CLASS
        $DDBDEF                         ;DEFINE DEVICE DATA BLOCK
        $DEVDEF                         ;DEFINE DEVICE CHARACTERISTICS
        $DPTDEF                         ;DEFINE DRIVER PROLOGUE TABLE
        $DYNDEF                         ;DEFINE DYNAMIC DATA STRUCTURES
        $EMBDEF                         ;DEFINE ERROR MESSAGE BUFFER
        $IDBDEF                         ;DEFINE INTERRUPT DATA BLOCK
        $IODEF                          ;DEFINE I/O FUNCTION CODES
        $IRPDEF                         ;DEFINE I/O REQUEST PACKET
        $PRDEF                          ;DEFINE PROCESSOR REGISTERS
        $SSDEF                          ;DEFINE SYSTEM STATUS CODES
        $UCBDEF                         ;DEFINE UNIT CONTROL BLOCK
        $VADEF                          ;DEFINE VIRTUAL ADDRESS FIELDS
        $VECDEF                         ;DEFINE INTERRUPT VECTOR BLOCK

;
; LOCAL MACROS
;

;       DISABLE INTERRUPTS AND CHECK IF POWER HAS FAILED

        .MACRO  CKPWR ?L1
        DSBINT                          ;DISABLE INTERRUPTS
        BBC     #UCB$V_POWER,-          ;IF CLR - NO POWER FAILURE
                UCB$W_STS(R5),L1        ;...
        ENBINT                          ;POWER FAILURE - ENABLE INTERRUPTS
        BRW     PWRFAIL                 ;EXIT
L1:                                     ;RETURN FOR NO POWER FAILURE
        .ENDM


;       CHECK IF DEVICE IS OFFLINE

        .MACRO  CKOFL ?L2,?L3
        BSBW    DY_MERGE                ;MERGE UNIT,DEN,IE,GO,HS,XBA BITS IN R2
        CKPWR                           ;CHECK FOR PWR FAILURE & DSBINT
        BISW3   R2,#F_READSTATUS,RY_CS(R4)  ;EXECUTE READ STATUS FUNCTION
        WFIKPCH L2,#10                  ; Wait for interrupt.
        IOFORK                          ;CREATE FORK PROCESS
L2:
        SETIPL  UCB$B_FIPL(R5)          ; Lower IPL in case due to TIMEOUT.
        BICW    #UCB$M_TIMOUT,UCB$W_STS(R5)  ;CLEAR DEVICE TIMOUT
        BITW    #RY_DB_M_DRDY,UCB$W_DY_DB(R5)  ;IS DRIVE READY?
        BNEQ    L3                      ;IF NEQ - YES, ONLINE
        MOVZWL  #SS$_MEDOFL,R0          ;SET MEDIUM OFFLINE STATUS
```

```
                     0000   185          BRW     FUNCXT                  ;AND EXIT
                     0000   186 L3:                                       ;RETURN FOR DEVICE ONLINE
                     0000   187          .ENDM
                     0000   188
                     0000   189 ;
                     0000   190 ; LOCAL SYMBOLS
                     0000   191 ;
                     0000   192
00000002             0000   193 RY_NUM_REGS     =2                      ;NUMBER OF DEVICE REGISTERS
000001EE             0000   194 RY_SSSD         =494                    ;S SIDED,S DENSITY MAXBLOCKS (26*76/4)
000003DC             0000   195 RY_SSDD         =988                    ;S SIDED,D DENSITY MAXBLOCKS (26*76/2)
000007C5             0000   196 RY_DSDD         =1989                   ;D SIDE,D DEN MXBLK (26*76/2)+(26*77/2)
00000040             0000   197 RY_SWPS         =64                     ;SINGLE DENSITY WORDS/SECTOR
0000001A             0000   198 RY_SECTORS      =26                     ;NUMBER OF SECTORS PER TRACK
0000004D             0000   199 RY_CYLINDERS    =77                     ;NUMBER OF CYLINDERS
00000002             0000   200 RY_RX01SW       =2                      ;UCB$B_DY_ER BIT FOR RX01 SW ERROR
00000001             0000   201 RY_DPPE         =1                      ;UCB$B_DY_ER BIT FOR PURGE ERROR
                     0000   202
                     0000   203 ; Symbols added for RX04 support.
                     0000   204
000007B8             0000   205 RY_SSQD         =1976                   ;S sided,q density maxblocks (26*76)
00000080             0000   206 RY_DWPS         =128                    ;Double density Words/sector.
00000100             0000   207 RY_QWPS         =256                    ;Quad density WORDS/SECTOR.
00000000             0000   208 RY_DENSITY_SINGLE=0                     ;Value to insert in RY_CS register.
00000001             0000   209 RY_DENSITY_DOUBLE=1                     ;       "  "
00000002             0000   210 RY_DENSITY_QUAD =2                      ;       "  "
                     0000   211
                     0000   212 ;
                     0000   213 ; UCB OFFSETS WHICH FOLLOW THE STANDARD UCB FIELDS
                     0000   214 ;
                     0000   215          $DEFINI UCB                     ;START OF UCB DEFINITIONS
                     0000   216
000000CC             0000   217 .=UCB$K_LCL_DISK_LENGTH                   ;BEGIN DEFINITIONS AT END OF UCB
                     00CC   218 $DEF     UCB$W_DY_WPS     .BLKW   1      ;Words per sector.
                     00CE   219 $DEF     UCB$W_DY_CS      .BLKW   1      ;CONTROL STATUS REGISTER
                     00D0   220 $DEF     UCB$W_DY_DB      .BLKW   1      ;DATA BUFFER REGISTER
                     00D2   221 $DEF     UCB$W_DY_DPN     .BLKW   1      ;DATA PATH NUMBER
                     00D4   222 $DEF     UCB$L_DY_DPR     .BLKL   1      ;DATAPATH REGISTER
                     00D8   223 $DEF     UCB$L_DY_FMPR    .BLKL   1      ;FINAL MAP REGISTER
                     00DC   224 $DEF     UCB$L_DY_PMPR    .BLKL   1      ;PREVIOUS MAP REGISTER
                     00E0   225 $DEF     UCB$B_DY_ER      .BLKB   1      ;SPECIAL ERROR REGISTER
000000E2             00E1   226                          .BLKB   1      ;Reserved.
                     00E2   227 $DEF     UCB$B_DY_LCT     .BLKB   1      ;LOOP COUNTER
                     00E3   228 $DEF     UCB$B_DY_XBA     .BLKB   1      ;BUS ADDRESS EXTENSION BITS
                     00E4   229 $DEF     UCB$W_DY_PWC     .BLKW   1      ;PARTIAL WORD COUNT
                     00E6   230 $DEF     UCB$W_DY_SBA     .BLKW   1      ;SAVED BUFFER ADDRESS
                     00E8   231 $DEF     UCB$L_DY_XFER    .BLKL   1      ;TRANSFER FUNCTION CSR BITS
                     00EC   232 $DEF     UCB$L_DY_LMEDIA  .BLKL   1      ;LOGICAL MEDIA ADDRESS
                     00F0   233 $DEF     UCB$Q_DY_EXTENDED_STATUS        ; Area into which we do READ ERROR
000000F8             00F0   234                          .BLKQ   1      ;  REGISTER command.
                     00F8   235
00000008             00F8   236          RY_EXTENDED_STATUS_LENGTH = .-UCB$Q_DY_EXTENDED_STATUS
                     00F8   237
                     00F8   238 $DEF     UCB$Q_DY_SVAPTETMP              ; Area in which we save UCB fields -
00000100             00F8   239                          .BLKQ   1      ;  SVAPTE, BOFF, and BCNT.
                     0100   240 $DEF     UCB$L_DY_MAPREGTMP              ; Area in which we save CRB fields -
00000104             0100   241                          .BLKL   1      ;  MAPREG, NUMREG, and DATAPATH.
```

```
                     0104   242 $DEF    UCB$L_DY_SAVECS .BLKL   1       ; Area in which we save CS and DB regs.
                     0108   243
00000108             0108   244 UCB$K_DY_LEN=.                          ;LENGTH OF UCB
                     0108   245
                     0108   246         $DEFEND UCB                     ;END OF UCB DEFINITONS
                     0000   247
                     0000   248 ;
                     0000   249 ; RX211/RX02 REGISTER OFFSETS FROM CSR ADDRESS
                     0000   250 ;
                     0000   251         $DEFINI RY                      ; START OF REGISTER DEFINITIONS
                     0000   252
                     0000   253 $DEF     RY_CS           .BLKW   1       ;CONTROL STATUS REGISTER (CSR)
                     0002   254         _VIELD  RY_CS,0,<-              ;START OF CSR BIT DEFINITIONS
                     0002   255                 <GO,,M>,-               ; GO
                     0002   256                 <FCODE,3>,-             ; FUNCTION CODE
                     0002   257                 <US,,M>,-               ; UNIT SELECT
                     0002   258                 <DONE,,M>,-             ; DONE - FUNCTION COMPLETE
                     0002   259                 <IE,,M>,-               ; INTERRUPT ENABLE
                     0002   260                 <TR,,M>,-               ; TRANSFER REQUEST
                     0002   261                 <DEN,2>,-               ; Density
                     0002   262                 <,1>,-                  ; RESERVED BIT
                     0002   263                 <RX02,,M>,-             ; DEVICE TYPE
                     0002   264                 <XBA,2>,-               ; BUS ADDRESS EXTENSION BITS
                     0002   265                 <INIT,,M>,-             ; INITIALIZE
                     0002   266                 <ERR,,M>-               ; ERROR
                     0002   267         >                               ;END CSR BIT DEFINITIONS
                     0002   268
                     0002   269 $DEF     RY_DB           .BLKW   1       ;DATA BUFFER REGISTER (DBR)
                     0004   270         _VIELD  RY_DB,0,<-              ;START OF DBR BIT DEFINITIONS
                     0004   271                 <CRC,,M>,-              ; CRC ERROR
                     0004   272                 <QDEN,,M>,-             ; Quad density
                     0004   273                 <ID,,M>,-               ; INITIALIZE DONE
                     0004   274                 <ACLO,,M>,-             ; AC PWR FAILURE
                     0004   275                 <DE,,M>,-               ; DENSITY ERROR
                     0004   276                 <DDEN,,M>,-             ; DRIVE DENSITY
                     0004   277                 <DELD,,M>,-             ; DELETED DATA
                     0004   278                 <DRDY,,M>,-             ; DRIVE READY
                     0004   279                 <US,,M>,-               ; UNIT SELECT
                     0004   280                 <RX04,,M>,-             ; RX04 bit
                     0004   281                 <WCO,,M>,-              ; WORD COUNT OVERFLOW
                     0004   282                 <NXM,,M>,-              ; NON-EXISTENT MEMORY
                     0004   283                 <,4>-                   ; RESERVED BITS
                     0004   284         >                               ;END DBR BIT DEFINITIONS
                     0004   285
                     0004   286         $DEFEND RY                      ;END RX211/RX02,RX03 REGISTER DEFINITIONS
                     0000   287
                     0000   288 ;
                     0000   289 ; HARDWARE FUNCTION CODES
                     0000   290 ;
                     0000   291
00000000             0000   292 F_FILLBUFFER    =0*2                    ;FILL BUFFER
00000002             0000   293 F_EMPTYBUFFER   =1*2                    ;EMPTY BUFFER
00000004             0000   294 F_WRITESECTOR   =2*2                    ;WRITE SECTOR
00000006             0000   295 F_READSECTOR    =3*2                    ;READ SECTOR
00000008             0000   296 F_SETDEN        =4*2                    ;SET DENSITY
0000000A             0000   297 F_READSTATUS    =5*2                    ;READ STATUS
0000000C             0000   298 F_WRITEDEL      =6*2                    ;WRITE DELETED DATA
```

```
0000000E  0000   299 F_READERROR     =7*2                    ;Read Error Register.
```

```
                        .SBTTL  STANDARD TABLES

;
; DRIVER PROLOGUE TABLE
;
;       THE DPT DESCRIBES DRIVER PARAMETERS AND I/O DATABASE FIELDS
;       THAT ARE TO BE INITIALIZED DURING DRIVER LOADING AND RELOADING
;

        DPTAB   -                               ;DPT CREATION MACRO
                END=DY_END,-                    ;END OF DRIVER LABEL
                ADAPTER=UBA,-                   ;ADAPTER TYPE = UNIBUS
                FLAGS=DPT$M_SVP,-               ;SYSTEM PAGE TABLE ENTRY REQUIRED
                DEFUNITS=2,-                    ;UNITS 0 AND 1
                UCBSIZE=UCB$K_DY_LEN,-          ;LENGTH OF UCB
                NAME=DYDRIVER                   ;DRIVER NAME

        DPT_STORE INIT                          ;START CONTROL BLOCK INIT VALUES
        DPT_STORE DDB,DDB$L_ACPD,L,<^A\F11\>  ;DEFAULT ACP NAME
        DPT_STORE DDB,DDB$L_ACPD+3,B,DDB$K_SLOW ;ACP CLASS
        DPT_STORE UCB,UCB$B_FIPL,B,8            ;FORK IPL
        DPT_STORE UCB,UCB$L_DEVCHAR,L,-         ;DEVICE CHARACTERISTICS
                <DEV$M_FOD-                     ; FILES ORIENTED
                !DEV$M_DIR-                     ; DIRECTORY STRUCTURED
                !DEV$M_AVL-                     ; AVAILABLE
                !DEV$M_ELG-                     ; ERROR LOGGING
                !DEV$M_SHR-                     ; SHAREABLE
                !DEV$M_IDV-                     ; INPUT DEVICE
                !DEV$M_ODV-                     ; OUTPUT DEVICE
                !DEV$M_RND>                     ; RANDOM ACCESS
        DPT_STORE UCB,UCB$L_DEVCHAR2,L,-        ; DEVICE CHARACTERISTICS
                <DEV$M_NNM>                     ; PREFIX NAME WITH "node$"
        DPT_STORE UCB,UCB$B_DEVCLASS,B,DC$_DISK  ;DEVICE CLASS
        DPT_STORE UCB,UCB$W_DEVBUFSIZ,W,512  ;DEFAULT BUFFER SIZE
        DPT_STORE UCB,UCB$B_SECTORS,B,26   ;NUMBER OF SECTORS PER TRACK
        DPT_STORE UCB,UCB$W_CYLINDERS,W,77   ;NUMBER OF TRACKS PER CYLINDER
        DPT_STORE UCB,UCB$B_DIPL,B,21       ;DEVICE IPL
        DPT_STORE UCB,UCB$B_ERTMAX,B,10 ;MAX ERROR RETRY COUNT
        DPT_STORE UCB,UCB$W_DEVSTS,W,-  ;INHIBIT LOG TO PHYS CONVERSION IN FDT
                <UCB$M_NOCNVRT>         ;...

        DPT_STORE REINIT                        ;START CONTROL BLOCK RE-INIT VALUES
        DPT_STORE CRB,CRB$L_INTD+4,D,DY_INT  ;INTERRUPT SERVICE ROUTINE ADDRESS
        DPT_STORE CRB,CRB$L_INTD+VEC$L_INITIAL,-  ;CONTROLLER INIT ADDRESS
                D,DY_RX211_INIT                 ;...
        DPT_STORE CRB,CRB$L_INTD+VEC$L_UNITINIT,- ;UNIT INIT ADDRESS
                D,DY_RX02_INIT                  ;...
        DPT_STORE DDB,DDB$L_DDT,D,DY$DDT        ;DDT ADDRESS

        DPT_STORE END                   ;END OF INITIALIZATION TABLE

;
; DRIVER DISPATCH TABLE
;
;       THE DDT LISTS ENTRY POINTS FOR DRIVER SUBROUTINES WHICH ARE
;       CALLED BY THE OPERATING SYSTEM.
;
```

```
                     ;
                     ;
                     DDTAB   -                               ;DDT CREATION MACRO
                             DEVNAM=DY,-                     ;NAME OF DEVICE
                             START=DY_STARTIO,-              ;START I/O ROUTINE
                             UNSOLIC=DY_UNSOLNT,-            ;UNSOLICITED INTERRUPT
                             FUNCTB=DY_FUNCTABLE,-           ;FUNCTION DECISION TABLE
                             CANCEL=0,-                      ;CANCEL=NO-OP FOR FILES DEVICE
                             REGDMP=DY_REGDUMP,-             ;REGISTER DUMP ROUTINE
                             DIAGBF=<<RY_NUM_REGS+7+5+3+1>*4>,-  ;BYTES IN DIAG BUFFER
                             ERLGBF=<<<RY_NUM_REGS+7+1>*4>+<EMB$L_DV_REGSAV>>;BYTES IN
                                                                             ;ERRLOG BUFFER
                     ; DIAGNOSTIC BUFFER SIZE = <<2 RX02 REGISTER LONGWORDS + 7 UCB FIELD LONGWORDS
                     ;                          + 5 IOC$DIAGBUFILL LONGWORDS + 3 BUFFER ALLOCATION
                     ;                          LONGWORDS + 1 LONGWORD FOR # REGISTERS IN DY_REGDUMP>
                     ;                          * 4 BYTES/LONGWORD>
                     ;
                     ; ERROR LOG BUFFER SIZE  = <<<2 RX02 REGISTER LONGWORDS + 7 UCB FIELD LONGWORDS
                     ;                          +1 LONGWORD FOR # REGISTERS IN DY_REGDUMP>
                     ;                          * 4 BYTES/LONGWORD> + BYTES NEEDED FOR ERROR LOGGER
                     ;                          TO SAVE SOFTWARE REGISTERS>
```

```
                     ;
                     ; FUNCTION DECISION TABLE
                     ;
                     ;       THE FDT LISTS VALID FUNCTION CODES, SPECIFIES WHICH
                     ;       CODES ARE BUFFERED, AND DESIGNATES SUBROUTINES TO
                     ;       PERFORM PREPROCESSING FOR PARTICULAR FUNCTIONS.
                     ;

                     DY_FUNCTABLE:
                             FUNCTAB ,-                      ;LIST LEGAL FUNCTIONS
                                     <FORMAT,-               ; SET MEDIA DENSITY AND REFORMAT DISK
                                     UNLOAD,-                ; UNLOAD
                                     PACKACK,-               ; PACK ACKNOWLEDGE
                                     AVAILABLE,-             ; AVAILABLE
                                     SENSECHAR,-             ; SENSE CHARACTERISTICS
                                     SETCHAR,-               ; SET CHARACTERISTICS
                                     SENSEMODE,-             ; SENSE MODE
                                     SETMODE,-               ; SET MODE
                                     READLBLK,-              ; READ LOGICAL BLOCK
                                     WRITELBLK,-             ; WRITE LOGICAL BLOCK
                                     READPBLK,-              ; READ PHYSICAL BLOCK
                                     WRITEPBLK,-             ; WRITE PHYSICAL BLOCK
                                     READVBLK,-              ; READ VIRTUAL BLOCK
                                     WRITEVBLK,-             ; WRITE VIRTUAL BLOCK
                                     ACCESS,-                ; ACCESS FILE / FIND DIRECTORY ENTRY
                                     ACPCONTROL,-            ; ACP CONTROL FUNCTION
                                     CREATE,-                ; CREATE FILE AND/OR DIRECTORY ENTRY
                                     DEACCESS,-              ; DEACCESS FILE
                                     DELETE,-                ; DELETE FILE AND/OR DIRECTORY ENTRY
                                     MODIFY,-                ; MODIFY FILE ATTRIBUTES
                                     MOUNT-                  ; MOUNT VOLUME
                                     >
                             FUNCTAB ,-                      ;BUFFERED FUNCTIONS
                                     <FORMAT,-               ; FORMAT
                                     UNLOAD,-                ; UNLOAD
                                     PACKACK,-               ; PACK ACKNOWLEDGE
                                     AVAILABLE,-             ; AVAILABLE
                                     SENSECHAR,-             ; SENSE CHARACTERISTICS
                                     SETCHAR,-               ; SET CHARACTERISTICS
                                     SENSEMODE,-             ; SENSE MODE
                                     SETMODE,-               ; SET MODE
                                     ACCESS,-                ; ACCESS FILE / FIND DIRECTORY ENTRY
                                     ACPCONTROL,-            ; ACP CONTROL FUNCTION
                                     CREATE,-                ; CREATE FILE AND/OR DIRECTORY ENTRY
                                     DEACCESS,-              ; DEACCESS FILE
                                     DELETE,-                ; DELETE FILE AND/OR DIRECTORY ENTRY
                                     MODIFY,-                ; MODIFY FILE ATTRIBUTES
                                     MOUNT-                  ; MOUNT VOLUME
                                     >
                             FUNCTAB DY_ALIGN,-              ;TEST ALIGNMENT FUNCTIONS
                                     <READLBLK,-             ; READ LOGICAL BLOCK
                                     READPBLK,-              ; READ PHYSICAL BLOCK
                                     READVBLK,-              ; READ VIRTUAL BLOCK
                                     WRITELBLK,-             ; WRITE LOGICAL BLOCK
                                     WRITEPBLK,-             ; WRITE PHYSICAL BLOCK
                                     WRITEVBLK-              ; WRITE VIRTUAL BLOCK
                                     >
```

```
                        436          FUNCTAB +ACP$READBLK,-          ;READ FUNCTIONS
                        437                  <READLBLK,-             ; READ LOGICAL BLOCK
                        438                  READPBLK,-              ; READ PHYSICAL BLOCK
                        439                  READVBLK-               ; READ VIRTUAL BLOCK
                        440                  >
                        441          FUNCTAB +ACP$WRITEBLK,-         ;WRITE FUNCTIONS
                        442                  <WRITELBLK,-            ; WRITE LOGICAL BLOCK
                        443                  WRITEPBLK,-             ; WRITE PHYSICAL BLOCK
                        444                  WRITEVBLK-              ; WRITE VIRTUAL BLOCK
                        445                  >
                        446          FUNCTAB +ACP$ACCESS,-           ;ACCESS FUNCTIONS
                        447                  <ACCESS,-               ; ACCEESS FILE / FIND DIRECTORY ENTRY
                        448                  CREATE-                 ; CREATE FILE AND/OR DIRECTORY ENTRY
                        449                  >
                        450          FUNCTAB +ACP$DEACCESS,-         ;DEACCESS FUNCTION
                        451                  <DEACCESS-              ; DEACCESS FILE
                        452                  >
                        453          FUNCTAB +ACP$MODIFY,-           ;MODIFY FUNCTIONS
                        454                  <ACPCONTROL,-           ; ACP CONTROL FUNCTION
                        455                  DELETE,-                ; DELETE FILE AND/OR DIRECTORY ENTRY
                        456                  MODIFY-                 ; MODIFY FILE ATTRIBUTES
                        457                  >
                        458          FUNCTAB +ACP$MOUNT,-            ;MOUNT FUNCTION
                        459                  <MOUNT-                 ; MOUNT VOLUME
                        460                  >
                        461          FUNCTAB +EXE$LCLDSKVALID,-      ;LOCAL DISK VALID FUNCTIONS
                        462                  <UNLOAD,-               ;UNLOAD VOLUME
                        463                   AVAILABLE,-            ;UNIT AVAILABLE
                        464                   PACKACK-               ;PACK ACKNOWLEDGE
                        465                  >
                        466          FUNCTAB +EXE$ZEROPARM,-         ;ZERO PARAMETER FUNCTIONS
                        467                  <UNLOAD,-               ; UNLOAD
                        468                  PACKACK,-               ; PACK ACKNOWLEDGE
                        469                  AVAILABLE,-             ; AVAILABLE
                        470                  >
                        471          FUNCTAB +EXE$ONEPARM,-          ;ONE PARAMETER FUNCTION
                        472                  <FORMAT-                ; FORMAT
                        473                  >
                        474          FUNCTAB +EXE$SENSEMODE,-        ;SENSE FUNCTIONS
                        475                  <SENSECHAR,-            ; SENSE CHARACTERISTICS
                        476                  SENSEMODE-              ; SENSE MODE
                        477                  >
                        478          FUNCTAB +EXE$SETCHAR,-          ;SET FUNCTIONS
                        479                  <SETCHAR,-              ; SET CHARACTERISTICS
                        480                  SETMODE-                ; SET MODE
                        481                  >
```

```
                     00D8   483          .SBTTL  CONTROLLER INITIALIZATION ROUTINE
                     00D8   484
                     00D8   485 ; ++
                     00D8   486 ;
                     00D8   487 ; DY_RX211_INIT - CONTROLLER INITIALIZATION ROUTINE
                     00D8   488 ;
                     00D8   489 ; FUNCTIONAL DESCRIPTION:
                     00D8   490 ;
                     00D8   491 ;       THIS ROUTINE INITIALIZES THE RX211 CONTROLLER FOR I/O OPERATIONS.
                     00D8   492 ;       IF THE INITIALIZATION IS NOT COMPLETE WITHIN ONE SECOND, CONTROL
                     00D8   493 ;       IS RETURNED TO THE CALLER.
                     00D8   494 ;
                     00D8   495 ;       THE OPERATING SYSTEM CALLS THIS ROUTINE:
                     00D8   496 ;               - AT SYSTEM STARTUP
                     00D8   497 ;               - DURING DRIVER LOADING
                     00D8   498 ;               - DURING RECOVERY FROM POWER FAILURE
                     00D8   499 ;       THE DRIVER CALLS THIS ROUTINE TO INIT AFTER AN NXM ERROR.
                     00D8   500 ;
                     00D8   501 ; INPUTS:
                     00D8   502 ;
                     00D8   503 ;       R4      - CSR ADDRESS (CONTROLLER STATUS REGISTER)
                     00D8   504 ;       R5      - IDB ADDRESS (INTERRUPT DATA BLOCK)
                     00D8   505 ;
                     00D8   506 ; OUTPUTS:
                     00D8   507 ;
                     00D8   508 ;       THE HEADS FOR ALL DRIVES CONNECTED TO THIS CONTROLLER ARE LOCATED AT
                     00D8   509 ;       TRACK ZERO, AND THE ERROR AND STATUS REGISTER IS CLEARED.
                     00D8   510 ;       ALL GENERAL REGISTERS (R0 - R15) ARE PRESERVED.
                     00D8   511 ;
                     00D8   512 ;--
                     00D8   513
                     00D8   514 DY_RX211_INIT:                          ;RX211 CONTROLLER INITIALIZATION
          7E   50    7D 00D8   515          MOVQ    R0,-(SP)            ;SAVE R0-R1
    64    4000 8F    B0 00DB   516          MOVW    #RY_CS_M_INIT,RY_CS(R4) ;EXECUTE RX211 INITIALIZATION
                     00E0   517          TIMEDWAIT TIME=#100*1000,-      ;ONE SECOND WAIT LOOP
                     00E0   518                    INS1=<BITW      #RY_CS_M_DONE,RY_CS(R4)>,- ;DONE ?
                     00E0   519                    INS2=<BNEQ      10$>,-  ;IF NEQ = YES
                     00E0   520                    DONELBL=10$             ;DONE LABEL
          50   8E    7D 0107   521 20$:     MOVQ    (SP)+,R0            ;RESTORE R0-R1
                     05 010A   522          RSB                         ;RETURN
```

```
                                010B   524          .SBTTL  INTERNAL CONTROLLER RE-INITIALIZATION
                                010B   525
                                010B   526 ;++
                                010B   527 ;
                                010B   528 ; RX211_REINIT - Internal subroutine used to issue an RX211 initialize function
                                010B   529 ;       without hanging on at elevated IPL waiting for it to finish.
                                010B   530 ;       Because the RX211 initialize does not interrupt when complete,
                                010B   531 ;       we rely upon a device timeout to resume the driver thread after
                                010B   532 ;       invoking the initialize.
                                010B   533 ;
                                010B   534 ; INPUTS:
                                010B   535 ;       R4 => RX211 CSR
                                010B   536 ;       R5 => UCB
                                010B   537 ;
                                010B   538
                                010B   539 RX211_REINIT:
                     53 8ED0    010B   540          POPL    R3                          ; Save return point.
                                010E   541          DSBINT
                64   4000 8F    B0 0114   542          MOVW    #RY_CS_M_INIT,RY_CS(R4) ; Execute RX211 initialize.
                                0119   543          WFIKPCH 10$,#3                  ; Wait for interrupt that doesn't come.
                                0123   544          IOFORK                          ; We should never come here.
                                0129   545 10$:
                                0129   546          SETIPL  UCB$B_FIPL(R5)          ; Lower to fork level.
                     0040 8F    AA 012D   547          BICW    #UCB$M_TIMOUT,-         ; Clear timeout status.
                       64 A5       0131   548                  UCB$W_STS(R5)
                          63    17 0133   549          JMP     (R3)                    ; Return to caller.
```

```
                     0135   551          .SBTTL  UNIT INITIALIZATION ROUTINE
                     0135   552
                     0135   553 ;++
                     0135   554 ;
                     0135   555 ; DY_RX02_INIT - UNIT INITIALIZATION ROUTINE
                     0135   556 ;
                     0135   557 ; FUNCTIONAL DESCRIPTION:
                     0135   558 ;
                     0135   559 ;       THIS ROUTINE SETS THE RX02 UNIT ONLINE.
                     0135   560 ;
                     0135   561 ;       NO ATTEMPT IS MADE TO READ THE DENSITY, OR # SIDES OF THE UNIT IN
                     0135   562 ;       THIS ROUTINE SINCE THE DRIVE MUST BE LOADED WITH A DISKETTE FOR
                     0135   563 ;       THAT OPERATION TO BE VALID.  THESE CHARACTERISTICS CAN BE UPDATED IN
                     0135   564 ;       THE UCB BY ISSUING AN IO$_PACKACK FUNCTION.
                     0135   565 ;
                     0135   566 ;       THE OPERATING SYSTEM CALLS THIS ROUTINE:
                     0135   567 ;               - AT SYSTEM STARTUP
                     0135   568 ;               - DURING DRIVER LOADING
                     0135   569 ;               - DURING RECOVERY FROM POWER FAILURE
                     0135   570 ;
                     0135   571 ; INPUTS:
                     0135   572 ;
                     0135   573 ;       R4      - CSR ADDRESS (CONTROLLER STATUS REGISTER)
                     0135   574 ;       R5      - UCB ADDRESS (UNIT CONTROL BLOCK)
                     0135   575 ;
                     0135   576 ; OUTPUTS:
                     0135   577 ;
                     0135   578 ;       THE UNIT IS SET ONLINE.
                     0135   579 ;       ALL GENERAL REGISTERS (R0-R15) ARE PRESERVED.
                     0135   580 ;
                     0135   581 ;--
                     0135   582
                     0135   583 DY_RX02_INIT:                                ;RX02 UNIT INITIALIZATION
                     0135   584
64 A5   10   A8      0135   585          BISW    #UCB$M_ONLINE,UCB$W_STS(R5)  ;SET UCB STATUS ONLINE
40 A5   01   90      0139   586          MOVB    #DC$_DISK,UCB$B_DEVCLASS(R5) ;SET DISK DEVICE CLASS
41 A5   0B   90      013D   587          MOVB    #DT$_RX02,UCB$B_DEVTYPE(R5)  ;ASSUME RX02 DEVICE TYPE
                05   0141   588          RSB                                  ;RETURN
```

```
                                0142   590          .SBTTL  DRIVER SPECIFIC SUBROUTINES
                                0142   591 ;
                                0142   592 ; DY_MERGE - MERGE CSR BITS
                                0142   593 ;
                                0142   594 ; FUNCTIONAL DESCRIPTION:
                                0142   595 ;
                                0142   596 ;       THIS ROUTINE IS CALLED FROM THE STARTIO HARDWARE FUNCTION
                                0142   597 ;       EXECUTION ROUTINES TO MERGE THE GO, UNITSELECT, INTERRUPT ENABLE,
                                0142   598 ;       AND DENSITY BITS IN R2 PRIOR TO INITIATING THE INTENDED I/O FUNCTION.
                                0142   599 ;
                                0142   600 ; INPUTS:
                                0142   601 ;
                                0142   602 ;       R5      - UCB ADDRESS
                                0142   603 ;
                                0142   604 ; OUTPUTS:
                                0142   605 ;
                                0142   606 ;       R2 CONTAINS THE CSR BITS FOR: GO, UNIT, IE, AND DENSITY.
                                0142   607 ;       ALL REGISTERS EXCEPT R0 AND R2 ARE PRESERVED.
                                0142   608 ;
                                0142   609
                                0142   610 DY_MERGE:                                    ;MERGE CSR BITS IN R2
            52   0041 8F    B0  0142   611          MOVW    #RY_CS_M_GO!RY_CS_M_IE,R2  ;SET GO AND IE BITS IN R2
      52    01    04   54 A5  F0  0147   612          INSV    UCB$W_UNIT(R5),#4,#T,R2 ;MERGE UNIT NUMBER IN R2<4>
                                014D   613          ASSUME  RY_DENSITY_SINGLE  EQ  0
        00B0 C5   01EE 8F    B1  014D   614          CMPW    #RY_SSSD,UCB$L_MAXBLOCK(R5)  ;SINGLE DENSITY?
                        13   13  0154   615          BEQL    10$                         ;IF EQL - YES
                        02   F0  0156   616          INSV    #RY_DENSITY_QUAD,-          ; Setup as if we have QUAD density
                        08       0158   617                  #RY_CS_V_DEN,-
                   52   02       0159   618                  #RY_CS_S_DEN,R2
                   07B8 8F  B1  015B   619          CMPW    #RY_SSQD,-
                   00B0 C5      015F   620                  UCB$L_MAXBLOCK(R5)          ; See if indeed QUAD density.
                        05  13  0162   621          BEQL    10$                         ; If QUAD, then we are all set.
                        01  F0  0164   622          INSV    #RY_DENSITY_DOUBLE,-        ; Else must be DOUBLE density so
                        08       0166   623                  #RY_CS_V_DEN,-              ;  setup CSR register value accordingly.
                   52   02       0167   624                  #RY_CS_S_DEN,R2
                            05  0169   625 10$:    RSB                                 ;RETURN
```

```
                        016A   627          .SBTTL  FDT ROUTINES
                        016A   628 ;++
                        016A   629 ;
                        016A   630 ; DY_ALIGN - FDT ROUTINE TO TEST XFER BYTE COUNT
                        016A   631 ;
                        016A   632 ; FUNCTIONAL DESCRIPTION:
                        016A   633 ;
                        016A   634 ;       THIS ROUTINE IS CALLED FROM THE FUNCTION DECISION TABLE DISPATCHER
                        016A   635 ;       TO CHECK THE BYTE COUNT PARAMETER SPECIFIED BY THE USER PROCESS
                        016A   636 ;       FOR AN EVEN NUMBER OF BYTES (WORD BOUNDARY).
                        016A   637 ;
                        016A   638 ; INPUTS:
                        016A   639 ;
                        016A   640 ;       R3      - IRP ADDRESS (I/O REQUEST PACKET)
                        016A   641 ;       R4      - PCB ADDRESS (PROCESS CONTROL BLOCK)
                        016A   642 ;       R5      - UCB ADDRESS (UNIT CONTROL BLOCK)
                        016A   643 ;       R6      - CCB ADDRESS (CHANNEL CONTROL BLOCK)
                        016A   644 ;       R7      - BIT NUMBER OF THE I/O FUNCTION CODE
                        016A   645 ;       R8      - ADDRESS OF FDT TABLE ENTRY FOR THIS ROUTINE
                        016A   646 ;       4(AP)   - ADDRESS OF FIRST FUNCTION DEPENDENT QIO PARAMETER
                        016A   647 ;
                        016A   648 ; OUTPUTS:
                        016A   649 ;
                        016A   650 ;       IF THE QIO BYTE COUNT PARAMETER IS ODD, THE I/O OPERATION IS
                        016A   651 ;       TERMINATED WITH AN ERROR. IF IT IS EVEN, CONTROL IS RETURNED
                        016A   652 ;       TO THE FDT DISPATCHER.
                        016A   653 ;
                        016A   654 ;--
                        016A   655
                        016A   656 DY_ALIGN:                                    ;CHECK BYTE COUNT AT P1(AP)
      01 04 AC    E8    016A   657          BLBS    4(AP),10$                  ;IF LBS - ODD BYTE COUNT
                  05    016E   658          RSB                                ;EVEN - RETURN TO CALLER
50    034C 8F     3C    016F   659 10$:     MOVZWL  #SS$_IVBUFLEN,R0           ;SET BUFFER ALIGNMENT STATUS
 00000000'GF      17    0174   660          JMP     G^EXE$ABORTIO              ;ABORT I/O
```

```
                                        .SBTTL  START I/O ROUTINE
                                ;++
                                ;
                                ; DY_STARTIO - START I/O ROUTINE
                                ;
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ;       THIS FORK PROCESS IS ENTERED FROM THE EXECUTIVE AFTER AN I/O REQUEST
                                ;       PACKET HAS BEEN DEQUEUED, AND PERFORMS THE FOLLOWING:
                                ;
                                ;               - ACTIVATES THE DISK AFTER SETTING UCB FIELDS, OBTAINING
                                ;                 UBA AND CONTROLLER RESOURCES, AND SETTING RX211 REGISTERS
                                ;
                                ;               - WAITS FOR AN INTERRUPT
                                ;
                                ;               - REGAINS CONTROL AFTER THE ISR SERVICES THE INTERRUPT, AND
                                ;                       -  RE-ACTIVATES THE DISK IF THE ORIGINAL FUNCTION
                                ;                          IS NOT YET COMPLETE, OR
                                ;                       -  COMPLETES THE I/O REQUEST BY RELEASING RESOURCES,
                                ;                          SETTING STATUS CODES, AND RETURNING TO THE EXECUTIVE.
                                ;
                                ; INPUTS:
                                ;
                                ;       R3              - IRP ADDRESS (I/O REQUEST PACKET)
                                ;       R5              - UCB ADDRESS (UNIT CONTROL BLOCK)
                                ;       IRP$L_MEDIA     - PARAMETER LONGWORD (LOGICAL BLOCK NUMBER)
                                ;
                                ; OUTPUTS:
                                ;
                                ;       R0      - FIRST I/O STATUS LONGWORD: STATUS CODE & BYTES XFERED
                                ;       R1      - SECOND I/O STATUS LONGWORD: 0 FOR DISKS
                                ;
                                ;       THE I/O FUNCTION IS EXECUTED.
                                ;
                                ;       ALL REGISTERS EXCEPT R0-R4 ARE PRESERVED.
                                ;
                                ;--
                                
                                DY_STARTIO:                             ;START I/O OPERATION
                                
                                ;
                                ;       PREPROCESS UCB FIELDS
                                ;
                                
                                        ASSUME  RY_EXTENDED_STATUS_LENGTH  EQ  8
00F0 C5   7C                    CLRQ    UCB$Q_DY_EXTENDED_STATUS(R5)    ; Zero READ ERROR REGISTER area.
                                
0081 C5   90                    MOVB    UCB$B_ERTMAX(R5),-      ;INITIALIZE ERROR RETRY COUNT
0080 C5                                 UCB$B_ERTCNT(R5)        ;...
00C0 C5   7E A5   AE            MNEGW   UCB$W_BCNT(R5),UCB$W_BCR(R5)  ;INIT NEG BYTES LEFT TO XFER
00D2 C5   B4                    CLRW    UCB$W_DY_DPN(R5)        ;CLEAR DATA PATH NO. FOR USE AS-
                                                                ;UBA RESOURCE ALLOCATION FLAG
00E0 C5   94                    CLRB    UCB$B_DY_ER(R5)         ;CLEAR SPECIAL ERROR REGISTER
009A C5   20 A3   B0            MOVW    IRP$W_FUNC(R3),UCB$W_FUNC(R5)  ;SAVE FUNCTION CODE
                  00   EF       EXTZV   #IRP$V_FCODE,-          ;EXTRACT I/O FUNCTION CODE
51   20 A3   06                         #IRP$S_FCODE,IRP$W_FUNC(R3),R1  ;...
```

```
                     0092 C5   51    90   019F   719          MOVB    R1,UCB$B_FEX(R5)          ;STORE FUNCTION DISPATCH INDEX
                       68 A5   02    AA   01A4   720          BICW    #UCB$M_DIAGBUF,UCB$W_DEVSTS(R5)  ;CLR DIAGNOSTIC BUFFER PRESENT
                               07    E1   01A8   721          BBC     #IRP$V_DIAGBUF,-          ;IF CLR - NO DIAG BUFFER
                            04 2A A3         01AA   722                  IRP$W_STS(R3),10$         ;...
                       68 A5   02    A8   01AD   723          BISW    #UCB$M_DIAGBUF,UCB$W_DEVSTS(R5)  ;SET DIAG BUFFER PRESENT
                                           01B1   724
                                           01B1   725 ;
                                           01B1   726 ;       BRANCH TO FUNCTION EXECUTION
                                           01B1   727 ;
                                           01B1   728
                               08    E0   01B1   729 10$:     BBS     #IRP$V_PHYSIO,-           ;IF SET - PHYSICAL I/O FUNCTION
                            0D 2A A3         01B3   730                  IRP$W_STS(R3),20$         ;...
                               0B    E0   01B6   731          BBS     #UCB$V_VALID,-            ;IF SET - VOLUME SOFTWARE VALID
                            08 64 A5         01B8   732                  UCB$W_STS(R5),20$         ;...
                     50   0254 8F    3C   01BB   733          MOVZWL  #SS$_VOLINV,R0            ;SET VOLUME INVALID STATUS
                             0613    31   01C0   734          BRW     RESETXFR                  ;RESET BYTE COUNT AND EXIT
                           51   01    91   01C3   735 20$:     CMPB    #IO$_UNLOAD, R1           ;Unload function?
                               08    13   01C6   736          BEQL    UNLOAD                    ;Branch if yes.
                           51   11    91   01C8   737          CMPB    #IO$_AVAILABLE, R1        ;Available function?
                               03    13   01CB   738          BEQL    AVAILABLE                 ;Branch if yes.
                              0082   31   01CD   739          BRW     FEXL                      ;Else, branch to execute function.
                                           01D0   740
                                           01D0   741 ;
                                           01D0   742 ;       UNLOAD and AVAILABLE Functions
                                           01D0   743 ;       Clear UCB$V_VALID in UCB$W_STS
                                           01D0   744 ;
                                           01D0   745 UNLOAD:
                                           01D0   746 AVAILABLE:
                     64 A5   0800 8F    AA   01D0   747          BICW    #UCB$M_VALID, -           ;Clear sofware volume valid bit.
                                           01D6   748                  UCB$W_STS(R5)
                                           01D6   749 ;       BRB     NORMAL                    ;Then complete the operation.
                                           01D6   750
                                           01D6   751
                                           01D6   752 ;
                                           01D6   753 ;       OPERATON COMPLETION
                                           01D6   754 ;
                                           01D6   755
                                           01D6   756 NORMAL:                                   ;SUCCESSFUL OPERATION COMPLETE
                           50   01    3C   01D6   757          MOVZWL  #SS$_NORMAL,R0            ;ASSUME NORMAL COMPLETION STATUS
                     0092 C5    0C    91   01D9   758          CMPB    #IO$_READPBLK,UCB$B_FEX(R5)  ;READ FUNCTION?
                               3F    12   01DE   759          BNEQ    FUNCXT                    ;IF NEQ - NO
                               06    E1   01E0   760          BBC     #RY_DB_V_DELD,-           ;IF CLR - NO DELETED DATA MARK
                            39 00D0 C5         01E2   761                  UCB$W_DY_DB(R5),FUNCXT    ;...
                     50     0661 8F    3C   01E6   762          MOVZWL  #SS$_RDDELDATA,R0         ;SET READ DELETED DATA STATUS
                               32    11   01EB   763          BRB     FUNCXT                    ;FUNCTION EXIT
                                           01ED   764
                                           01ED   765 RETRYERR:                                 ;RETRIABLE ERROR
                           0080 C5    97   01ED   766          DECB    UCB$B_ERTCNT(R5)          ;ANY RETRIES LEFT?
                               10    13   01F1   767          BEQL    FATALERR                  ;IF EQL - NO
                     0080 C5    01    91   01F3   768          CMPB    #1,UCB$B_ERTCNT(R5)       ; See if only one more retry left.
                               03    12   01F8   769          BNEQ    10$                       ; If NOT, branch around.
                             FF0E    30   01FA   770          BSBW    RX211_REINIT              ; If YES, re-INITIALIZE RX211.
                                           01FD   771 10$:
                       53    58 A5    D0   01FD   772          MOVL    UCB$L_IRP(R5),R3          ; Refresh R3 => IRP.
                               4F    11   0201   773          BRB     FEXL                      ;RETRY FUNCTION
                                           0203   774
                                           0203   775 FATALERR:                                 ;UNRECOVERABLE ERROR
```

```
                     50   01F4 8F    3C   0203   776          MOVZWL  #SS$_PARITY,R0           ;ASSUME PARITY ERROR STATUS
                          00   E0    0208   777          BBS     #RY_DB_V_CRC,-              ;IF SET - CRC ERROR
                        11 00D0 C5          020A   778                  UCB$W_DY_DB(R5),FUNCXT ;...
                     50   008C 8F    3C   020E   779          MOVZWL  #SS$_DRVERR,R0          ;ASSUME DRIVE ERROR STATUS
                          18   B3    0213   780          BITW    #RY_DB_M_DE!RY_DB_M_ACLO,-  ;DENSITY OR PWR ERROR?
                          00D0 C5          0215   781                  UCB$W_DY_DB(R5)        ;...
                          05   12    0218   782          BNEQ    FUNCXT                      ;IF NEQ - YES
                     50   0054 8F    3C   021A   783          MOVZWL  #SS$_CTRLERR,R0         ;SET CONTROLLER ERROR STATUS
                                          021F   784
                                          021F   785 FUNCXT:                               ;FUNCTION EXIT
                                50   DD   021F   786          PUSHL   R0                      ;SAVE FINAL REQUEST STATUS
                      00000000'GF   16   0221   787          JSB     G^IOC$DIAGBUFILL        ;FILL DIAGNOSTIC BUFFER IF PRESENT
                           00D2 C5   B5   0227   788          TSTW    UCB$W_DY_DPN(R5)        ;ARE UBA RESOURCES ALLOCATED?
                                14   13   022B   789          BEQL    10$                     ;IF EQL - NO
                           00C0 C5   A1   022D   790          ADDW3   UCB$W_BCR(R5),-         ;CALCULATE BYTES TRANSFERRED
                     02 AE      32   A3         0231   791                  IRP$W_BCNT(R3),2(SP) ;AND PUT IN I/O STATUS BLOCK
                                          0235   792          RELDPR                          ;RELEASE DATA PATH
                                          023B   793          RELMPR                          ;RELEASE MAP REGISTERS
                                          0241   794 10$:     RELCHAN                         ;RELEASE CHANNEL IF OWNED
                                51   D4   0247   795          CLRL    R1                      ;CLEAR 2ND LONGWORD OF IOSB
                                50 8ED0   0249   796          POPL    R0                      ;GET 1ST LONGWORD OF IOSB
                                          024C   797          REQCOM                          ;COMPLETE REQUEST
```

```
                     0252   799 ; FEXL - RX211 HARDWARE FUNCTION EXECUTION
                     0252   800 ;
                     0252   801 ; THIS ROUTINE IS CALLED VIA A BRB FROM STARTIO. PARAMETERS ARE LOADED
                     0252   802 ; INTO DEVICE REGISTERS AND THE FUNCTION IS INITIATED. THE RETURN ADDRESS
                     0252   803 ; IS STORED IN THE UCB AND A WAITFOR INTERRUPT IS EXECUTED. WHEN THE
                     0252   804 ; INTERRUPT OCCURS, CONTROL IS RETURNED TO THE CALLER.
                     0252   805 ;
                     0252   806 ; INPUTS:
                     0252   807 ;       R3 = IRP ADDRESS (I/O REQUEST PACKET)
                     0252   808 ;       R5 = UCB ADDRESS (UNIT CONTROL BLOCK)
                     0252   809 ;       00(SP) = RETURN ADDRESS OF CALLER
                     0252   810 ;
                     0252   811 ; OUTPUTS:
                     0252   812 ;       THERE ARE FOUR EXITS FROM THIS ROUTINE:
                     0252   813 ;
                     0252   814 ;       1. SPECIAL CONDITION - THIS EXIT IS TAKEN IF A POWER FAILURE OCCURS
                     0252   815 ;             OR THE OPERATION TIMES OUT.
                     0252   816 ;
                     0252   817 ;       2. FATAL ERROR - THIS EXIT IS TAKEN IF A FATAL CONTROLLER OR DRIVE
                     0252   818 ;             ERROR OCCURS OR IF ANY ERROR OCCURS AND ERROR RETRY IS EITHER
                     0252   819 ;             INHIBITED OR EXHAUSTED.
                     0252   820 ;
                     0252   821 ;       3. RETRIABLE ERROR - THIS EXIT IS TAKEN IF A RETRIABLE CONTROLLER
                     0252   822 ;             OR DRIVE ERROR OCCURS AND ERROR RETRY IS NEITHER INHIBITED
                     0252   823 ;             NOR EXHAUSTED.
                     0252   824 ;
                     0252   825 ;       4. SUCCESSFUL OPERATION - THIS EXIT IS TAKEN IF NO ERRORS OCCUR
                     0252   826 ;             DURING THE OPERATION.
                     0252   827 ;
                     0252   828 ;       IN ALL CASES IF AN ERROR OCCURS, AN ATTEMPT IS MADE TO LOG THE ERROR.
                     0252   829 ;       IN ALL CASES FINAL DEVICE REGISTERS ARE RETURNED VIA THE UCB.
                     0252   830 ;       UCB$W_BCR(R5) = NEGATIVE BYTES REMAINING TO TRANSFER
                     0252   831
                     0252   832 FEXL:                                          ;FUNCTION EXECUTOR
  50   24 A5   D0    0252   833         MOVL    UCB$L_CRB(R5),R0               ;GET ADDRESS OF PRIMARY CRB
  51   2C A0   D0    0256   834         MOVL    CRB$L_INTD+VEC$L_IDB(R0),R1    ;GET ADDRESS OF IDB
  04 A1     55   D1    025A   835         CMPL    R5,IDB$L_OWNER(R1)             ;DOES THIS PROCESS OWN CHANNEL?
            05   12    025E   836         BNEQ    10$                            ;IF NEQ - NO
     54     61   D0    0260   837         MOVL    IDB$L_CSR(R1),R4               ;SET ASSIGNED CHANNEL CSR ADDRESS
            06   11    0263   838         BRB     20$                            ;
                       0265   839 10$:     REQPCHAN                       ;REQUEST CHANNEL (RETURNS R4 = CSR ADR)
                       026B   840
64   0800 8F   B3    026B   841 20$:     BITW    #RY_CS_M_RX02,RY_CS(R4) ;IS DEVICE RX02?
            13   12    0270   842         BNEQ    30$                            ;IF NEQ - YES
00E0 C5     02   88    0272   843         BISB    #RY_RX01SW,UCB$B_DY_ER(R5)  ;SET ERROR BIT IN UCB
 00000000'GF     16    0277   844         JSB     G^ERL$DEVICERR                 ;ALLOCATE AND FILL ERROR MESSAGE BUFFER
50   0054 8F   3C    027D   845         MOVZWL  #SS$_CTRLERR,R0                ;SET CONTROLLER ERROR (RX01 SWITCH SET)
         0551   31    0282   846         BRW     RESETXFR                       ;EXIT
                       0285   847
0092 C5     1E   91    0285   848 30$:     CMPB    #IO$_FORMAT,UCB$B_FEX(R5)  ;FORMAT FUNCTION?
            0D   13    028A   849         BEQL    FORMAT                         ;IF EQL - YES
0092 C5     08   91    028C   850         CMPB    #IO$_PACKACK,UCB$B_FEX(R5)  ;PACK ACKNOWLEDGE FUNCTION?
            03   13    0291   851         BEQL    40$                            ;IF EQL - YES
         015E   31    0293   852         BRW     XFER                           ;MUST BE A TRANSFER FUNCTION
         00BB   31    0296   853 40$:     BRW     PACKACK                        ;PACK ACKNOWLEDGE FUNCTION
```

```
                     0299   855 ;
                     0299   856 ; FORMAT FUNCTION EXECUTION (SET MEDIA DENSITY)
                     0299   857 ;
                     0299   858 ; FUNCTIONAL DESCRIPTION:
                     0299   859 ;
                     0299   860 ; THIS FUNCTION CAUSES THE ENTIRE DISKETTE TO BE REASSIGNED TO A NEW
                     0299   861 ; DENSITY.  THIS OPERATION TAKES ABOUT 15 SECONDS TO COMPLETE.
                     0299   862 ;
                     0299   863 ; IT IS ASSUMED THAT AN IO$_PACKACK HAS ALREADY BEEN PERFORMED ON THIS
                     0299   864 ; DISKETTE TO SET UP UCB$B_TRACKS.
                     0299   865 ;
                     0299   866 ; THE DRIVER EXITS WITH SS$_CTRLERR STATUS IF SINGLE DENSITY FORMAT
                     0299   867 ; IS REQUESTED FOR A DOUBLE SIDED DISKETTE.
                     0299   868 ;
                     0299   869 ; The Driver exits with SS$_FORMAT status if an attempt is made to reformat
                     0299   870 ;       a quad density diskette.  The diskette is not modified.
                     0299   871 ;
                     0299   872 ; INPUTS:
                     0299   873 ;       R3      - IRP ADDRESS
                     0299   874 ;       R4      - CSR ADDRESS
                     0299   875 ;       R5      - UCB ADDRESS
                     0299   876 ;
                     0299   877
                     0299   878 FORMAT:                                      ;REFORMAT DISK TO NEW DENSITY
                     0299   879
                     0299   880 ;
                     0299   881 ; SET NEW DENSITY (VIA MAXBLOCK) IN UCB
                     0299   882 ;
                     0299   883
       38 A3    D0   0299   884         MOVL    IRP$L_MEDIA(R3),-       ;SET PARAMETER LONGWORD IN UCB
     00BC C5         029C   885                 UCB$L_MEDIA(R5)         ;...
    02   45 A5   91  029F   886         CMPB    UCB$B_TRACKS(R5),#2     ;IS IT DOUBLE SIDED?
          16 19  02A3   887         BLSS    10$                     ;IF LSS - NO
00B0 C5  07C5 8F  3C  02A5   888         MOVZWL  #RY_DSDD,UCB$L_MAXBLOCK(R5)  ;SET DOUBLE SIDED MAXBLOCKS
   02    00BC C5  D1  02AC   889         CMPL    UCB$L_MEDIA(R5),#2      ;IS DOUBLE DENSITY REQUESTED?
          1D 13  02B1   890         BEQL    20$                     ;IF EQL - YES
   50    0054 8F  3C  02B3   891         MOVZWL  #SS$_CTRLERR,R0         ;SET ERROR STATUS
            FF64  31  02B8   892         BRW     FUNCXT                  ;AND EXIT
                  02BB   893
00B0 C5  01EE 8F  3C  02BB   894 10$:    MOVZWL  #RY_SSSD,UCB$L_MAXBLOCK(R5)  ;ASSUME SINGLE DENSITY
   02    00BC C5  D1  02C2   895         CMPL    UCB$L_MEDIA(R5),#2      ;IS DOUBLE DENSITY REQUESTED?
          07 19  02C7   896         BLSS    20$                     ;IF LSS - NO, SINGLE DENSITY
00B0 C5  03DC 8F  3C  02C9   897         MOVZWL  #RY_SSDD,UCB$L_MAXBLOCK(R5)  ;SET DOUBLE DENSITY MAXBLOCKS
                  02D0   898
                  02D0   899 ;
                  02D0   900 ; REFORMAT DISKETTE
                  02D0   901 ;
                  02D0   902
            FE6F  30  02D0   903 20$:    BSBW    DY_MERGE                ;MERGE GO,UNIT,IE,DEN IN R2
   64    08   52  A9  02D3   904         BISW3   R2,#F_SETDEN,RY_CS(R4)  ;INITIATE SET DENSITY FUNCTION
         7E   50  7D  02D7   905         MOVQ    R0,-(SP)                ;SAVE R0-R1
                  02DA   906         TIMEDWAIT TIME=#100*1000,-      ;ONE SECOND WAIT TIMEOUT
                  02DA   907                 INS1=<BITB      #RY_CS_M_TR!RY_CS_M_DONE,RY_CS(R4)>,- ;T/R OR DONE?
                  02DA   908                 INS2=<BNEQ      25$>,-  ;IF LSS - TRANSFER COMPLETE (T/R)
                  02DA   909                                 -       ;IF NON-ZERO - DONE BIT SET - ERROR
                  02DA   910                                 -       ;IF EQL - NEITHER, WAIT
                  02DA   911                 DONELBL=25$
```

```
                50    8E   7D  0302    912          MOVQ    (SP)+,R0                        ;RESTORE R0-R1
              64    A0 8F   93  0305    913          BITB    #RY_CS_M_TR!RY_CS_M_DONE,RY_CS(R4) ;T/R OR DONE?
                     05   19  0309    914          BLSS    26$                             ;IF LSS - TRANSFER COMPLETE (T/R)
                     03   13  030B    915          BEQL    26$                             ;IF EQL - TIME HAS EXPIRED
                   0416   31  030D    916          BRW     RETREG                          ;DONE BIT SET - ERROR
                          0310    917 26$:                                                ;NORMAL RETURN
                          0310    918          CKPWR                                      ;DSBINT & CHECK FOR PWR FAILURE
          02 A4    0049 8F   B0  0321    919          MOVW    #^X49,RY_DB(R4)                 ;PUT ASCII "I" IN DBR TO START FNTN
                          0327    920          WFIKPCH SPECOND,#25                        ;WAITFOR INTERRUPT
                          0331    921          IOFORK                                     ;CREATE FORK PROCESS (&JSB BACK TO ISR)
                          0337    922
                     0F   E1  0337    923          BBC     #RY_CS_V_ERR,-                  ; If no error at all, branch around.
                  14 00CE C5          0339    924                  UCB$W_DY_CS(R5),30$
                     04   E1  033D    925          BBC     #RY_DB_V_DE,-                   ; If no DENSITY error, branch around.
                  0E 00D0 C5          033F    926                  UCB$W_DY_DB(R5),30$
                     01   E1  0343    927          BBC     #RY_DB_V_QDEN,-                 ; If NOT quad density, branch around.
                  08 00D0 C5          0345    928                  UCB$W_DY_DB(R5),30$
                50    00BC 8F   3C  0349    929          MOVZWL  #SS$_FORMAT,R0                  ; If we tried to change QUAD diskette.
                       FECE   31  034E    930          BRW     FUNCXT                          ;  Return error and branch.
                          0351    931 30$:
                   03D2   31  0351    932          BRW     RETREG                          ;
```

```
                     0354    934 ;
                     0354    935 ; PACK ACKNOWLEDGE FUNCTION EXECUTION
                     0354    936 ;
                     0354    937 ; INPUTS:
                     0354    938 ;
                     0354    939 ;       R4      - CSR ADDRESS
                     0354    940 ;       R5      - UCB ADDRESS
                     0354    941 ;
                     0354    942 ; FUNCTIONAL DESRIPTION:
                     0354    943 ;
                     0354    944 ;       THIS OPERATION ESTABLISHES THE CURRENT DISKETTE'S DENSITY AND
                     0354    945 ;       NUMBER OF SIDES.  THIS INFORMATION IS THEN STORED IN THE UCB.
                     0354    946 ;
                     0354    947 ;       IO$PACKACK MUST BE THE FIRST FUNCTION ISSUED TO A DISKETTE AFTER
                     0354    948 ;       IT HAS BEEN PLACED IN A DRIVE.
                     0354    949 ;
                     0354    950 ; OUTPUTS:
                     0354    951 ;
                     0354    952 ;       UCB$L_MAXBLOCK, UCB$B_TRACKS, UCB$B_SECTORS, UCB$W_CYLINDERS,
                     0354    953 ;       UCB$B_DEVTYPE, AND UCB$B_DEVCLASS ARE UPDATED.  UCB$V_VALID IS
                     0354    954 ;       SET IN UCB$W_STS.
                     0354    955 ;
                     0354    956
                     0354    957 PACKACK:                                ; PACK ACKNOWLEDGE
0800 8F    A8        0354    958         BISW    #UCB$M_VALID,-
  64 A5              0358    959          UCB$W_STS(R5)                 ; Set software volume valid bit.
     1A    90        035A    960         MOVB    #RY_SECTORS,-
  44 A5              035C    961          UCB$B_SECTORS(R5)             ; Set sectors/track
  4D 8F    9B        035E    962         MOVZBW  #RY_CYLINDERS,-
  46 A5              0361    963          UCB$W_CYLINDERS(R5)           ; Set # cylinders
     01    90        0363    964         MOVB    #DC$_DISK,-
  40 A5              0365    965          UCB$B_DEVCLASS(R5)            ; Set disk device class
     0B    90        0367    966         MOVB    #DT$_RX02,-
  41 A5              0369    967          UCB$B_DEVTYPE(R5)             ; Assume RX02 device type
45 A5    01    90    036B    968         MOVB    #1,UCB$B_TRACKS(R5)           ; Assume single sided
26658002 8F    D0    036F    969         MOVL    #^X26658002,-
    008C C5          0375    970          UCB$L_MEDIA_ID(R5)            ; Set media ident "DY RX02"
    01EE 8F    3C    0378    971         MOVZWL  #RY_SSSD,-
    00B0 C5          037C    972          UCB$L_MAXBLOCK(R5)            ; Assume single density
                     037F    973
       FDC0    30    037F    974         BSBW    DY_MERGE                ;MERGE GO,UNIT,DEN,IE IN R2
                     0382    975         CKPWR                           ;DSBINT & CHECK FOR PWR FAILURE
64   0A    52    A9 0393    976         BISW3   R2,#F_READSTATUS,RY_CS(R4)  ;EXECUTE READ STATUS FUNCTION
                     0397    977         WFIKPCH SPECOND,#10             ; Wait for interrupt.
                     03A1    978         IOFORK                          ;CREATE FORK PROCESS (& JSB BACK TO ISR)
                     03A7    979
00D0 C5   0080 8F    B3 03A7    980         BITW    #RY_DB_M_DRDY,UCB$W_DY_DB(R5)  ;WAS DRIVE READY?
          08    12    03AE    981         BNEQ    10$                     ;IF NEQ - YES
   50   01A4 8F    3C 03B0    982         MOVZWL  #SS$_MEDOFL,R0          ;SET MEDIUM OFFLINE STATUS
        FE67    31    03B5    983         BRW     FUNCXT                  ;AND EXIT
                     03B8    984
          04    E5    03B8    985 10$:    BBCC    #RY_DB_V_DE,-           ; If clear, Single density so
  14 00D0 C5         03BA    986          UCB$W_DY_DB(R5),15$           ;  branch around.
    07B8 8F    3C    03BE    987         MOVZWL  #RY_SSQD,-              ; If NOT single, setup for QUAD and
    00B0 C5          03C2    988          UCB$L_MAXBLOCK(R5)            ;  then we will test to see if so.
          01    E0    03C5    989         BBS     #RY_DB_V_QDEN,-         ; If set, then it IS quad density so
07 00D0 C5         03C7    990          UCB$W_DY_DB(R5),15$           ;  we branch around next instruction.
```

```
                00B0 C5   03DC 8F    3C   03CB   991
                                                03D2   992 15$:
                                      B3   03D2   993
                                                03D3   994
                                                03D3   995
                                                03D3   996
                00D0 C5   0C09 8F                03D3   997
                               07     12   03D9   998
                          8000 8F     AA   03DB   999
                          00CE C5                03DF  1000
                                                03E2  1001
                               09     E1   03E2  1002 20$:
                     09 00D0 C5                03E4  1003
                               0C     90   03E8  1004
                            41 A5                03EA  1005
                        008C C5    02     C0   03EC  1006
                                 0332     31   03F1  1007 30$:

MOVZWL  #RY_SSDD,UCB$L_MAXBLOCK(R5)  ;SET DOUBLE DENSITY IN UCB
BITW    #RY_DB_M_CRC!-                ;ANY ERRORS BESIDES DENSITY ERROR?
        RY_DB_M_ACLO!-                ;...
        RY_DB_M_WCO!-                 ;...
        RY_DB_M_NXM,-                 ;...
        UCB$W_DY_DB(R5)               ;...
BNEQ    20$                           ;IF NEQ - YES
BICW    #RY_CS_M_ERR,-
         UCB$W_DY_CS(R5)              ; No, clear csr error bit
                                      ;
BBC     #RY_DB_V_RX04,-               ; See if controller is RX04.
        UCB$W_DY_DB(R5),30$           ;  and if NOT branch around.
MOVB    #DT$_RX04,-
         UCB$B_DEVTYPE(R5)            ; Set proper device type.
ADDL    #2,UCB$L_MEDIA_ID(R5)         ; Set media ident "DY RX04"
BRW     RETREG                        ;
```

```
                     03F4  1009 ;
                     03F4  1010 ; TRANSFER FUNCTION EXECUTION
                     03F4  1011 ;
                     03F4  1012 ;       FUNCTIONS INCLUDE:
                     03F4  1013 ;
                     03F4  1014 ;               WRITE DATA, AND
                     03F4  1015 ;               READ DATA
                     03F4  1016 ;
                     03F4  1017 ; INPUTS:
                     03F4  1018 ;
                     03F4  1019 ;       R3      - IRP ADDRESS
                     03F4  1020 ;       R4      - DEVICE CSR ADDRESS
                     03F4  1021 ;       R5      - UCB ADDRESS
                     03F4  1022 ;
                     03F4  1023 ; FUNCTIONAL DESCRIPTION:
                     03F4  1024 ;
                     03F4  1025 ; THE LBN IS CONVERTED TO CYLINDER, TRACK, AND SECTOR, THEN SKEW AND
                     03F4  1026 ; INTERLEAVE FACTORS ARE CALCULATED TO ARRIVE AT A PHYSICAL MEDIA ADDRESS.
                     03F4  1027 ;
                     03F4  1028 ; A UNIBUS DATAPATH IS REQUESTED FOLLOWED BY THE APPROPRIATE NUMBER OF MAP
                     03F4  1029 ; REGISTERS REQUIRED FOR THE TRANSFER.
                     03F4  1030 ;
                     03F4  1031 ; SINCE THE RX211 ALLOWS A MAXIMUM DATA TRANSFER OF ONE SECTOR, SINGLE
                     03F4  1032 ; SECTOR TRANSFERS ARE REPEATED (VIA THE "COMXFER:" LOOP) UNTIL THE I/O
                     03F4  1033 ; REQUEST IS COMPLETE.
                     03F4  1034 ;
                     03F4  1035 ; EACH SECTOR TRANSFER IS ACCOMPLISHED BY A SEQUENCE OF TWO FUNCTION CODES:
                     03F4  1036 ;       F_FILLBUFFER AND F_WRITESECTOR FOR A WRITE FUNCTION, OR
                     03F4  1037 ;       F_READSECTOR AND F_EMPTYBUFFER FOR A READ FUNCTION.
                     03F4  1038 ; THE CSR BITS FOR THE FIRST FUNCTION IN THE SEQUENCE ARE LOADED INTO THE
                     03F4  1039 ; LOWER WORD OF UCB$L_DY_XFER; THOSE FOR THE SECOND FUNCTION ARE PUT IN
                     03F4  1040 ; THE UPPER WORD.  AFTER EXECUTING EACH FUNCTION, UCB$L_DY_XFER IS ROTATED
                     03F4  1041 ; SO THAT THE LOWER WORD ALWAYS CONTAINS THE CSR BITS FOR THE NEXT FUNCTION.
                     03F4  1042 ;
                     03F4  1043 ; A PROTOCALL OF LOADING THE RX211 DATA BUFFER REGISTER (DBR) WITH TWO UCB
                     03F4  1044 ; FIELDS IS REQUIRED AFTER LOADING THE CSR.  R3 IS LOADED AND ROTATED SO
                     03F4  1045 ; THAT ITS LOWER WORD ALWAYS CONTAINS THE FIRST UCB OFFSET TO BE LOADED
                     03F4  1046 ; INTO THE DBR FOR THE CURRENT FUNCTION CODE.
                     03F4  1047 ;
                     03F4  1048 ; THE CHANNEL AND UBA RESOURCES ARE NOT RELEASED UNTIL THE ENTIRE I/O
                     03F4  1049 ; REQUEST IS COMPLETE.
                     03F4  1050 ;
                     03F4  1051 ; IT IS ASSUMED THAT AN IO$_PACKACK FUNCTION HAS ALREADY BEEN PERFORMED
                     03F4  1052 ; ON THIS DISKETTE TO SET UP UCB$B_TRACKS AND UCB$L_MAXBLOCK.
                     03F4  1053 ;
                     03F4  1054
                     03F4  1055 XFER:                                        ;TRANSFER FUNCTION EXECUTION
                     03F4  1056
00D2 C5   B5         03F4  1057          TSTW    UCB$W_DY_DPN(R5)            ;IS THIS A RETRY?
     03   13         03F8  1058          BEQL    2$                          ;IF EQL - NO
00C6   31            03FA  1059          BRW     15$                         ;DATAPATH ALREADY OWNED
                     03FD  1060
                     03FD  1061 ;
                     03FD  1062 ; FIRST TRANSFER OF THIS I/O REQUEST
                     03FD  1063 ;
                     03FD  1064
                     03FD  1065 ;
```

```
                                03FD  1066 ; DETERMINE SECTOR SIZE
                                03FD  1067 ;
                                03FD  1068
00CC C5   0040 8F   B0          03FD  1069 2$:      MOVW    #RY_SWPS,UCB$W_DY_WPS(R5)       ; Assume single dens. WORDS/SECTOR
01EE 8F   00B0 C5   B1          0404  1070          CMPW    UCB$L_MAXBLOCK(R5),#RY_SSSD  ;SINGLE DENSITY?
                      17  15    040B  1071          BLEQ    5$                          ;IF LEQ - YES
00CC C5   0080 8F   B0          040D  1072          MOVW    #RY_DWPS,UCB$W_DY_WPS(R5)       ; Assume double dens. WORDS/SECTOR
03DC 8F   00B0 C5   B1          0414  1073          CMPW    UCB$L_MAXBLOCK(R5),#RY_SSDD     ; Double density?
                      07  15    041B  1074          BLEQ    5$                          ; If LEQ - yes.
00CC C5   0100 8F   B0          041D  1075          MOVW    #RY_QWPS,UCB$W_DY_WPS(R5)       ; Adjust for QUAD density.
                                0424  1076
                                0424  1077 ;
                                0424  1078 ; CONVERT LOGICAL BLOCK NUMBER TO CYLINDER, TRACK, AND SECTOR
                                0424  1079 ;
                                0424  1080 ;       LBN = LBN * (SECTORS/BLOCK)
                                0424  1081 ;       LBN/(SECTORS/TRACK) = D + SECTOR
                                0424  1082 ;       D/(TRACKS/CYLINDER) = CYLINDER + TRACK
                                0424  1083 ;
                                0424  1084
  00EC C5   38 A3   D0          0424  1085 5$:      MOVL    IRP$L_MEDIA(R3),UCB$L_DY_LMEDIA(R5)  ;ASSUME PHYSICAL I/O
                08  E0          042A  1086          BBS     #IRP$V_PHYSIO,-                 ;IF SET - PHYSICAL I/O
                2F 2A A3        042C  1087                  IRP$W_STS(R3),10$               ;...
        50  00CC C5   3C         042F  1088          MOVZWL  UCB$W_DY_WPS(R5),R0     ; Get words per sector.
50  0100 8F   50   A7           0434  1089          DIVW3   R0,#256,R0              ;FORM SECTORS/BLOCK IN R0
          50   38 A3   C4       043A  1090          MULL    IRP$L_MEDIA(R3),R0      ;SCALE LBN IN R0
          52   44 A5   9A       043E  1091          MOVZBL  UCB$B_SECTORS(R5),R2    ;PUT SECTORS/TRACK IN R2
               51   D4          0442  1092          CLRL    R1                      ;CLEAR HIGH PART OF DIVIDEND
00EC C5   50   50   52   7B      0444  1093          EDIV    R2,R0,R0,UCB$L_DY_LMEDIA(R5)  ;CALCULATE SECTOR NUMBER AND STORE
          52   45 A5   9A       044B  1094          MOVZBL  UCB$B_TRACKS(R5),R2     ;PUT TRACKS/CYLINDER IN R2
51  50   50   52   7B           044F  1095          EDIV    R2,R0,R0,R1             ;CALCULATE TRACK AND CYLINDER
     00ED C5   51   90          0454  1096          MOVB    R1,UCB$L_DY_LMEDIA+1(R5)  ;STORE TRACK NUMBER
     00EE C5   50   B0          0459  1097          MOVW    R0,UCB$L_DY_LMEDIA+2(R5)  ;STORE CYLINDER NUMBER
                                045E  1098
                                045E  1099 ;
                                045E  1100 ; Output of above code is to produce the logical sector number in UCB$L_DY_LMEDIA
                                045E  1101 ;       in the following format:
                                045E  1102 ;
                                045E  1103 ;                   31              16 15    8 7      0
                                045E  1104 ;                   .....................................
                                045E  1105 ;                   :                   :       :       :
                                045E  1106 ;                   :   cylinder      : track : sector:
                                045E  1107 ;                   :       #         :   #   :   #   :
                                045E  1108 ;                   :                 :(always:  see  :
                                045E  1109 ;                   :    0 to 76      : zero) : below :
                                045E  1110 ;                   .....................................
                                045E  1111 ;
                                045E  1112 ;                   Sector number ranges:
                                045E  1113 ;                     Physical I/O  1 to 26
                                045E  1114 ;                     Logical I/O   0 to 25
                                045E  1115
  51   44 A5   9A               045E  1116 10$:     MOVZBL  UCB$B_SECTORS(R5), R1   ; Get maximum sectors information.
00EC C5   51   B1               0462  1117          CMPW    R1, UCB$L_DY_LMEDIA(R5) ; Maximum sector exceeded?
04 2A A3   08   E0              0467  1118          BBS     #IRP$V_PHYSIO, -        ; Seperate the logical from the
                                046C  1119                  IRP$W_STS(R3), 110$     ; physical I/O; branch if physical.
           12   1B              046C  1120          BLEQU   190$                    ; Branch if too big for logical I/O.
           08   11              046E  1121          BRB     130$                    ; All ok, so far, continue tests.
           0E   1F              0470  1122 110$:    BLSSU   190$                    ; Branch if physical sector too big.
```

```
                00EC C5   D5  0472  1123          TSTL    UCB$L_DY_LMEDIA(R5)     ; Zero physical sector is also illegal.
                     08   13  0476  1124          BEQL    190$                    ; Branch if zero physical sector.
          00EE C5    46 A5   B1  0478  1125 130$:  CMPW    UCB$W_CYLINDERS(R5), -  ; Check for, maximum cylinder
                                047E  1126                  UCB$L_DY_LMEDIA+2(R5)   ; exceeded.
                     08   1A  047E  1127          BGTRU   12$                     ; Branch if max. cylinder not exceeded.
             50   0134 8F   3C  0480  1128 190$:  MOVZWL  #SS$_IVADDR, R0         ; Otherwise, give invalid address
                       FD97   31  0485  1129          BRW     FUNCXT                  ; status and kill request.
                              0488  1130
                              0488  1131 ;
                              0488  1132 ; ALLOCATE UBA RESOURCES
                              0488  1133 ;
                              0488  1134
                              0488  1135 12$:   REQDPR                          ;REQUEST DATAPATH
                              048E  1136          REQMPR                          ;REQUEST MAP REGISTERS
                              0494  1137          LOADUBA                         ;LOAD UNIBUS MAP REGISTERS
               51    24 A5   D0  049A  1138          MOVL    UCB$L_CRB(R5),R1        ;GET CRB ADDRESS
                  05    00   EF  049E  1139          EXTZV   #VEC$V_DATAPATH,#VEC$S_DATAPATH,-  ;EXTRACT DATAPATH NUMBER -
                50    37 A1       04A1  1140                  CRB$L_INTD+VEC$B_DATAPATH(R1),R0   ;FOR UBA RESOURCE FLAG
                00D2 C5   50   B0  04A4  1141          MOVW    R0,UCB$W_DY_DPN(R5)     ;INDICATE UBA RESOURCES ALLOCATED
                50    7C A5   3C  04A9  1142          MOVZWL  UCB$W_BOFF(R5),R0       ;GET BYTE OFFSET IN PAGE
                      34 A1   F0  04AD  1143          INSV    CRB$L_INTD+VEC$W_MAPREG(R1),-  ;INSERT HIGH 7 BITS OF ADDRESS
             50    07      09  04B0  1144                  #9,#7,R0                ;...
                00E6 C5   50   B0  04B3  1145          MOVW    R0,UCB$W_DY_SBA(R5)     ;PUT BUFFER ADDRESS IN UCB
          50    34 A1    02   07  EF  04B8  1146          EXTZV   #7,#2,CRB$L_INTD+VEC$W_MAPREG(R1),R0  ;GET MEMORY EXTENSION BITS
                00E3 C5   50   90  04BE  1147          MOVB    R0,UCB$B_DY_XBA(R5)     ;AND SAVE THEM IN THE UCB
                              04C3  1148
                              04C3  1149 ;
                              04C3  1150 ; Output of above section of code is put the UNIBUS Virtual Address of the
                              04C3  1151 ;       transfer into UCB$W_DY_SBA and the two high order bits of this UNIBUS
                              04C3  1152 ;       Virtual Address into UCB$B_DY_XBA.
                              04C3  1153 ;
                              04C3  1154
                              04C3  1155 ;
                              04C3  1156 ; SET CSR BITS IN UCB$L_DY_XFER
                              04C3  1157 ; SET UCB OFFSETS IN R3 FOR USE AS POINTERS DURING DEVICE DBR PROTOCALL
                              04C3  1158 ;
                              04C3  1159
                       FC7C   31  04C3  1160 15$:   BSBW    DY_MERGE                ;SET GO,IE,UNIT,DEN BITS IN R2
                              04C6  1161
                              04C6  1162                  ;SET UCB OFFSETS IN R3
               53    E4 8F   9A  04C6  1163          MOVZBL  #UCB$W_DY_PWC,R3        ;ASSUME WC OFFSET AS POINTER TO UCB-
                              04CA  1164                  ;FIELDS FOR 2ND FUNCTION CODE
          53   53    10   78  04CA  1165          ASHL    #16,R3,R3               ;MAKE ROOM FOR SECTOR ADDRESS
               53    BC 8F   90  04CE  1166          MOVB    #UCB$L_MEDIA,R3         ;ASSUME DA OFFSET AS POINTER TO UCB-
                              04D2  1167                  ;FIELDS FOR 1ST FUNCTION CODE
                              04D2  1168
                0092 C5    0C   91  04D2  1169          CMPB    #IO$_READPBLK,UCB$B_FEX(R5)  ;READ FUNCTION?
                           0E   12  04D7  1170          BNEQ    20$                     ;IF NEQ - NO, MUST BE WRITE
                              04D9  1171
                              04D9  1172                  ;READ FUNCTION
          00E8 C5    52    06   A9  04D9  1173          BISW3   #F_READSECTOR,R2,-      ;SET READ SECTOR AS 1ST FUNCTION
                              04DF  1174                  UCB$L_DY_XFER(R5)       ;...
          00EA C5    52    02   A9  04DF  1175          BISW3   #F_EMPTYBUFFER,R2,-     ;SET EMPTY BUFFER AS 2ND FUNCTION
                              04E5  1176                  UCB$L_DY_XFER+2(R5)     ;...
                           20   11  04E5  1177          BRB     COMXFER                 ;
                              04E7  1178
                              04E7  1179 20$:                                     ;WRITE FUNCTION
```

```
                53   53   10   9C  04E7  1180          ROTL    #16,R3,R3                       ;SHIFT ORDER OF UCB OFFSETS FOR WRITE
            00E8 C5   52   00   A9  04EB  1181          BISW3   #F_FILLBUFFER,R2,-              ;SET FILL BUFFER AS 1ST FUNCTION
                                    04F1  1182                  UCB$L_DY_XFER(R5)               ;...
            00EA C5   52   04   A9  04F1  1183          BISW3   #F_WRITESECTOR,R2,-             ;ASSUME WRITE SECTOR AS 2ND FUNCTION
                                    04F7  1184                  UCB$L_DY_XFER+2(R5)             ;...
                           06   E1  04F7  1185          BBC     #IO$V_DELDATA,-                 ;IF CLR - NOT WRITE DELETED DATA
                     0A 009A C5        04F9  1186                  UCB$W_FUNC(R5),COMXFER          ;...
                        00EA C5   B4  04FD  1187          CLRW    UCB$L_DY_XFER+2(R5)             ;CLEAR 2ND FUNCTION FIELD
            00EA C5   52   0C   A9  0501  1188          BISW3   #F_WRITEDEL,R2,-                ;SET WRITE DELETED DATA AS 2ND FUNCTION
                                    0507  1189                  UCB$L_DY_XFER+2(R5)             ;...
                                    0507  1190
                                    0507  1191  ;
                                    0507  1192  ; COMMON TRANSFER POINT - LOOP POINT FOR INDIVIDUAL SECTOR TRANSFERS
                                    0507  1193  ;
                                    0507  1194  ; NOTE: CODE IN THIS LOOP IS IN-LINE AS MUCH AS POSSIBLE TO DECREASE
                                    0507  1195  ;       EXECUTION TIME IN ORDER THAT THE NEXT INTERLEAVED SECTOR ON THE
                                    0507  1196  ;       DISKETTE IS NOT MISSED (WHICH WOULD CAUSE A WAIT FOR AN ENTIRE
                                    0507  1197  ;       DISKETTE REVOLUTION).
                                    0507  1198  ;
                                    0507  1199  ; INPUTS TO LOOP:
                                    0507  1200  ;
                                    0507  1201  ;       R3      - UCB OFFSETS FOR DEVICE DBR PROTOCALL
                                    0507  1202  ;       R4      - DEVICE CSR ADDRESS
                                    0507  1203  ;       R5      - UCB ADDRESS
                                    0507  1204  ;       UCB$L_DY_XFER   - LOW WORD: FCODE,GO,IE,DENSITY,UNIT FOR 1ST FUNCITON
                                    0507  1205  ;                       - HIGH WORD: FCODE,GO,IE,DENSITY,UNIT FOR 2ND FUNCTION
                                    0507  1206  ;
                                    0507  1207
                                    0507  1208  COMXFER:                                       ;START TRANSFER LOOP
                                    0507  1209
                                    0507  1210  ;
                                    0507  1211  ; CALCULATE SKEW AND INTERLEAVE FACTORS
                                    0507  1212  ;
                                    0507  1213  ; IF THE PHYSICAL I/O FLAG IS SET, THE ADDRESS IN UCB$L_DY_LMEDIA
                                    0507  1214  ; IS MOVED TO UCB$L_MEDIA.
                                    0507  1215  ; IF LOGICAL I/O IS BEING PERFORMED, THE LOGICAL ADDRESS IN UCB$L_DY_LMEDIA
                                    0507  1216  ; IS CONVERTED TO A PHYSICAL DISK ADDRESS BY APPLYING INTERLEAVE AND SKEW
                                    0507  1217  ; FACTORS, AND THE FIRST TRACK (RESERVED FOR INDUSTRY COMPATIBLILITY)
                                    0507  1218  ; IS SKIPPED.  THE RESULT IS PLACED IN UCB$L_MEDIA.
                                    0507  1219  ;
                                    0507  1220
              51     58 A5   D0     0507  1221          MOVL    UCB$L_IRP(R5),R1                ;GET ADDRESS OF REQUEST PACKET
            52     00BC C5   9E     050B  1222          MOVAB   UCB$L_MEDIA(R5),R2              ;POINT TO PHYSICAL MEDIA ADDRESS
            62     00EC C5   D0     0510  1223          MOVL    UCB$L_DY_LMEDIA(R5),(R2)        ;COPY LOGICAL ADDRESS
                        08   E0     0515  1224          BBS     #IRP$V_PHYSIO,-                 ;IF SET - PHYSICAL I/O
                     40 2A A1        0517  1225                  IRP$W_STS(R1),10$               ;...
                        51   62   9A  051A  1226          MOVZBL  (R2),R1                         ;GET CURRENT LOGICAL SECTOR
            50          51   01   78  051D  1227          ASHL    #1,R1,R0                        ; 2* Current Logical Sector => R0 needed
                                    0521  1228                                                  ;  in case of QUAD density to compute
                                    0521  1229                                                  ;  interleave factor of four.
                        51   0C   91  0521  1230          CMPB    #12,R1                          ;SET C IF SECTOR .GT. 12
                        51   51   D8  0524  1231          ADWC    R1,R1                           ;DOUBLE SECTOR #, ADD INTERLEAVE FACTOR
                        0100 8F   B1  0527  1232          CMPW    #RY_QWPS,-                      ; See if this is a QUAD density diskette
                        00CC C5       052B  1233                  UCB$W_DY_WPS(R5)
                             03   12  052E  1234          BNEQ    5$                              ; If NOT, branch around.
                        51   50   C0  0530  1235          ADDL    R0,R1                           ; If QUAD, add in 2*Sector for interleave
                                    0533  1236                                                  ;  factor of 4.
```

```
                        0533  1237 5$:
          50   02 A2    9A  0533  1238          MOVZBL  2(R2),R0                    ;GET CYLINDER NUMBER
50   51    50   06    7A  0537  1239          EMUL    #6,R0,R1,R0                 ;COMPUTE SKEW (6 * CYL + SECTOR)
       7E    44 A5    9A  053C  1240          MOVZBL  UCB$B_SECTORS(R5),-(SP)     ;GET SECTORS/TRACK
51   50    50    8E    7B  0540  1241          EDIV    (SP)+,R0,R0,R1              ;MODULO SECTOR INTO SECTORS PER TRACK
                 51    D6  0545  1242          INCL    R1                          ;OFFSET SECTOR NUMBER BY ONE
              62   51    90  0547  1243          MOVB    R1,(R2)                     ;SAVE SECTOR NUMBER IN UCB
                        054A  1244
                01 A2    96  054A  1245          INCB    1(R2)                       ;INCREMENT PAST RESERVED TRACK
          45 A5 01 A2    91  054D  1246          CMPB    1(R2),UCB$B_TRACKS(R5)      ;STILL WITHIN DISK DIMENSIONS?
                 06    19  0552  1247          BLSS    10$                         ;IF LSS - YES
                01 A2    94  0554  1248          CLRB    1(R2)                       ;RESET TRACK ADDRESS
                02 A2    96  0557  1249          INCB    2(R2)                       ;INCREMENT CYLINDER ADDRESS
                        055A  1250
                        055A  1251 ;
                        055A  1252 ; CALCULATE WORD COUNT FOR THIS TRANSFER
                        055A  1253 ;
                        055A  1254
             00C0 C5    AE  055A  1255 10$:    MNEGW   UCB$W_BCR(R5),-             ;GET BYTES LEFT TO TRANSFER AND -
             00E4 C5        055E  1256                 UCB$W_DY_PWC(R5)            ;ASSUME ONLY ONE TRANSFER NEEDED
          00E4 C5    02    A6  0561  1257          DIVW    #2,UCB$W_DY_PWC(R5)         ;FORM WORDS LEFT TO TRANSFER
             00E4 C5    B1  0566  1258          CMPW    UCB$W_DY_PWC(R5),-
             00CC C5        056A  1259                 UCB$W_DY_WPS(R5)            ; Are additional transfers required?
                 07    1B  056D  1260          BLEQU   20$                         ;IF LEQU - NO
             00CC C5    B0  056F  1261          MOVW    UCB$W_DY_WPS(R5),-          ; Set word count for one sector.
             00E4 C5        0573  1262                 UCB$W_DY_PWC(R5)            ;...
                        0576  1263
             00E3 C5    F0  0576  1264 20$:    INSV    UCB$B_DY_XBA(R5),-          ;PUT EXTENDED BA IN 1ST FUNCTION<13:12>
00E8 C5      02    0C        057A  1265                 #12,#2,UCB$L_DY_XFER(R5)    ;...
             00E9 C5    90  057F  1266          MOVB    UCB$L_DY_XFER+1(R5),-       ;PUT XBA AND HS IN 2ND FUNCTION TOO
             00EB C5        0583  1267                 UCB$L_DY_XFER+3(R5)         ;...
                        0586  1268
                        0586  1269 ;
                        0586  1270 ; EXECUTE TRANSFER FUNCTION
                        0586  1271 ;
                        0586  1272 ; INPUTS:
                        0586  1273 ;
                        0586  1274 ;       UCB$L_DY_XFER   :       CSR2        :       CSR1        :
                        0586  1275 ;                       :...................:...................:
                        0586  1276 ;
                        0586  1277 ;               R3      :       DBR3        :       DBR1        :
                        0586  1278 ;                       :...................:...................:
                        0586  1279 ;
                        0586  1280 ;               CSRn = BITS FOR nTH LOAD OF DEVICE CSR
                        0586  1281 ;               DBRn = OFFSET IN UCB FOR nTH LOAD OF DEVICE DBR
                        0586  1282 ;
                        0586  1283 ; FUNCTIONAL DESCRIPTION:
                        0586  1284 ;       THE CSR IS LOADED WITH THE LOW WORD OF UCB$L_DY_XFER.
                        0586  1285 ;       THE DBR IS LOADED WITH THE UCB FIELD SPECIFIED BY THE UCB OFFSET
                        0586  1286 ;       IN THE LOW WORD OF R3.
                        0586  1287 ;       THE DBR IS THEN LOADED WITH THE NEXT SEQUENTIAL UCB FIELD.
                        0586  1288 ;       AFTER THE INTERRUPT, UCB$L_DY_XFER AND R3 ARE ROTATED, AND THE
                        0586  1289 ;       PROCESS IS REPEATED FOR FUNCTION 2.
                        0586  1290 ;
                        0586  1291
00E2 C5      02    90  0586  1292          MOVB    #2,UCB$B_DY_LCT(R5)         ;SET LOOP COUNTER
                        058B  1293 30$:
```

```
                     64 00E8 C5    B0  058B  1294          MOVW    UCB$L_DY_XFER(R5),RY_CS(R4)  ;PUT FUNCTION IN CSR
                        7E   50    7D  0590  1295          MOVQ    R0,-(SP)                     ;SAVE R0-R1
                                       0593  1296          TIMEDWAIT TIME=#100*1000,-           ;ONE SECOND WAIT TIMEOUT
                                       0593  1297                  INS1=<BITB      #RY_CS_M_TR!RY_CS_M_DONE,RY_CS(R4)>,- ;T/R OR DONE?
                                       0593  1298                  INS2=<BNEQ      32$>,-       ;IF LSS - TRANSFER COMPLETE (T/R)
                                       0593  1299                                   -           ;IF NON-ZERO - DONE BIT SET - ERROR
                                       0593  1300                                   -           ;IF EQL - NEITHER, WAIT
                                       0593  1301                  DONELBL=32$
                           50   8E    7D  05BB  1302          MOVQ    (SP)+,R0                     ;RESTORE R0-R1
                     64    A0 8F    93  05BE  1303          BITB    #RY_CS_M_TR!RY_CS_M_DONE,RY_CS(R4) ;T/R OR DONE?
                              05    19  05C2  1304          BLSS    33$                          ;IF LSS - TRANSFER COMPLETE (T/R)
                              03    13  05C4  1305          BEQL    33$                          ;IF EQL - TIME HAS EXPIRED
                                 015D   31  05C6  1306          BRW     RETREG                       ;DONE BIT SET - ERROR
                                       05C9  1307 33$:                                          ;NORMAL RETURN
                                       05C9  1308
                                       05C9  1309                                               ;LOAD WORD COUNT OR SECTOR ADR IN DBR
                           50   53    3C  05C9  1310          MOVZWL  R3,R0                        ;GET UCB OFFSET
                           50   55    C0  05CC  1311          ADDL    R5,R0                        ;CALCULATE UCB FIELD ADDRESS
                     02    A4 80    B0  05CF  1312          MOVW    (R0)+,RY_DB(R4)              ;PUT UCB FIELD IN DBR
                           7E   50    7D  05D3  1313          MOVQ    R0,-(SP)                     ;SAVE R0-R1
                                       05D6  1314          TIMEDWAIT TIME=#100*1000,-           ;ONE SECOND WAIT TIMEOUT
                                       05D6  1315                  INS1=<BITB      #RY_CS_M_TR!RY_CS_M_DONE,RY_CS(R4)>,- ;T/R OR DONE?
                                       05D6  1316                  INS2=<BNEQ      36$>,-       ;IF LSS - TRANSFER COMPLETE (T/R)
                                       05D6  1317                                   -           ;IF NON-ZERO - DONE BIT SET - ERROR
                                       05D6  1318                                   -           ;IF EQL - NEITHER, WAIT
                                       05D6  1319                  DONELBL=36$
                           50   8E    7D  05FE  1320          MOVQ    (SP)+,R0                     ;RESTORE R0-R1
                     64    A0 8F    93  0601  1321          BITB    #RY_CS_M_TR!RY_CS_M_DONE,RY_CS(R4) ;T/R OR DONE?
                              05    19  0605  1322          BLSS    37$                          ;IF LSS - TRANSFER COMPLETE (T/R)
                              03    13  0607  1323          BEQL    37$                          ;IF EQL - TIME HAS EXPIRED
                                 011A   31  0609  1324          BRW     RETREG                       ;DONE BIT SET - ERROR
                                       060C  1325 37$:                                          ;NORMAL RETURN
                                       060C  1326
                                       060C  1327                                               ;LOAD BUS ADR OR CYLINDER ADR IN DBR
                                       060C  1328          DSBINT
                  06 64 A5    05    E1  0612  1329          BBC     #UCB$V_POWER,UCB$W_STS(R5),35$  ;IF CLR - NO POWER FAILURE
                                       0617  1330          ENBINT                               ;ENABLE INTERRUPTS
                                 01C6   31  061A  1331          BRW     PWRFAIL                      ;HANDLE POWER FAILURE
                        02 A4    60    B0  061D  1332 35$:     MOVW    (R0),RY_DB(R4)               ;PUT NEXT UCB FIELD IN DBR
                                       0621  1333          WFIKPCH SPECOND,#2                   ;WAIT FOR INTERRUPT
                                       062B  1334          IOFORK                               ;CREATE FORK PROCESS (&JSB BACK TO ISR)
                     53    53    10    9C  0631  1335          ROTL    #16,R3,R3                    ;SETUP UCB FIELDS FOR NEXT FUNCTION
                     00E8  C5    10    9C  0635  1336          ROTL    #16,UCB$L_DY_XFER(R5),-      ;SET UP NEXT FUNCTION
                           00E8  C5        063A  1337                  UCB$L_DY_XFER(R5)            ;...
                                       063D  1338
               03 00CE  C5    0F    E1  063D  1339          BBC     #RY_CS_V_ERR,UCB$W_DY_CS(R5),40$  ;IF CLR - NO ERRORS
                                 0088   31  0643  1340          BRW     DY_PURGE                     ; Error - Goto Purge datapath
                                       0646  1341
                           00E2  C5    97  0646  1342 40$:     DECB    UCB$B_DY_LCT(R5)             ;DECREMENT LOOP COUNTER
                                 03    15  064A  1343          BLEQ    45$                          ;IF LEQ - DONE, DON'T LOOP AGAIN
                                    FF3C   31  064C  1344          BRW     30$                          ;LOOP FOR 2ND FUNCTION
                                       064F  1345
                                       064F  1346 ;
                                       064F  1347 ; UPDATE BUFFER ADDRESS, DISK ADDRESS, AND BYTES REMAINING FOR NEXT SECTOR
                                       064F  1348 ;
                                       064F  1349
                                       064F  1350                                               ;UPDATE BYTES REMAINING TO TRANSFER
```

```
                50   00E4 C5   3C   064F  1351 45$:    MOVZWL  UCB$W_DY_PWC(R5),R0     ;GET WORDS TRANSFERRED
                     50   02   C4   0654  1352         MULL    #2,R0                   ;FORM BYTES TRANSFERRED
                00C0 C5   50   A0   0657  1353         ADDW    R0,UCB$W_BCR(R5)        ;UPDATE NEG BYTES REMAINING TO TRANSFER
                                    065C  1354
                                    065C  1355                                         ;UPDATE BUFFER ADDRESS
                51   00E6 C5   3C   065C  1356         MOVZWL  UCB$W_DY_SBA(R5),R1     ;GET ORIGINAL BUFFER ADDRESS IN R1
          51    02   10   00E3 C5   F0   0661  1357         INSV    UCB$B_DY_XBA(R5),#16,#2,R1  ;INSERT EXTENDED BITS
                     51   50   C0   0668  1358         ADDL    R0,R1                   ;UPDATE BA WITH BYTES TRANSFERRED
                50   51   02   10   EF   066B  1359         EXTZV   #16,#2,R1,R0            ;GET NEW MEMORY EXTENSION BITS
                00E3 C5   50   90   0670  1360         MOVB    R0,UCB$B_DY_XBA(R5)     ;AND SAVE IN UCB
                00E6 C5   51   B0   0675  1361         MOVW    R1,UCB$W_DY_SBA(R5)     ;SAVE BUFFER ADDRESS IN UCB
                                    067A  1362
                                    067A  1363                                         ;UPDATE DISK ADDRESS
                                    067A  1364 ;
                                    067A  1365 ; Here we update the disk address contained in UCB$L_DY_LMEDIA.
                                    067A  1366 ;       If we are doing LOGICAL I/O then we simply add one to
                                    067A  1367 ;       the logical sector number and if the sum of this addition
                                    067A  1368 ;       is EQUAL to the # of sectors on a track (26) we have an
                                    067A  1369 ;       overflow condition so we zero the logical sector # and bump
                                    067A  1370 ;       the logical track #.  We do the same for the logical track #
                                    067A  1371 ;       and the logical cylinder number.
                                    067A  1372 ;
                                    067A  1373 ;       Unfortunately if we are doing PHYSICAL I/O we have one little
                                    067A  1374 ;       glitch in that physical sector numbers are in the range
                                    067A  1375 ;       of 1 to 26 rather than in the range of 0 to 25.  Therefore
                                    067A  1376 ;       the following code treats the updating of the disk address
                                    067A  1377 ;       slightly differently in the case of LOGICAL and PHYSICAL I/O.
                                    067A  1378 ;
                                    067A  1379
                52   00EC C5   9E   067A  1380         MOVAB   UCB$L_DY_LMEDIA(R5),R2  ; R2 => Logical Media Address.
                  51    44 A5   9E   067F  1381         MOVAB   UCB$B_SECTORS(R5),R1    ; R1 => disk dimensions.
                  50    58 A5   D0   0683  1382         MOVL    UCB$L_IRP(R5),R0        ; R0 => IRP.
                          08   E0   0687  1383         BBS     #IRP$V_PHYSIO,-         ; If SET this IS PHYSICAL I/O so
                       11 2A A0        0689  1384                 IRP$W_STS(R0),60$       ;  branch to special treatment.
                     50   02   D0   068C  1385         MOVL    #2,R0                   ; Set loop count for LOGICAL I/O case.
                          62   96   068F  1386 50$:     INCB    (R2)                    ; Increment sector, track or cyl. #
                     81   62   91   0691  1387         CMPB    (R2),(R1)+              ; Test against limit for field.
                          2F   1F   0694  1388         BLSSU   80$                     ; LSSU implies NO overflow - so goto OK
                          82   94   0696  1389         CLRB    (R2)+                   ; Overflow, so reset to zero and
                            F4 50   F4   0698  1390         SOBGEQ  R0,50$                  ;  if GEQ loop to increment next field
                          1A   11   069B  1391         BRB     70$                     ; If we overflowed cylinders, branch.
                                    069D  1392 60$:                                    ; Special PHYSICAL I/O case.
                          62   96   069D  1393         INCB    (R2)                    ; Increment sector #.
                     81   62   91   069F  1394         CMPB    (R2),(R1)+              ; Compare to limit.
                          21   15   06A2  1395         BLEQ    80$                     ; If < or = to 26 - OK so branch.
                     82   01   90   06A4  1396         MOVB    #1,(R2)+                ; If overflow reset to 1 for sectors.
                          62   96   06A7  1397         INCB    (R2)                    ; And bump tracks.
                     81   62   91   06A9  1398         CMPB    (R2),(R1)+              ; Compare to limit.
                          17   1F   06AC  1399         BLSSU   80$                     ; If < OK so branch out.
                          82   94   06AE  1400         CLRB    (R2)+                   ; Clear overflowed track field and
                          62   96   06B0  1401         INCB    (R2)                    ;  increment cylinders.
                     61   62   91   06B2  1402         CMPB    (R2),(R1)               ; Test if we overflowed cylinders.
                          0E   1F   06B5  1403         BLSSU   80$                     ; If NOT, branch around to OK.
                                    06B7  1404 70$:                                    ; Here we have overflowed the cylinder
                                    06B7  1405                                         ;  field, but if the XFER is done it
                                    06B7  1406                                         ;  doesn't matter.
                00C0 C5   B5   06B7  1407         TSTW    UCB$W_BCR(R5)           ; Beyond last LBN - is XFER complete?
```

```
                     11   13  06BB  1408                BEQL    DY_PURGE                        ; If EQL - yes, so branch around.
               50 0134 8F   3C  06BD  1409                MOVZWL  #SS$_IVADDR,R0                  ;SET INVALID DISK ADDRESS STATUS
                  FB5A   31  06C2  1410                BRW     FUNCXT                          ;AND EXIT
                             06C5  1411
                             06C5  1412 80$:
                  00C0 C5    B5  06C5  1413                TSTW    UCB$W_BCR(R5)                   ; Any bytes remaining to transfer?
                     03      13  06C9  1414                BEQL    DY_PURGE                        ;IF EQL - TRANSFER COMPLETE
                  FE39       31  06CB  1415                BRW     COMXFER                         ;MORE BYTES REMAINING - CONTINUE
                             06CE  1416
                             06CE  1417                                                          ;
                             06CE  1418                                                          ; END OF "COMXFER:" LOOP
                             06CE  1419                                                          ;
                             06CE  1420
                             06CE  1421 ;
                             06CE  1422 ; PURGE DATAPATH
                             06CE  1423 ;
                             06CE  1424
                             06CE  1425 DY_PURGE:                                                ;PURGE DATAPATH
            00000000'GF      16  06CE  1426                JSB     G^IOC$PURGDATAP                 ;PURGE DATAPATH
                  04 50      E8  06D4  1427                BLBS    R0,DY_SAVE                      ;IF SET - NO PURGE ERROR
                  00E0 C5    96  06D7  1428                INCB    UCB$B_DY_ER(R5)                 ;SET PURGE ERROR
                             06DB  1429
                             06DB  1430 ;
                             06DB  1431 ; SAVE UBA REGISTERS FOR REGDUMP ROUTINE
                             06DB  1432 ;
                             06DB  1433
                             06DB  1434 DY_SAVE:                                                 ;SAVE UBA REGISTERS
                  00D4 C5 51 D0  06DB  1435                MOVL    R1,UCB$L_DY_DPR(R5)             ;SAVE DATAPATH REGISTER
                  51 00E6 C5 3C  06E0  1436                MOVZWL  UCB$W_DY_SBA(R5),R1             ;GET ORIGINAL BUFFER ADDRESS
51 02 10          00E3 C5 F0  06E5  1437                INSV    UCB$B_DY_XBA(R5),#16,#2,R1      ;INSERT EXTENDED ADDRESS BITS
                        50 D4  06EC  1438                CLRL    R0                              ;CLEAR R0 FOR WORD COUNT
                     7E A5 A1  06EE  1439                ADDW3   UCB$W_BCNT(R5),-                ;CALCULATE BYTES TRANSFERRED
                  50 00C0 C5   06F1  1440                        UCB$W_BCR(R5),R0                ;...
                        50 02 A6  06F5  1441                DIVW    #2,R0                           ;FORM WORDS TRANSFERRED IN R0
                        51 50 C0  06F8  1442                ADDL    R0,R1                           ;FORM FINAL BUFFER ADDRESS IN R1
               50 51 F9 8F 78  06FB  1443                ASHL    #-7,R1,R0                       ;SHIFT IN R0 FOR FINAL MAP REG NO.
                  50 01EF 8F B1  0700  1444                CMPW    #495,R0                         ;LEGAL MAP REGISTER?
                        05 18  0705  1445                BGEQ    10$                             ;IF GEQ - YES
                  50 01EF 8F 3C  0707  1446                MOVZWL  #495,R0                         ;RESTRICT MAP REGISTER NUMBER
               00D8 C5 6240 D0  070C  1447 10$:            MOVL    (R2)[R0],UCB$L_DY_FMPR(R5)      ;SAVE FINAL MAP REGISTER NUMBER
                     00DC C5 D4  0712  1448                CLRL    UCB$L_DY_PMPR(R5)               ;CLEAR PREVIOUS MAP REGISTER CONTENTS
                        50 D7  0716  1449                DECL    R0                              ;CALCULATE PREVIOUS MAP REGISTER NUMBER
                  0F    00 EC  0718  1450                CMPV    #VEC$V_MAPREG,#VEC$S_MAPREG,-   ;ANY PREVIOUS MAP REGISTER?
                     50 34 A3    071B  1451                        CRB$L_INTD+VEC$W_MAPREG(R3),R0  ;...
                        06 14  071E  1452                BGTR    RETREG                          ;IF GTR - NO
               00DC C5 6240 D0  0720  1453                MOVL    (R2)[R0],UCB$L_DY_PMPR(R5)      ;SAVE PREVIOUS MAP REGISTER
                             0726  1454
                             0726  1455 ;
                             0726  1456 ; DETERMINE EXIT - SPECIAL CONDITION, FATAL ERROR, RETRIABLE ERROR, OR SUCCESS
                             0726  1457 ;
                             0726  1458
                             0726  1459 RETREG:                                                  ;DETERMINE EXIT
            05 00CE C5 0F E0  0726  1460                BBS     #RY_CS_V_ERR,UCB$W_DY_CS(R5),2$ ;IF SET - DEVICE ERROR
                  72 00E0 C5 E9  072C  1461                BLBC    UCB$B_DY_ER(R5),10$             ;IF CLR - NO PURGE ERROR
                             0731  1462 2$:
                             0731  1463                ASSUME  UCB$W_DY_DB     EQ      UCB$W_DY_CS+2
                     00CE C5 D0  0731  1464                MOVL    UCB$W_DY_CS(R5),-               ; Remember values before reading
```

```
                0104 C5      0735  1465                    UCB$L_DY_SAVECS(R5)     ;  extended sense.
                  013D   30  0738  1466            BSBW    READ_ERROR_REGISTER     ; Read hardware error data into UCB.
                         0  073B  1467            CKOFL                           ; Check if device is offline.
                0104 C5  D0  077E  1468            MOVL    UCB$L_DY_SAVECS(R5),-   ; Restore values after reading
                00CE C5      0782  1469                    UCB$W_DY_CS(R5)         ;  extended sense.
            00000000'GF  16  0785  1470            JSB     G^ERL$DEVICERR          ;ALLOCATE AND FILL ERROR MESSAGE BUFFER
          18 009A C5    0F  E0  078B  1471            BBS     #IO$V_INHRETRY,UCB$W_FUNC(R5),20$  ;IF SET - RETRY INHIBITED
          0F 00D0 C5    0B  E0  0791  1472            BBS     #RY_DB_V_NXM,UCB$W_DY_DB(R5),15$  ;IF SET - NONEXISTENT MEMORY
          03 00D0 C5    00  E0  0797  1473            BBS     #RY_DB_V_CRC,UCB$W_DY_DB(R5),5$  ;IF SET - CRC ERROR
                  F96B   30  079D  1474            BSBW    RX2T1_REINIT                    ; Else go try to reset RX211.
                         0  07A0  1475
                         0  07A0  1476 ;
                         0  07A0  1477 ; RETRIABLE ERROR EXIT
                         0  07A0  1478 ;
                         0  07A0  1479
                  FA4A   31  07A0  1480 5$:     BRW     RETRYERR                ;RETRY EXIT
                         0  07A3  1481
                         0  07A3  1482 ;
                         0  07A3  1483 ; SUCCESSFUL OPERATION EXIT
                         0  07A3  1484 ;
                         0  07A3  1485
                  FA30   31  07A3  1486 10$:    BRW     NORMAL                  ;SUCCESSFUL EXIT
                         0  07A6  1487
                         0  07A6  1488 ;
                         0  07A6  1489 ; FATAL ERROR EXIT
                         0  07A6  1490 ;
                         0  07A6  1491
                         0  07A6  1492 15$:                                    ;NXM ERROR - INIT TO CLEAR
                  F962   30  07A6  1493            BSBW    RX211_REINIT            ; Execute RX211 initialize.
                  FA57   31  07A9  1494 20$:    BRW     FATALERR                ;FATAL ERROR EXIT
                         0  07AC  1495
                         0  07AC  1496 ;
                         0  07AC  1497 ; SPECIAL CONDITION EXIT (POWER FAILURE OR DEVICE TIMEOUT)
                         0  07AC  1498 ;
                         0  07AC  1499
                         0  07AC  1500 SPECOND:
                      05 E0  07AC  1501            BBS     #UCB$V_POWER,-          ;IF SET - POWER FAILURE
                32 64 A5     07AE  1502                    UCB$W_STS(R5),PWRFAIL   ;IF CLR - DEVICE TIMEOUT
            00000000'GF  16  07B1  1503            JSB     G^ERL$DEVICTMO          ;LOG DEVICE TIMEOUT
                         0  07B7  1504            SETIPL  UCB$B_FIPL(R5)          ;LOWER TO FORK LEVEL
                  F94D   30  07BB  1505            BSBW    RX211_REINIT            ; Execute RX211 initialize.
              50  022C 8F   3C  07BE  1506            MOVZWL  #SS$_TIMEOUT,R0         ;SET DEVICE TIMEOUT STATUS
                0080 C5    97  07C3  1507            DECB    UCB$B_ERTCNT(R5)        ;ANY ERROR RETRIES REMAINING?
                      0D   13  07C7  1508            BEQL    RESETXFR                ;IF EQL - NO
          64 A5   0040 8F    AA  07C9  1509            BICW    #UCB$M_TIMOUT,UCB$W_STS(R5)  ;CLEAR TIMEOUT STATUS
              53    58 A5    D0  07CF  1510            MOVL    UCB$L_IRP(R5),R3        ;RESTORE IRP ADDRESS
                  FA7C   31  07D3  1511            BRW     FEXL                    ;RETURN
                         0  07D6  1512
                         0  07D6  1513 RESETXFR:                               ;RESET TRANSFER BYTE COUNT
              53    58 A5    D0  07D6  1514            MOVL    UCB$L_IRP(R5),R3        ;GET ADDRESS OF I/O PACKET
          00C0 C5   32 A3    AE  07DA  1515            MNEGW   IRP$W_BCNT(R3),UCB$W_BCR(R5)  ;RESET BYTE COUNT
                  FA3C   31  07E0  1516            BRW     FUNCXT                  ;EXIT
                         0  07E3  1517
                         0  07E3  1518 PWRFAIL:                                ;POWER FAILURE
              64 A5    20   AA  07E3  1519            BICW    #UCB$M_POWER,UCB$W_STS(R5)  ;CLEAR POWER FAILURE BIT
                00D2 C5    B5  07E7  1520            TSTW    UCB$W_DY_DPN(R5)        ;ARE UBA RESOURCES ALLOCATED?
                      0C   13  07EB  1521            BEQL    10$                     ;IF EQL - NO
```

```
                     07ED  1522          RELDPR                          ;RELEASE DATA PATH
                     07F3  1523          RELMPR                          ;RELEASE MAP REGISTERS
                     07F9  1524 10$:     RELCHAN                         ;RELEASE CHANNEL IF OWNED
53   58 A5    D0     07FF  1525          MOVL    UCB$L_IRP(R5),R3        ;GET ADDRESS OF I/O PACKET
     2C A3    7D     0803  1526          MOVQ    IRP$L_SVAPTE(R3),-      ;RESTORE TRANSFER PARAMETERS
     78 A5           0806  1527                  UCB$L_SVAPTE(R5)        ;...
      F96F    31     0808  1528          BRW     DY_STARTIO              ;START REQUEST OVER
```

```
                                080B  1530          .SBTTL  INTERRUPT SERVICE ROUTINE
                                080B  1531 ;++
                                080B  1532 ; DY_INT - RX211 INTERRUPT SERVICE ROUTINE
                                080B  1533 ;
                                080B  1534 ; FUNCTIONAL DESCRIPTION:
                                080B  1535 ;
                                080B  1536 ;       THIS ROUTINE IS ENTERED VIA A JSB INSTRUCTION WHEN AN INTERRUPT
                                080B  1537 ;       OCCURS ON AN RX211 DISK CONTROLLER. IF THE INTERRUPT IS NOT EXPECTED,
                                080B  1538 ;       THE UNSOLICITED INTERRUPT ROUTINE DISMISSES THE INTERRUPT. IF
                                080B  1539 ;       THE INTERRUPT IS EXPECTED, DEVICE REGISTERS ARE SAVED AND THE
                                080B  1540 ;       DRIVER IS CALLED AT ITS INTERRUPT RETURN ADDRESS. THE DRIVER FORKS,
                                080B  1541 ;       CAUSING A RETURN TO THIS ROUTINE, WHICH RESTORES GENERAL REGISTERS
                                080B  1542 ;       AND DISMISSES THE INTERRUPT.
                                080B  1543 ;
                                080B  1544 ; INPUTS:
                                080B  1545 ;
                                080B  1546 ;       00(SP)  - POINTER TO ADDRESS OF THE IDB
                                080B  1547 ;       04(SP)  - SAVED R0
                                080B  1548 ;       08(SP)  - SAVED R1
                                080B  1549 ;       12(SP)  - SAVED R2
                                080B  1550 ;       16(SP)  - SAVED R3
                                080B  1551 ;       20(SP)  - SAVED R4
                                080B  1552 ;       24(SP)  - SAVED R5
                                080B  1553 ;       28(SP)  - PC AT THE TIME OF THE INTERRUPT
                                080B  1554 ;       32(SP)  - PSL AT THE TIME OF THE INTERRUPT
                                080B  1555 ;
                                080B  1556 ; OUTPUTS:
                                080B  1557 ;
                                080B  1558 ;       DEVICE REGISTERS ARE SAVED, IPL IS LOWERED TO FORK LEVEL, THE
                                080B  1559 ;       INTERRUPT IS DISMISSED, ALL REGISTERS EXCEPT R0-R5 ARE PRESERVED.
                                080B  1560 ;--
                                080B  1561
                                080B  1562 DY_INT::                                     ;INTERRUPT SERVICE ROUTINE
                      53   9E   D0  080B  1563          MOVL    @(SP)+,R3               ;REMOVE ADDRESS OF IDB FROM STACK
                                080E  1564          ASSUME  IDB$L_CSR    EQ  0
                                080E  1565          ASSUME  IDB$L_OWNER  EQ  4
                      54   63   7D  080E  1566          MOVQ    (R3),R4                 ;GET ADDRESS OF CSR AND UCB
                           55   D5  0811  1567          TSTL    R5                      ; Make sure we have OWNER.
                           3D   13  0813  1568          BEQL    DY_UNSOLNT              ; EQL implies RX controller has NO owner.
                00CE  C5   64   B0  0815  1569          MOVW    RY_CS(R4),UCB$W_DY_CS(R5)  ;SAVE CONTROL STATUS REGISTER
                      53   03   9A  081A  1570          MOVZBL  #F_READSECTOR/2,R3      ;GET READ SECTOR FUNCTION CODE
53   00E8  C5   03   01   ED  081D  1571          CMPZV   #1,#3,UCB$L_DY_XFER(R5),R3  ;WAS THIS A READ SECTOR FUNCTION?
                           08   12  0824  1572          BNEQ    10$                     ;IF NEQ - NO, SAVE ORIGINAL DELD BIT
        00D0  C5   02   A4   B0  0826  1573          MOVW    RY_DB(R4),UCB$W_DY_DB(R5)  ;SAVE DATA BUFFER REGISTER
                           13   11  082C  1574          BRB     20$                     ;
                      FFBF 8F   AA  082E  1575 10$:    BICW    #^C<RY_DB_M_DELD>,-     ;SAVE DELETED DATA BIT NOW IN UCB
                      00D0 C5       0832  1576                  UCB$W_DY_DB(R5)         ;...
53   02   A4   0040 8F   AB  0835  1577          BICW3   #RY_DB_M_DELD,RY_DB(R4),R3  ;GET ALL BUT DELD BIT FROM DBR
           00D0  C5   53   A8  083C  1578          BISW    R3,UCB$W_DY_DB(R5)      ;SAVE DATA BUFFER REGISTER
           64   4041 8F   AA  0841  1579 20$:    BICW    #RY_CS_M_IE!RY_CS_M_GO!-  ;DISABLE FURTHER INTERRUPTS
                           0846  1580                  RY_CS_M_INIT,RY_CS(R4)  ;...
                      01   E5  0846  1581          BBCC    #UCB$V_INT,-            ;IF CLR - INTERRUPT NOT EXPECTED
                07 64   A5       0848  1582                  UCB$W_STS(R5),DY_UNSOLNT  ;...
                           084B  1583
                53   10   A5   7D  084B  1584          MOVQ    UCB$L_FR3(R5),R3        ;RESTORE DRIVER CONTEXT
                     0C   B5   16  084F  1585          JSB     @UCB$L_FPC(R5)          ;CALL DRIVER AT INTERRUPT RETURN ADDRESS
                           0852  1586
```

```
                     0852  1587 DY_UNSOLNT:                                ;UNSOLICITED INTERRUPT
3F   BA              0852  1588          POPR    #^M<R0,R1,R2,R3,R4,R5>   ;RESTORE R0-R5
     02              0854  1589          REI                              ;RETURN FROM INTERRUPT
```

```
                     0855  1591          .SBTTL  REGISTER DUMP ROUTINE
                     0855  1592 ;++
                     0855  1593 ;
                     0855  1594 ; DY_REGDUMP - REGISTER DUMP ROUTINE
                     0855  1595 ;
                     0855  1596 ; FUNCTIONAL DESCRIPTION:
                     0855  1597 ;
                     0855  1598 ; THIS ROUTINE IS CALLED TO SAVE THE DEVICE REGISTERS AND UBA RESOURCE
                     0855  1599 ; REGISTERS IN A SPECIFIED BUFFER. IT IS CALLED FROM THE DEVICE ERROR
                     0855  1600 ; LOGGING ROUTINE AND FROM THE DIAGNOSTIC BUFFER FILL ROUTINE.
                     0855  1601 ;
                     0855  1602 ; INPUTS:
                     0855  1603 ;
                     0855  1604 ;       R0      - ADDRESS OF REGISTER SAVE BUFFER
                     0855  1605 ;       R4      - ADDRESS OF DEVICE CONTROL STATUS REGISTER (CSR)
                     0855  1606 ;       R5      - ADDRESS OF UNIT CONTROL BLOCK (UCB)
                     0855  1607 ;       UCB$B_DY_ER   - SPECIAL ERRORS: BIT 0 - DATAPATH PURGE ERROR
                     0855  1608 ;                                       BIT 1 - RX211 SWITCH SET FOR RX01
                     0855  1609 ;
                     0855  1610 ; OUTPUTS:
                     0855  1611 ;
                     0855  1612 ;       THE DEVICE AND UBA REGISTERS ARE SAVED IN THE SPECIFIED BUFFER.
                     0855  1613 ;       R0 CONTAINS THE ADDRESS OF THE NEXT EMPTY LONGWORD IN THE BUFFER.
                     0855  1614 ;       ALL REGISTERS EXCEPT R1 AND R2 ARE PRESERVED.
                     0855  1615 ;
                     0855  1616 ;--
                     0855  1617
                     0855  1618 DY_REGDUMP:                                  ;REGISTER DUMP ROUTINE
     80    09   D0   0855  1619          MOVL    #<RY_NUM_REGS+7>,(R0)+      ; Insert number of registers.
51   00CE  C5   DE   0858  1620          MOVAL   UCB$Q_DY_CS(R5),R1          ;GET ADDRESS OF SAVED DEVICE REGISTERS
     80    81   3C   085D  1621          MOVZWL  (R1)+,(R0)+                 ;DUMP DEVICE CONTROL STATUS REGISTER
     80    81   3C   0860  1622          MOVZWL  (R1)+,(R0)+                 ;DUMP DEVICE DATA BUFFER REGISTER
     80    81   3C   0863  1623          MOVZWL  (R1)+,(R0)+                 ;DUMP DATAPATH NUMBER
     80    81   D0   0866  1624          MOVL    (R1)+,(R0)+                 ;DUMP DATAPATH REGISTER
     80    81   D0   0869  1625          MOVL    (R1)+,(R0)+                 ;DUMP FINAL MAP REGISTER
     80    81   D0   086C  1626          MOVL    (R1)+,(R0)+                 ;DUMP PREVIOUS MAP REGISTER
     80    81   9A   086F  1627          MOVZBL  (R1)+,(R0)+                 ;DUMP SPECIAL ERROR REGISTER
                     0872  1628          ASSUME  RY_EXTENDED_STATUS_LENGTH  EQ  8
80   00F0  C5   7D   0872  1629          MOVQ    UCB$Q_DY_EXTENDED_STATUS(R5),(R0)+      ; Copy ERROR REGISTER data.
                05   0877  1630          RSB                                 ;RETURN
```

```
                        .SBTTL  READ_ERROR_REGISTER - Subroutine to read hardware error data

;
; READ_ERROR_REGISTER - subroutine called after a hardware error condition and
;       used to issue the READ ERROR REGISTER command.
;
; The Read Error Register command performs a DMA transfer of 4 words (8 bytes)
;       of hardware error status.  In order to accomplish our task here we must:
;
;       1. Save CRB and UCB fields having to do with the data transfer I/O
;             operation in progress.  These fields are:
;
;                a) CRB$L_INTD+VEC$W_MAPREG - the first UBA map register used
;                      to map the I/O buffer in Unibus virtual space.
;                b) CRB$L_INTD+VEC$B_NUMREG - the number UBA map registers
;                      currently allocated to map the I/O buffer.
;                c) CRB$L_INTD+VEC$B_DATAPATH - the UBA datapath being used for
;                      the transfer in progress.
;                d) UCB$L_SVAPTE, UCB$W_BOFF, and UCB$W_BCNT.
;
;       2. Load a zero into CRB$L_INTD+VEC$B_DATAPATH since the DMA transfer
;             of 8 bytes can easily make use of the direct datapath.
;
;       3. Load UCB$L_SVAPTE with the system virtual address of the page table
;             entry which maps the UCB$Q_DY_EXTENDED_STATUS field, the field
;             into which we will do the DMA transfer of the 8 bytes.
;
;       4. Load UCB$W_BOFF with the offset in its page of UCB$Q_DY_EXTENDED_STATUS.
;
;       5. Load UCB$W_BCNT with the length of UCB$Q_DY_EXTENDED_STATUS (8 bytes).
;
;       6. Once the above fields (steps 2-6) are loaded we can make use of
;             system routines to:
;
;                a) REQMPR - request UBA map registers to map
;                      UCB$Q_DY_EXTENDED_STATUS.
;                b) LOADUBA - load the allocated map registers with the
;                      appropriate data to realize the mapping.
;
;       7. Calculate the Unibus virtual address of UCB$Q_DY_EXTENDED_STATUS
;             and produce the values to insert into the RX211 (RX4T1)
;             registers, according to protocol, to effect the Read Error
;             Register command.
;
;       8. Execute the command.
;
;       9. Release UBA map registers and restore CRB and UCB fields.
;
;       10. If no TIMEOUT or POWERFAIL occurred, return to caller, else branch
;             to SPECOND.
;
; INPUTS:
;                R4 => CSR
;                R5 => UCB
;
; OUTPUTS:
;                Error Register data in UCB$Q_DY_EXTENDED_STATUS.
```

```
                                0878  1689 ;
                                0878  1690 ;               Registers R0, R1 and R2 are modified.
                                0878  1691 ;
                                0878  1692
                                0878  1693
                                0878  1694 READ_ERROR_REGISTER:
                009C C5 8ED0    0878  1695          POPL    UCB$L_DPC(R5)                   ; Save caller's return address.
                                087D  1696
                                087D  1697          ASSUME  VEC$W_MAPREG+2  EQ  VEC$B_NUMREG
                                087D  1698          ASSUME  VEC$B_NUMREG+1  EQ  VEC$B_DATAPATH
           50    24 A5    D0    087D  1699          MOVL    UCB$L_CRB(R5),R0                ; R0 => CRB.
                 34 A0    D0    0881  1700          MOVL    CRB$L_INTD+VEC$W_MAPREG(R0),-   ; Save MAPREG, NUMREG, and
              0100 C5          0884  1701                  UCB$L_DY_MAPREGTMP(R5)          ;  DATAPATH of current operation
                 37 A0    94    0887  1702          CLRB    CRB$L_INTD+VEC$B_DATAPATH(R0)   ; Insure direct path for READERROR
                                088A  1703
                                088A  1704          ASSUME  UCB$L_SVAPTE+4  EQ  UCB$W_BOFF
                                088A  1705          ASSUME  UCB$W_BOFF+2    EQ  UCB$W_BCNT
                 78 A5    7D    088A  1706          MOVQ    UCB$L_SVAPTE(R5),-              ; Save contents of UCB$L_SVAPTE,
              00F8 C5          088D  1707                  UCB$Q_DY_SVAPTETMP(R5)          ;  UCB$W_BOFF, and UCB$W_BCNT.
                                0890  1708
                                0890  1709 ;
                                0890  1710 ; Upto here we have saved all relevent data from the CRB and UCB.  Now we
                                0890  1711 ;       doctor up those fields in the CRB and UCB in order to:
                                0890  1712 ;
                                0890  1713 ;               1. Request UBA map registers to map the 4 word field
                                0890  1714 ;                       in the UCB which will serve as the target of the
                                0890  1715 ;                       READ ERROR REGISTER command.
                                0890  1716 ;
                                0890  1717 ;               2. Load these UBA map registers with the UBA Virtual Address
                                0890  1718 ;                       of this target area.
                                0890  1719 ;
                                0890  1720
                      08    B0  0890  1721          MOVW    #RY_EXTENDED_STATUS_LENGTH,-    ; Put length of target area so
                   7E A5        0892  1722                  UCB$W_BCNT(R5)                  ;  to allocate correct number
                                0894  1723                                                  ;  of UBA map registers.
                                0894  1724
           50    00F0 C5  9E    0894  1725          MOVAB   UCB$Q_DY_EXTENDED_STATUS(R5),R0 ; R0 => target area.
     7C A5 50    FE00 8F  AB    0899  1726          BICW3   #^XFE00,R0,UCB$W_BOFF(R5)       ; Put offset in page of target.
        50 50    15   09  EF    08A0  1727          EXTZV   S^#VA$V_VPN,S^#VA$S_VPN,R0,R0   ; R0 = VPN of target's page in
                                08A5  1728                                                  ;  system space.
                                08A5  1729
           51 00000000'GF    D0  08A5  1730          MOVL    G^MMG$GL_SPTBASE,R1             ; R1 => base of S0 page table.
              78 A5   6140   DE  08AC  1731          MOVAL   (R1)[R0],UCB$L_SVAPTE(R5)       ; NOT SURE IF THIS SHOULDN'T BE
                                08B1  1732                                                  ;  INDIRECT MOV.************
                                08B1  1733
                                08B1  1734          REQMPR                                  ; Request map registers.
                                08B7  1735          LOADUBA                                 ; Load map registers with proper
                                08BD  1736                                                  ;  contents to map the target.
                                08BD  1737
                                08BD  1738 ;
                                08BD  1739 ; Now we calculate the UBA virtual address of the target so as to be able to
                                08BD  1740 ;       issue the proper device command.
                                08BD  1741 ;
                   F882     30  08BD  1742          BSBW    DY_MERGE                        ; Merge GO_BIT, IE, etc into R2.
                52    0E    A8  08C0  1743          BISW    #F_READERROR,R2                 ; Or in the command.
                                08C3  1744
           51    24 A5    D0  08C3  1745          MOVL    UCB$L_CRB(R5),R1                ; R1 => CRB.
```

```
                50   7C A5    3C   08C7  1746          MOVZWL  UCB$W_BOFF(R5),R0                ; R0 = page offset of target.
                                   08CB  1747
                     34 A1    F0   08CB  1748          INSV    CRB$L_INTD+VEC$W_MAPREG(R1),-   ; Place low order 7 bits of map
              50     07  09         08CE  1749                  #9,#7,R0                        ;  reg number into R0 giving
                                   08D1  1750                                                  ;  low order 16 bits of UBA
                                   08D1  1751                                                  ;  virtual address of target.
                                   08D1  1752
                02   07  EF         08D1  1753          EXTZV   #7,#2,-                         ; Get high order 2 bits of map
                51     34 A1       08D4  1754                  CRB$L_INTD+VEC$W_MAPREG(R1),R1  ;  register number.
                   0C   51  F0      08D7  1755          INSV    R1,#RY_CS_V_XBA,-               ; Or in the high order two bits
                   52   02          08DA  1756                  #RY_CS_S_XBA,R2                 ;  of the UBA virtual address.
                                   08DC  1757
                   64   52  B0      08DC  1758          MOVW    R2,RY_CS(R4)                    ; Move command to hardware reg.
                                   08DF  1759
                   7E   50  7D      08DF  1760          MOVQ    R0,-(SP)                ;SAVE R0-R1
                                   08E2  1761          TIMEDWAIT TIME=#100*1000,-      ;ONE SECOND WAIT TIMEOUT
                                   08E2  1762                  INS1=<BITB      #RY_CS_M_TR!RY_CS_M_DONE,RY_CS(R4)>,- ;T/R OR DONE?
                                   08E2  1763                  INS2=<BNEQ      5$>,-   ;IF LSS - TRANSFER COMPLETE (T/R)
                                   08E2  1764                                  -       ;IF NON-ZERO - DONE BIT SET - ERROR
                                   08E2  1765                                  -       ;IF EQL - NEITHER, WAIT
                                   08E2  1766                  DONELBL=5$
                50   8E  7D         090A  1767          MOVQ    (SP)+,R0                ;RESTORE R0-R1
              64     A0 8F  93      090D  1768          BITB    #RY_CS_M_TR!RY_CS_M_DONE,RY_CS(R4) ;T/R OR DONE?
                        04  19      0911  1769          BLSS    6$                      ;IF LSS - TRANSFER COMPLETE (T/R)
                        02  13      0913  1770          BEQL    6$                      ;IF EQL - TIME HAS EXPIRED
                        26  11      0915  1771          BRB     20$                     ;DONE BIT SET - ERROR
                                   0917  1772 6$:                                      ;NORMAL RETURN
                                   0917  1773
                                   0917  1774 ;
                                   0917  1775 ; Now we load the UBA virtual address into the hardware DB register and wait
                                   0917  1776 ;       for the interrupt to occur.
                                   0917  1777 ;
                                   0917  1778
                                   0917  1779          DSBINT
             05 64 A5    05   E1   091D  1780          BBC     #UCB$V_POWER,UCB$W_STS(R5),10$  ; If clear, then proceed.
                                   0922  1781          ENBINT
                        1C  11      0925  1782          BRB     30$                     ; Branch around if POWERFAIL.
                                   0927  1783
                                   0927  1784 10$:
                02 A4   50   B0      0927  1785          MOVW    R0,RY_DB(R4)            ; Load register according to
                                   092B  1786                                          ;  protocol for command.
                                   092B  1787
                                   092B  1788          WFIKPCH 30$,#2                  ; Wait for interrupt.
                                   0935  1789          IOFORK
                        06  11      093B  1790          BRB     30$                     ; Branch around timeout re-entry.
                                   093D  1791 20$:
          64 A5   0040 8F   A8      093D  1792          BISW    #UCB$M_TIMOUT,UCB$W_STS(R5)     ; Set timeout flag.
                                   0943  1793 30$:
                                   0943  1794          SETIPL  UCB$B_FIPL(R5)          ; Lower IPL in case TIMEOUT.
                                   0947  1795
                                   0947  1796 ;
                                   0947  1797 ; Now we deallocate the Unibus map register we allocated above to map the
                                   0947  1798 ;       target area and then we restore the UCB and CRB fields to their
                                   0947  1799 ;       original values.
                                   0947  1800 ;
                                   0947  1801
                                   0947  1802          RELMPR
```

```
                     094D  1803
    00F8 C5   7D     094D  1804          MOVQ    UCB$Q_DY_SVAPTETMP(R5),-        ; Restore UCB$L_SVAPTE,
      78 A5          0951  1805                  UCB$L_SVAPTE(R5)                ;  UCB$W_BOFF and UCB$W_BCNT.
50    24 A5   D0     0953  1806          MOVL    UCB$L_CRB(R5),R0                ; R0 => CRB
    0100 C5   D0     0957  1807          MOVL    UCB$L_DY_MAPREGTMP(R5),-        ; Restore MAPREG, NUMREG and
      34 A0          095B  1808                  CRB$L_INTD+VEC$W_MAPREG(R0)     ;  DATAPATH.
                     095D  1809
    0060 8F   B3     095D  1810          BITW    #UCB$M_TIMOUT!UCB$M_POWER,-     ; See if we had a POWERFAIL
      64 A5          0961  1811                  UCB$W_STS(R5)                   ;  or a TIMEOUT.
         03   13     0963  1812          BEQL    40$                             ; EQL implies NO - so branch.
        FE44  31     0965  1813          BRW     SPECOND                         ; Branch out if POWER or TIMEOUT.
                     0968  1814 40$:
    009C D5   17     0968  1815          JMP     @UCB$L_DPC(R5)                  ; Return to caller.
                     096C  1816 DY_END:                                  ;ADDRESS OF LAST LOCATION IN DRIVER
                     096C  1817          .END
```

```
$$$                     = 00000020 R      02
$$OP                    = 00000002
ACP$ACCESS                ********    X   03
ACP$DEACCESS              ********    X   03
ACP$MODIFY                ********    X   03
ACP$MOUNT                 ********    X   03
ACP$READBLK               ********    X   03
ACP$WRITEBLK              ********    X   03
ATS_UBA                 = 00000001
AVAILABLE                 000001D0 R      03
COMXFER                   00000507 R      03
CRB$L_INTD              = 00000024
DC$_DISK                = 00000001
DDB$K_SLOW              = 00000003
DDB$L_ACPD              = 00000010
DDB$L_DDT               = 0000000C
DEV$M_AVL               = 00040000
DEV$M_DIR               = 00000008
DEV$M_ELG               = 00400000
DEV$M_FOD               = 00004000
DEV$M_IDV               = 04000000
DEV$M_NNM               = 00000200
DEV$M_ODV               = 08000000
DEV$M_RND               = 10000000
DEV$M_SHR               = 00010000
DPT$C_LENGTH            = 00000038
DPT$C_VERSION           = 00000004
DPT$INITAB                00000038 R      02
DPT$M_SVP               = 00000002
DPT$REINITAB              00000074 R      02
DPT$TAB                   00000000 R      02
DT$_RX02                = 0000000B
DT$_RX04                = 0000000C
DY$DDT                    00000000 RG     03
DYN$C_CRB               = 00000005
DYN$C_DDB               = 00000006
DYN$C_DPT               = 0000001E
DYN$C_UCB               = 00000010
DY_ALIGN                  0000016A R      03
DY_END                    0000096C R      03
DY_FUNCTABLE              00000038 R      03
DY_INT                    0000080B RG     03
DY_MERGE                  00000142 R      03
DY_PURGE                  000006CE R      03
DY_REGDUMP                00000855 R      03
DY_RX02_INIT              00000135 R      03
DY_RX21T_INIT             000000D8 R      03
DY_SAVE                   000006DB R      03
DY_STARTIO                0000017A R      03
DY_UNSOLNT                00000852 R      03
EMB$L_DV_REGSAV         = 0000004E
ERL$DEVICERR              ********    X   03
ERL$DEVICTMO              ********    X   03
EXE$ABORTIO               ********    X   03
EXE$GL_TENUSEC            ********    X   03
EXE$GL_UBDELAY            ********    X   03
EXE$IOFORK                ********    X   03
EXE$LCLDSKVALID           ********    X   03
EXE$ONEPARM               ********    X   03
EXE$SENSEMODE             ********    X   03
EXE$SETCHAR               ********    X   03
EXE$ZEROPARM              ********    X   03
FATALERR                  00000203 R      03
FEXL                      00000252 R      03
FORMAT                    00000299 R      03
FUNCTAB_LEN             = 000000A0
FUNCXT                    0000021F R      03
F_EMPTYBUFFER           = 00000002
F_FILLBUFFER            = 00000000
F_READERROR             = 0000000E
F_READSECTOR            = 00000006
F_READSTATUS            = 0000000A
F_SETDEN                = 00000008
F_WRITEDEL              = 0000000C
F_WRITESECTOR           = 00000004
IDB$L_CSR               = 00000000
IDB$L_OWNER             = 00000004
IO$V_DELDATA            = 00000006
IO$V_INHRETRY           = 0000000F
IO$_ACCESS              = 00000032
IO$_ACPCONTROL          = 00000038
IO$_AVAILABLE           = 00000011
IO$_CREATE              = 00000033
IO$_DEACCESS            = 00000034
IO$_DELETE              = 00000035
IO$_FORMAT              = 0000001E
IO$_MODIFY              = 00000036
IO$_MOUNT               = 00000039
IO$_PACKACK             = 00000008
IO$_READLBLK            = 00000021
IO$_READPBLK            = 0000000C
IO$_READVBLK            = 00000031
IO$_SENSECHAR           = 0000001B
IO$_SENSEMODE           = 00000027
IO$_SETCHAR             = 0000001A
IO$_SETMODE             = 00000023
IO$_UNLOAD              = 00000001
IO$_VIRTUAL             = 0000003F
IO$_WRITELBLK           = 00000020
IO$_WRITEPBLK           = 0000000B
IO$_WRITEVBLK           = 00000030
IOC$DIAGBUFILL            ********    X   03
IOC$LOADUBAMAP            ********    X   03
IOC$MNTVER                ********    X   03
IOC$PURGDATAP             ********    X   03
IOC$RELCHAN               ********    X   03
IOC$RELDATAP              ********    X   03
IOC$RELMAPREG             ********    X   03
IOC$REQCOM                ********    X   03
IOC$REQDATAP              ********    X   03
IOC$REQMAPREG             ********    X   03
IOC$REQPCHANL             ********    X   03
IOC$RETURN                ********    X   03
IOC$WFIKPCH               ********    X   03
```

```
IRP$L_MEDIA                     =  00000038
IRP$L_SVAPTE                    =  0000002C
IRP$S_FCODE                     =  00000006
IRP$V_DIAGBUF                   =  00000007
IRP$V_FCODE                     =  00000000
IRP$V_PHYSIO                    =  00000008
IRP$W_BCNT                      =  00000032
IRP$W_FUNC                      =  00000020
IRP$W_STS                       =  0000002A
MASKH                           =  00000008
MASKL                           =  04000000
MMG$GL_SPTBASE                     ********    X    03
NORMAL                             000001D6  R      03
PACKACK                            00000354  R      03
PR$_IPL                         =  00000012
PWRFAIL                            000007E3  R      03
READ_ERROR_REGISTER                00000878  R      03
RESETXFR                           000007D6  R      03
RETREG                             00000726  R      03
RETRYERR                           000001ED  R      03
RX211_REINIT                       0000010B  R      03
RY_CS                              00000000
RY_CS_M_DONE                    =  00000020
RY_CS_M_ERR                     =  00008000
RY_CS_M_GO                      =  00000001
RY_CS_M_IE                      =  00000040
RY_CS_M_INIT                    =  00004000
RY_CS_M_RX02                    =  00000800
RY_CS_M_TR                      =  00000080
RY_CS_S_DEN                     =  00000002
RY_CS_S_XBA                     =  00000002
RY_CS_V_DEN                     =  00000008
RY_CS_V_ERR                     =  0000000F
RY_CS_V_XBA                     =  0000000C
RY_CYLINDERS                    =  0000004D
RY_DB                              00000002
RY_DB_M_ACLO                    =  00000008
RY_DB_M_CRC                     =  00000001
RY_DB_M_DE                      =  00000010
RY_DB_M_DELD                    =  00000040
RY_DB_M_DRDY                    =  00000080
RY_DB_M_NXM                     =  00000800
RY_DB_M_WCO                     =  00000400
RY_DB_V_CRC                     =  00000000
RY_DB_V_DE                      =  00000004
RY_DB_V_DELD                    =  00000006
RY_DB_V_NXM                     =  0000000B
RY_DB_V_QDEN                    =  00000001
RY_DB_V_RX04                    =  00000009
RY_DENSITY_DOUBLE               =  00000001
RY_DENSITY_QUAD                 =  00000002
RY_DENSITY_SINGLE               =  00000000
RY_DPPE                         =  00000001
RY_DSDD                         =  000007C5
RY_DWPS                         =  00000080
RY_EXTENDED_STATUS_LENGTH       =  00000008
RY_NUM_REGS                     =  00000002
RY_QWPS                         =  00000100
RY_RX01SW                       =  00000002
RY_SECTORS                      =  0000001A
RY_SSDD                         =  000003DC
RY_SSQD                         =  000007B8
RY_SSSD                         =  000001EE
RY_SWPS                         =  00000040
SIZ...                          =  00000004
SPECOND                            000007AC  R      03
SS$_CTRLERR                     =  00000054
SS$_DRVERR                      =  0000008C
SS$_FORMAT                      =  000000BC
SS$_IVADDR                      =  00000134
SS$_IVBUFLEN                    =  0000034C
SS$_MEDOFL                      =  000001A4
SS$_NORMAL                      =  00000001
SS$_PARITY                      =  000001F4
SS$_RDDELDATA                   =  00000661
SS$_TIMEOUT                     =  0000022C
SS$_VOLINV                      =  00000254
UCB$B_DEVCLASS                  =  00000040
UCB$B_DEVTYPE                   =  00000041
UCB$B_DIPL                      =  0000005E
UCB$B_DY_ER                        000000E0
UCB$B_DY_LCT                       000000E2
UCB$B_DY_XBA                       000000E3
UCB$B_ERTCNT                    =  00000080
UCB$B_ERTMAX                    =  00000081
UCB$B_FEX                       =  00000092
UCB$B_FIPL                      =  0000000B
UCB$B_SECTORS                   =  00000044
UCB$B_TRACKS                    =  00000045
UCB$K_DY_LEN                    =  00000108
UCB$K_LCL_DISK_LENGTH           =  000000CC
UCB$L_CRB                       =  00000024
UCB$L_DEVCHAR                   =  00000038
UCB$L_DEVCHAR2                  =  0000003C
UCB$L_DPC                       =  0000009C
UCB$L_DY_DPR                       000000D4
UCB$L_DY_FMPR                      000000D8
UCB$L_DY_LMEDIA                    000000EC
UCB$L_DY_MAPREGTMP                 00000100
UCB$L_DY_PMPR                      000000DC
UCB$L_DY_SAVECS                    00000104
UCB$L_DY_XFER                      000000E8
UCB$L_FPC                       =  0000000C
UCB$L_FR3                       =  00000010
UCB$L_IRP                       =  00000058
UCB$L_MAXBLOCK                  =  000000B0
UCB$L_MEDIA                     =  000000BC
UCB$L_MEDIA_ID                  =  0000008C
UCB$L_SVAPTE                    =  00000078
UCB$M_DIAGBUF                   =  00000002
UCB$M_NOCNVRT                   =  00000004
UCB$M_ONLINE                    =  00000010
UCB$M_POWER                     =  00000020
UCB$M_TIMOUT                    =  00000040
```

```
UCB$M_VALID                     = 00000800
UCB$Q_DY_EXTENDED_STATUS          000000F0
UCB$Q_DY_SVAPTETMP                000000F8
UCB$V_INT                       = 00000001
UCB$V_POWER                     = 00000005
UCB$V_VALID                     = 0000000B
UCB$W_BCNT                      = 0000007E
UCB$W_BCR                       = 000000C0
UCB$W_BOFF                      = 0000007C
UCB$W_CYLINDERS                 = 00000046
UCB$W_DEVBUFSIZ                 = 00000042
UCB$W_DEVSTS                    = 00000068
UCB$W_DY_CS                       000000CE
UCB$W_DY_DB                       000000D0
UCB$W_DY_DPN                      000000D2
UCB$W_DY_PWC                      000000E4
UCB$W_DY_SBA                      000000E6
UCB$W_DY_WPS                      000000CC
UCB$W_FUNC                      = 0000009A
UCB$W_STS                       = 00000064
UCB$W_UNIT                      = 00000054
UNLOAD                            000001D0 R      03
VA$S_VPN                        = 00000015
VA$V_VPN                        = 00000009
VEC$B_DATAPATH                  = 00000013
VEC$B_NUMREG                    = 00000012
VEC$L_IDB                       = 00000008
VEC$L_INITIAL                   = 0000000C
VEC$L_UNITINIT                  = 00000018
VEC$S_DATAPATH                  = 00000005
VEC$S_MAPREG                    = 0000000F
VEC$V_DATAPATH                  = 00000000
VEC$V_MAPREG                    = 00000000
VEC$W_MAPREG                    = 00000010
XFER                              000003F4 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	00000108 (264.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
$$$105_PROLOGUE	00000089 (137.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
$$$115_DRIVER	0000096C (2412.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.04	00:00:00.43
Command processing	140	00:00:00.39	00:00:03.97
Pass 1	591	00:00:18.19	00:01:09.87
Symbol table sort	0	00:00:02.33	00:00:08.72

Pass 2	324	00:00:04.32	00:00:13.52
Symbol table output	31	00:00:00.16	00:00:00.33
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1122	00:00:25.46	00:01:36.87

The working set limit was 2400 pages.
153208 bytes (300 pages) of virtual memory were used to buffer the intermediate code.
There were 120 pages of symbol table space allocated to hold 2222 non-local and 76 local symbols.
1817 source lines were read in Pass 1, producing 22 object records in Pass 2.
53 pages of virtual memory were used to define 49 macros.

+----------------------------+
! Macro library statistics !
+----------------------------+

Macro library name	Macros defined
_255DUA28:[SYS.OBJ]LIB.MLB;1	34
_255DUA28:[SYSLIB]STARLET.MLB;2	10
TOTALS (all libraries)	44

2470 GETS were required to define 44 macros.

There were no errors, warnings or information messages.

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