RD31/32/51/52/53/54

RQDX3 RX33

RQDX3 FORMATTER
CZRQCE0

AH-U110E-MC
1 OF 1 AUG 1987

digital
MADE IN

B

.MAIN. MACRO V05.03 Thursday 15-Jan-87 14:33

```
.REM %(

                        IDENTIFICATION
                        --------------

PRODUCT CODE:   AC-U109C-MC

PRODUCT NAME:   CZRQCE0  RQDX3 FORMATTER

PRODUCT DATE:   JANUARY 16,  1987

MAINTAINER:     DIAGNOSTIC ENGINEERING

AUTHOR:         DIAGNOSTIC ENGINEERING
```

Table of contents .

TABLE OF CONTENTS

1.0 ABSTRACT

This formatter was written to format Winchester drives attached to the RQDX3 disk controller. All new drives being attached to the RQDX3 controller must be formatted so that the drive can be brought online for use by a MSCP server or in simpler terms to be used by an operating system. This disk formatter is similar to the RQDX1/2 disk formatter in that the same standard DUP dialog is used and similar standard formatter questions are passed by the controller to the host user. The formatter is different from the RQDX1/2 disk formatter because a table of disk formatting parameters is passed to the controller. The RQDX1/2 disk controller already has these tables in its firmware.

. The format program actual has 2 controller run programs in it. If the controller is an RQDX3, the program will down line load a program into the controller which will identify the drive according to its cylinder size. Since each of the DEC drives have a different cylinder size it will know which drive it is and therefore which parameter or UIT table to pass to the controller. The second program is already contained in the microcode. This program called "FORMAT" does the actual formatting of the drive. The host program just passes information back and forth to the controller local program.

The UIT, Unit Information Table is picked by the down line loaded auto sizer program (AUTOSZ). After the drive is known the format program will be run on the controller. This format program (FORMAT) is very similar to the RQDX1/2 format program. The only difference as stated before is that the UIT will be down line loaded into the drive if the down line load question is asked. Every time the drive is brought on line the UIT table which was placed on the drive by this formatter program will be transfered into the controller with all the drive parameters. As long as the UIT still exists on the drive it does not have to be passed in by the host user. Only if the user requests to "Down line load" information to the controller will the UIT table be passed to the drive. Note the RX33 floppy drive does not use the UIT tables. The RX33 drive parameters are stored in the firmware so a table wasn't necessary.

The UIT table contains information about the drive such as size, number of tracks per surface, etc. This information is already know for certain DEC acquired Winchester drives. These tables are usually different for the different drives manufactured. CAUTION do not use non DEC drives you are liable to destroy Format and Data stored on them.

All though not a goal of the diagnostic this program can be used to run standard DUP dialog local programs such as 'DIRECT'. These local programs are stored in the firmware.

2.0 HOW TO RUN IT?

2.1 HARDWARE REQUIREMENTS

An RQDX3 disk controller and one or more Winchester or RX33 dr'ves configured into a Q bus PDP-11 system.

2.2 SOFTWARE REQUIREMENTS

This d'agnost'c was written using DRS the D'agnost'c Supervisor. The diagnost'c is expected to be run under XXDP diagnostic operat'ng system. It 's also possible to run the formatter under APT.

2.3 QUESTIONS ASKED AND THEIR ANSWERS

2.3.1 HARDWARE QUESTIONS FROM DIAGNOSTIC SOFTWARE

The diagnostic 's a standard DRS program with the standard DRS commands. Below I have a script of the questions asked an the answers to the 'nitial DRS questions. The Default value for the IP address is 172150. This 's standard configuration address for the first MSCP controller on a system. Any other MSCP controllers on the system will have to be in the floating address space of the IO page. The default vector address is 154 any other value between 0 774 could be used but is not suggested. If you want the default answers then just hit the "return" key on the keyboard. The Formatter w'll run an auto sizer to determine the proper drive characteristic table to g've to the controller. Th's auto sizer will figure out how many cylinders on the drive and through a small look up table we decide which table to down-l'ne load to the RQDX3 controller. The user will have to enter a dr've number and a ser'al number. After th's a warn'ng message w'll appear ask'ng if the user wants to proceed. The default 's no so the/ user must type 'Y' in order to format his dr'ves.

Typical Diagnostic Script:

```
boot up XXDP
.RUN ZRQC??
ZRQCE0.BIN

DRSXM-A0
ZRQC-E-0
RQDX3 D'sk Format Utility
Unit is RD51,RD52,RD53,RD54,RX33,RD31,RD32        Please type yes to 'Change HW?'
Restart Address is 141656
DR>START

Change HW ?  Y
# Units ?  1

IP Address 172150 ? <rtn>
Vector Address 154 ? <rtn>
Logical Drive (0-255) 0 ? <rtn>
Drive Serial Number(1-32000) 12345 ? <rtn>

***** WARNING all the data on this drive will be DESTROYED ****

Proceed to format the drive N ? <Y><rtn>
```

2.3.2 UIT TABLES

The UIT tables are stored in th's program. There are 10 large data tables formed 'n this diagnostic that contain the drive parameters for certain DEC drives. There are only 6 RQDX3 Winchester drive manufactured. So only 6 of the tables contain any information. The others are there for future drives. The AUTOSZ program ran previous to the FORMAT program will determine what type of drive is to be formatted and which table to pass to the d'sk controller. Once in the disk controller the table will be written to the disk drive. This table should never be erased unless the dr've is broken or format is run again.

NOTE th's is only for the RQDX3 disk controller and NOT for the RQDX1/2.

Unit Information Tables listed:

Enter UIT:

UIT	Drive Name
0:	RD51
1:	RD52 part # 30 21721-02 (1 light on front panel)
2:	RD52 part # 30-23227-02 (2 lights on front panel)
3:	RD53
4:	RD31
5:	RD54
6:	RD32
7:	
10:	

2.4 PROGRAM MESSAGES AND FORMAT COMPLETION

When the format finally starts a "Format Begun" message will appear and in the end a "Format Complete" message w'll appear. There may be 60+ minutes between the messages. If the extended messages are allowed 3 "Verificat'on Pass XXXXXX Begun" messages may appear. Theses messages tell when the controller checks the blocks for bad spots in the disk surface. These passes take several minutes each and touch all the cyl'nders on the dr've. At the end of the format if extended messages are on a table w'll be printed out report'ng the results of the format. Usually there are several bad spots on a disk. This 's very common and is NOT a m'stake. These bad blocks are revectored to new areas on the disk. If the manufacturer's bad block information is used which is usually the case. There will only be 1 verificat'on pass. After the drive formats the autosizer program w'll be run again. This will park the heads on the inner most cylinder. Some manufactures have a parking area where the heads are placed before the dr've is physically moved or shipped to the customer. If you plan on mov'ng your system you should backup your system and run the formatter to put the heads on the parking area. This will help prevent damage to the heads and formatted data surfaces.

Complet'on Report:

xxx	Revectored LBNs
xxx	Pr'mary revectored LBNs
xxx	Secondary/tertiary revectored LBNs

```
xxx     Bad Blocks in the RCT area due to data errors
xxx     Bad Blocks in the DBN area due to data errors
xxx     Bad Blocks in the XBN area due to data errors
xxx     Blocks retried on check pass
FCT was not used
Format Completed

RQDX Drive xxxx finished

pass aborted for this unit
ZRQC EOP 1
    0 Cumulative errors

        Note that every time the disk formats successfully, the program
        drops the UNIT. This is purposely done so one doesn't reformat
        it twice.


        RX33 diskette formatting is a little varied in that several extra
        questions will be asked. These questions where installed mainly to
        protect the person trying to format a diskette on the same drive as
        their boot media. If the drive doing the formatting is not the boot
        drive then please ignore the warnings.

WARNING - Remove boot diskette if in drive.
Insert a diskette to be formatted & press <RETURN>.

Format Complete
FCT was not used
Format completed

Do you want to format another diskette?

If boot drive, reinsert boot diskette & press <RETURN>.

RQDX Drive xxxx finished
pass aborted for this unit
ZRQC EOP 1
    0 Cumulative errors


     2.5      EXECUTION TIME
    The execution time for this diagnostic varies greatly according
    to the size of the drive being formatted. If an error in the
    drive configuration or state such as a write protect switch
    being on, an error will occur right after all the questions have
    been answered. If there are no errors, the formatter will take
    between 5 minutes to 60 minutes depending on the drive being formatted.
    A RD51 takes around 10 minutes to format depending on the way
    questions are answered. A RD52 take between 10 & 25 minutes to format
    and a RD53 a very long time to format. The program checks continuously
    to make sure the controller is still working. If no progress is
    indicated by the progress indicator a timeout error will occur. If
    the disk controller goes off line for some unapparent reason the
    formatter will know. Either way if one checks the light on the
    Winchester to see if it is lite or check  the READY light of the drive
    for a flickering light, this will tell the user that the formatter is
```

working. When the formatter completes a Format complete' message w'll appear on the term'nal.

3. ERRORS

There are many types of errors possible wh'le formatting a drive. First the system has to be configured rignt. The drives have to be jumpered r'ght along w'th the disk controller. If you get an error read the ent're error message carefully. See if there 's something simple wrong such as loss and misconf'gured dr'ves before calling FS. This is usually the case very seldom do the dr've or controller break. So check the cables, check the jumpers, try several times and if you still can't format then call F'eld Service.

error #	Comment	Problem
0,SF0	;unknown response	Not a DUP standard local program or Data Error in local program execution.
1,HRD0	;Fatal DUP type returned	Error w'th Format program check detailed error message more then likely this will be a drive error or drive configuration error. If the detailed message has a GET STATUS error. This means that the drive you asked to format had the wrong status. Example offline, wr'te protected, RX50 instead of an RDxx, power plug is loose, jumpers are wrong.
2,DF3	;Can't do remote programs"	Wrong controller or bad microcode controller error.
3,SFT0	;"already active will do an ABORT cmd"	Wrong controller or bad microcode controller error. The controller was expected to be in an 'dle state but was found in an act've state. Try again and if still there check for ECOs and new M'crocode.
4,DF2	;wrong step bit set after interrupt	Controller 'nitial'zat'on error. Controller 's broken or at wrong address and someth'ng is in its place.
5,DF1	;controller timeout dur'ng hard in't	Controller error, controller is slow or it can t interrupt the Q bus. Contrcller 's dead.
6,SFT1	;wrong model #,wrong controller	This is not really an error. You are us'ng the wrong formatter program to for the wrong disk controller. It st'll m'ght work but no guarantees.
7,DF4	;NXM trap at controller IP address	Wrong configurat'on address of the controller check for wrong jumper sett'ngs.
8,SF100	;Unexpected 'nterrupt	Something in system interrupting or late interrupt. This could be the system clock or an 'nterrupt from an IO port.

If the interrupt is at address 4,10 probably a software error
Try again.

9,DF12 ;Fatal SA error
Controller crashed check detailed error message either dead
controller or configuration error.

10,DF11 ;Bad response packet
inappropriate command or soft controller error check
detail message for more info.

11,DF13 ;no progress shown after cmd timeout
The controller didn't indicate progress which means that it is
working very slow or is stuck. Leave the program running for a
couple minutes. If this message repeats then the drive is likely
broken. If you just get 1 message it is possible the controller
took to long to revector a block. This is probably a drive error
or a drive with many revector blocks.

12,DF14 ;no interrupt after get dust status command controller dead
The controller got lost. The program running ir the controller
got out of synch with the host program. This could mean several
things. Check for a loose controller or board loose cables. Try running
again after rebooting the system. If you still get the error check
the controller.

4. PROGRAM DESIGN AND FLOW

The program is kind of simple. There is only 1 command ring and
1 response ring. For every command send there is expected 1 response.
If the command sent times out a "Get DUST Status" command is sent to
check on the controller's progress. This usually happens when the
actual format is being done. The rest of the commands pass information
back and forth from the user to the controller without ever timing
out. This program is written according to UQSSP and DUP specs. This
specs can be acquired from NEWTON::ARCH$FILES:. At the start of the
program the INIT sequence brings the controller into the higher
protocolstate of running DUP commands. Once initialized the controller
executed a GET DUST STATUS command to make sure the controller is in an
Idle state.

If idle which it should be the program asks for a program name to run.
The EXECUTE LOCAL PROGRAM command is executed which should start the
program into the DUP dialog loop. This dialog is described in the DUP
spec. Here several SEND DATA and RECEIVE DATA commands are executed to
ask questions and supply information on the success and completion of
the local FORMAT program running in the RQDX3.

A pass will occur when the formatter has completed formatting
all the logical units.

5.0 GLOSSARY

ZRQCE0 follows the module name format described in the

RQ -- Identifies the hardware and thus the modu'e.

--C-- D'stiguishes between two or more different diagnostics for the same gener'c device. The sequence A, B, C, ETC. must be used for each additional diagnostic.

---E- Spec'f'es the module revision.

- --0 Specifies tre number of patches.

7.0 BIBLIOGRAPHY

UQSSP (NEWTON::ARCH$FILES:)
MSCP (NEWTON::ARCH$FILES:)
DUP (NEWTON::ARCH$FILES:)
DRS programmers manual (JON::disk$user1.[diaglib.drs])
XXDP programmer gu'de (JON::disk$user1:[diaglib.xxdp])

8.0 REVISION HISTORY

Revis'on B conta'ns an autosizing routine which w'll size the drive instead of having the user pick the drive table. Th's w'll keep people out of the systems and lower the changes of loose cables etc. Also added a AUTO mode which allows no manual interventions. Set up the default p-table to format drive 0-3. Since floppies are always the last dr've in the syste this is gauranteed to format all the drives in the system and error when 't gets to the floppy.

Revision C contains several changes. First RX33,RD31,RD54 support was adde'. The RX33 boot device questions where added. The autos'zer was fixed to also size for floppies. The Autosizer errors are now reported to the host along w'th what drives are located on what units and there drive size or floppy type. The default quest'on in manual mode was changed so that 'f an FCT (factory control table) is not present 'Bad Block Informat'on' it w'll not continue on. This was changed for all dr'ves except the RD51 which doesn't have a FCT table. Also there was a small change to the autosizer wh'ch affects vers'on C1 hardware etched RQDX3 boar ds specially the ones without the LUN ECO. The autosizer now run i.. the beg nn'ng and tne end. A head parking feature was added so that RD31 and RD32 heads would be parked 'n the 'nner most cylinder upon complet'on of the program. The autosizer ut'l'ty was updated to display a little more 'nformation.

Revision D replaces the head parking feature w'.h an MSCP test feature. This feature performs a ser'es of reads, writes, and compares to ver'fy that the media has been forma'ted correctly and that there are no defective blocks on the med'a. If a defect've block 's found, the med'a shoulc be promptly d'scarded to preserve the integr'ty of the data stored on it. Th's revis'on also has prov'sions to format an RD32. During the formatting of an RDxx, this formatter w'll generate a format

```
                                progress report (for RQDX3 uCode version 2 or higher) which displays
                                the progress made by the formatter at one minute intervals.  Finally,
                                code was added to correct the RQDX3 uCode version 2 inability to locate
                                the FCT during the FCT seek segment of the format.

                                Revision E implements fixes to known problems.
                        )*
```

M1


```
                                .MCALL SVC
                                SVC
                                .ENABLE ABS,AMA
                        000052  .=52
000052  010000                    .word       bit12                   ;extended monitor in XXDP
                        002000  .=2000
002000                          BGNMOD MOD1
002000                          POINTER BGNDU,BGNCLN,BGNPROT,BGNSETUP
002000                          HEADER ZRQC,E,0,600,0
002122                          DISPATCH 1
002126                          DESCRIPT <RQDX3 Format Disk Utility>
002160                          DEVTYPE <RD51,RD52,RD53,RD31,RD54,RX33,RD32 *** Answer "Y" to "Change HW (L) ?" ***>

```

```
  480 002274                            BGNHW DFPTBL
  481 002276  172150                            .WORD   172150          ;IP address
  482 002300  000154                            .WORD   154             ;Vector address
  483 002302  000000                            .WORD   000000          ;unit zero as default drive
  484 002304  030071                            .WORD   012345.         ;serial number
  485 002306  100000                            .word   100000          ;auto s'zer="yes", warning="no" or don't
  486                                                                   ;continue
  487 002310                            ENDHW
  488
```

```
490 002310                              EQUALS
                                        ;
                                        ; BIT DIFINITIONS
                                        ;
                100000                  BIT15== 100000
                040000                  BIT14== 40000
                020000                  BIT13== 20000
                010000                  BIT12== 10000
                004000                  BIT11== 4000
                002000                  BIT10== 2000
                001000                  BIT09== 1000
                000400                  BIT08== 400
                000200                  BIT07== 200
                000100                  BIT06== 100
                000040                  BIT05== 40
                000020                  BIT04== 20
                000010                  BIT03== 10
                000004                  BIT02== 4
                000002                  BIT01== 2
                000001                  BIT00== 1
                                        ;
                001000                  BIT9==  BIT09
                000400                  BIT8==  BIT08
                000200                  BIT7==  BIT07
                000100                  BIT6==  BIT06
                000040                  BIT5==  BIT05
                000020                  BIT4==  BIT04
                000010                  BIT3==  BIT03
                000004                  BIT2==  BIT02
                000002                  BIT1==  BIT01
                000001                  BIT0==  BIT00
                                        ;
                                        ; EVENT FLAG DEFINITIONS
                                        ;   EF32:EF17 RESERVED FOR SUPERVISOR TO PROGRAM COMMUNICATION
                                        ;
                                                                ;  BIT POSITION IN SECOND STATUS WORD
                000040                  EF.START==      32.     ; (100000) START COMMAND WAS ISSUED
                000037                  EF.RESTART==    31.     ; (040000) RESTART COMMAND WAS ISSUED
                000036                  EF.CONTINUE==   30.     ; (020000) CONTINUE COMMAND WAS ISSUED
                000035                  EF.NEW==        29.     ; (010000) A NEW PASS HAS BEEN STARTED
                000034                  EF.PWR==        28.     ; (004000) A POWER-FAIL/POWER-UP OCCURRED
                                        ;
                                        ;
                                        ; PRIORITY LEVEL DEFINITIONS
                                        ;
                000340                  PRI07== 340
                000300                  PRI06== 300
                000240                  PRI05== 240
                000200                  PRI04== 200
                000140                  PRI03== 140
                000100                  PRI02== 100
                000040                  PRI01== 40
                000000                  PRI00== 0
                                        ;
                                        ; OPERATOR FLAG BITS
                                        ;
                000004                  EVL==       4
```

```
                000010          LOT==       10
                000020          ADR==       20
                000040          IDU==       40
                000100          ISR==      100
                000200          UAM==      200
                000400          BOE==      400
                001000          PNT==     1000.
                002000          PRI==     2000
                004000          IXE==     4000
                010000          IBE==    10000
                020000          IER==    20000
                040000          LOE==    40000
                100000          HOE==   100000
     491                        .sbttl Literals
     492
     493                        ;+
     494                        ; Mask values to mask out specified flags
     495                        ;-
     496        000010                  UITothr = 10            ;UIT other
     497                                                        ;if UIT doesn't exist
     498
     499                        ;+
     500                        ; Misc.
     501                        ;-
     502        000004                  MaxDrv  = 4             ;Maximum Number of drives
     503        000002                  DUP.id  = bit1          ;DUP connection ID
     504        000000                  MSCP.id = 0             ;MSCP connection ID
     505        000007                  Mrqdx1  = 7.            ;model number for RQDX1
     506        000023                  Mrqdx3  = 19.           ;model number for RQDX3
     507        000001                  stdaln  = bit0          ;stand-alone modifier
     508        000367                  retry   = 367           ;Number of retries UDC
     509        000100                  LKvec   = 100           ;Line Clock (KW11-L) Vector
     510
     511                            ;+
     512                            ; Opcodes for DUP commands
     513                            ;
     514        000001                      op.gds  = 1
     515        000006                      op.abrt = 6
     516        000004                      op.sen  = 4
     517        000005                      op.rec  = 5
     518        000003                      op.elp  = 3
     519        000002                      op.esp  = 2
     520        000200                      op.end  = 200
     521                            ;+
     522                            ; Opcodes for MSCP commands       GJK
     523                            ;
     524        000004                      op.scc  = 4
     525        000003                      op.gus  = 3
     526        000011                      op.onl  = 11
     527        000041                      op.rd   = 41
     528        000042                      op.wr   = 42
     529                            ;+
     530                            ; Message type masks
     531                            ;-
     532        000001                      Question = 1
     533        000002                      DefQuest = 2
     534        000003                      Inform   = 3
```

L'terals

```
000004                          term'nat = 4
000005                          ftlerr   = 5
000006                          specl    = 6

177760                          type     = 177760
170000                          msgnbr   = 170000
                        ;+
                        ;Auto s'zer literals
                        ;-

                        ; Interrupt Service Rout'nes and Priority Levels

100002                  i$udc   =       100002          ; Pointer to UDC interrupt handler
100006                  i$clk   =       100006          ; Pointer to Clock interrupt handler
100016                  '$sec   =       100016          ; Po'nter to Sector Done Interrupt handler
000000                  ps0     =       0               ; Allow Any Interrupts
000340                  ps7     =       340             ; Inhibit Interrupts

                        ; CSRs

140002                  rw$pll  =       140002
140004                  w$fpl   =       140004
140006                  r$fps   =       140006
140010                  r$dat   =       140010
140012                  r$cmd   =       140012
140020                  w$dat   =       140020
140022                  w$cmd   =       140022

                        ; RECEIVE DATA ASCII reply message types:

000020                  .a.typ  =       20                      ; ASCII Message Type Multiplier
000020                  .a.que  =       1*.a.typ                ; Question
000040                  .a.def  =       2*.a.typ                ; Default quest'on
000060                  .a.'nf  =       3*.a.typ                ; Informat'on
000100                  .a.ter  =       4*.a.typ                ; Term'nation
000120                  .a.fat  =       5*.a.typ                ; Fatal error

                        ; RECEIVE DA'A binary message types.

000140                  .b.spl  =       6*.a.typ                ; Spec'al

                        ; Status Codes returned by SIZER (Success is zero)

000001                  erudon          =       1               ; UDC Never Done
000002                  eru'nt          =       2               ; UDC Never Interrupted
000003                  ersek0          =       3               ; Couldn t Restore to Cyl 0

                        ; UDC Commands

000000                  u.res           =       0               ; Reset 9224
000001                  u.dd            =       1               ; Deselect Dr've
000003                  u.rd            =       3               ; Restore Dr've
000005                  u.si1           =       5               ; Step In One Cylinder
000007                  u.so1           =       7               ; Step Out One Cylinder
000044                  u.srd           =       44              ; Select Winchester Drive
000054                  u.srx           =       54              ; Select Floppy Drive
000100                  u.srp           =       100             ; Set Reg'ster Po'nter
```

Literals

```
592          000300                 rd.mode          =       300             ; RD Mode
593
594                                 ; Literals Used by Line Clock
595
596          177546                 LKS              ==      177546          ;LINE CLOCK --- bit 6 enables interrupts
597          000100                 LKSvct           ==      100             ;Line Clock Vctr Addr -- pri 5 or lower
598
```

```
                        .sbttl  Macro Def'n't'ons


                        ;*
                        ;       Execute a GET DUST STATUS command and the check the response.
                        ;

000000                  A=0
000001                  B=1
                        .MACRO  GETDUST                         ;Execute a GET DUST STATUS command
                        B=B+1                                   ;increment the CRN number
                        gdstmp  \B                              ;call variable B as if it were a
                                                                ;number (\)
                        .ENDM

                        .MACRO  GDSTMP  B
                                .list
                        GDS'B:  bit     #bit15,cmdrng+2         ;test ownersh'p of r'ng make sure we own
                                                                ;'t
                                bne     GDS'B                   ;'f we don't own it wait unt'l we do
                                mov     #14.,cmdlen             ;load length of packet to be sent
                                movb    #0,cmdlen+2             ;load msg type and credit
                                movb    #dup.id,cmdlen+3        ;load DUP connect'on ID
                                'nc     cmdpak                  ;load new CRN
                                clr     cmdpak+2
                                clr     cmdpak+4
                                clr     cmdpak+6
                                mov     #op.gds,cmdpak+10       ;load up opcode
                                clr     cmdpak+12               ;no mod'f'ers

                                mov     #RFD'B,@vector          ;New vector place
                                mov     #rsppak,rsprng          ;load response packet area into r'ng
                                mov     #cmdpak,cmdrng          ;load command packet area 'nto r'ng
                                mov     #140000,RSPRNG+2        ;Port ownersh'p b't.
                                mov     #bit15,CMDRNG+2
                                jsr     pc,POLLWT               ;Go to poll and wa't rout'ne.

                        ;*******************************************************************

                            RFD'B:                              ;Intr to here.
                                add     #6,sp                   ;f'x stack for interrupt (4), pollwt
                                                                ;subrtn (2)
                                mov     #'ntsrv,@vector         ;Change vector
                                jsr     pc,RSPCHK               ;Go to rout'ne that will check on
                                                                ;the response recvd from the mut.
                                                                ;it will check the cmd ref
                                                                ;num, the endcode and status.
                                .nlist
                        .ENDM
```

Macro Definitions

```
651
652
653                             ;+
654                             ;       Execute an ABORT command and then checks the response.
655                             ;-
656
657
658                             .MACRO  ABRT                            ;Execute an ABORT command
659                             B=B+1                                   ;increment the CRN number
660                             abrttmp \B                              ;call variable B as if it were a
661                                                                     ;number (\)
662                             .ENDM
663
664                             .MACRO  ABRTTMP B       ,
665                                     .list
666                             ABRT'B: bit     #bit15,cmdrng+2         ;test ownersh'p of ring make sure we
667                                                                     ;own it
668                                     bne     ABRT'B                  ;if we don't own it wait until we do
669                                     mov     #14.,cmdlen             ;load length of packet to be sent
670                                     movb    #0,cmdlen+2             ;load msg type and credit
671                                     movb    #dup.id,cmdlen+3        ;load DUP connect'on ID
672                                     inc     cmdpak                  ;load new CRN
673                                     clr     cmdpak+2
674                                     clr     cmdpak+4
675                                     clr     cmdpak+6
676                                     mov     #op.abrt,cmdpak+10      ;load up opcode
677                                     clr     cmdpak+12               ;no mod'fiers
678
679                                     mov     #RFD'B,@vector          ;New vector place
680                                     mov     #rsppak,rsprng          ;load response packet area into ring
681                                     mov     #cmdpak,cmdrng          ;load command packet area into ring
682                                     mov     #140000,RSPRNG+2        ;Port ownership bit.
683                                     mov     #bit15,CMDRNG+2
684                                     jsr     pc,POLLWT               ;Go to poll and wait routine.
685
686                             ;*************************************************************************
687
688                                 RFD'B:                              ;Intr to here.
689                                     add     #6,sp                   ;fix stack for interrupt (4), pollwt
690                                                                     ;subrtn (2)
691                                     mov     #intsrv,@vector         ;Change vector
692                                     jsr     pc,RSPCHK               ;Go to routine that will check on
693                                                                     ;the response recvd from the mut.
694                                                                     ;it will check the cmd ref
695                                                                     ;num, the endcode and status.
696                                     .nlist
697                             .ENDM
```

Macro Definitions

```
699
700
701                             ;+
702                             ;       Execute a Send data cmd in dup and then check the response
703                             ;                       for the proper 'nfo
704                             ;-
705
706
707                             .MACRO  SENDDAT SPLACE,SBYTCN           ;Execute a Send Data command
708                             B=B+1                                   ;increment the CRN number
709                             sendtmp \B,SPlace,Sbytcn                ;call variable A,B as 'f 't were a
710                                                                     ;number (\)
711                             .ENDM
712
713                             .MACRO  SENDTMP B,Splace,Sbytcnt
714                                     .list
715                             SDT'B:  bit     #bit15,cmdrng+2         ;test ownership of r'ng make sure we
716                                                                     ;own 't
717                                     bne     SDT'B                   ;if we don't own it wait unt'l we do
718                                     mov     #34,cmdlen              ;load length of packet to be sent
719                                     movb    #0,cmdlen+2             ;load msg type and credit
720                                     movb    #dup.id,cmdlen+3        ;load DUP connect'on ID
721                                     inc     cmdpak                  ;load new CRN
722                                     clr     cmdpak+2
723                                     clr     cmdpak+4
724                                     clr     cmdpak+6
725                                     mov     #op.sen,cmdpak+10       ;load up opcode
726                                     clr     cmdpak+12               ;no mod'fiers
727                                     mov     Sbytcnt,cmdpak+14
728                                     clr     cmdpak+16
729                                     mov     Splace,cmdpak+20        ;load address of buffer descr'ptor
730                                     clr     cmdpak+22
731                                     clr     cmdpak+24
732                                     clr     cmdpak-26
733                                     clr     cmdpak+30
734                                     clr     cmdpak+32
735
736                                     mov     #RFD'B,@vector          ;New vector place
737                                     mov     #rsppak,rsprng          ;load response packet area into r'ng
738                                     mov     #cmdpak,cmdrng          ;load command packet area 'nto ring
739                                     mov     #140000,RSPRNG+2        ;Port ownership bit.
740                                     mov     #b't15,CMDRNG+2
741                                     jsr     pc,POLLWT               ;Go to poll and wait routine.
742
743                             ;*************************************************************************
744
745                                 RFD'B:                              ;Intr to here.
746                                     add     #6,sp                   ;fix stack for interrupt (4), pollwt
747                                                                     ;subrtn (2)
748                                     mov     #'ntsrv,@vector         ;Change vector
749                                     jsr     pc,RSPCHK               ;Go to rout'ne that will check on
750                                                                     ;the response recvd from the mut.
751                                                                     ;'t w'll check the cmd ref
752                                                                     ;num, the endcode and status.
753                                     .nlist
754                             .ENDM
```

Macro Definitions

```
                        ;+
                        ;       Execute a Receive Data command and the check the response.
                        ;-


                        .MACRO  RECVDAT Rplace,Rbytcnt          ;Execute a Send Data command
                        B=B+1                                   ;increment the CRN number
                        recvtmp \B,Rplac-,Rbytcnt               ;call variable A,B as if it were a
                                                                ;number (\)
                        .ENDM

                        .MACRO  RECVTMP B,RPlace,Rbytcnt
                                .list
                        RCD'B:  bit     #bit15,cmdrng+2         ;test ownership of ring make sure we
                                                                ;own it
                                bne     RCD'B                   ;if we don t own it wait until we co
                                mov     #34,cmdlen              ;load length of packet to be sent
                                movb    #0,cmdlen+2             ;load msg type and credit
                                movb    #dup.id,cmdlen+3        ;load DUP connection ID
                                inc     cmdpak                  ;load new CRN
                                clr     cmdpak+2
                                clr     cmdpak+4
                                clr     cmdpak+6
                                mov     #op.rec,cmdpak+10       ;load up opcode
                                clr     cmdpak+12               ;no modifiers
                                mov     Rbytcnt,cmdpak+14
                                clr     cmdpak+16
                                mov     Rplace,cmdpak+20        ;load address of buffer descriptor
                                clr     cmdpak+22
                                clr     cmdpak+24
                                clr     cmdpak+26
                                clr     cmdpak+30
                                clr     cmdpak+32

                                mov     #RFD'B,@vector          ;New vector place
                                mov     #rsppak,rsprng          ;load response packet area into ring
                                mov     #cmdpak,cmdrng          ;load command packet area into ring
                                mov     #140000,RSPRNG+2        ;Port ownership bit.
                                mov     #bit15,CMDRNG+2
                                jsr     pc,POLLWT               ;Go to poll and wait routine.

                        ;*****************************************************************************

                            RFD'B:                              ;Intr to here.
                                add     #6,sp                   ;fix stack for interrupt (4), pollwt
                                                                ;subrtn (2)
                                mov     #intsrv,@vector         ;Change vector
                                jsr     pc,RSPCHK               ;Go to routine that will check on
                                                                ;the response recvd from the mut.
                                                                ;it will check the cmd ref
                                                                ;num, the endcode and status.
                                .nlist
                        .ENDM
```

Macro Definitions

```
812
813
814                                     ;*
815                                     ;       Execute a Execute Local Program command and the check the response.
816                                     ;-
817
818
819                                     .MACRO  EXLCPRG Enamadr                 ;Execute a Send Data command
820                                     B=B+1                                   ;increment the CRN number
821                                     elptmp  \B,Enamadr                      ;call variable A,B as if it were a
822                                                                             ;number (\)
823                                     .ENDM
824
825                                     .MACRO  ELPTMP  B,Enamadr
826                                             .list
827                                     ELP'B:  bit     #bit15,cmdrng+2         ;test ownership of ring make sure we
828                                                                             ;own it
829                                             bne     ELP'B                   ;if we don t own it wait until we do
830                                             mov     #22,cmdlen              ;load length of packet to be sent
831                                             movb    #0,cmdlen+2             ;load msg type and credit
832                                             movb    #dup.id,cmdlen+3        ;load DUP connection ID
833                                             inc     crnpak                  ;load new CRN
834                                             clr     cmdpak+2
835                                             clr     cmdpak+4
836                                             clr     cmdpak+6
837                                             mov     #op.elp,cmdpak+10       ;load up opcode
838                                             mov     #stdaln,cmdpak+12       ;stand alone modifier
839                                             mov     #6,r0                   ;6 letters transfer
840                                             mov     #cmdpak+14,r1           ;starting address to place program name
841                                             mov     #Enamadr,r2             ;start of Program Name
842                                     rfdj'B: movb    (r2)+,(r1)+             ;add 2 to bycnt then store
843                                             sob     r0,rfdj'B
844
845                                             mov     #RFD'B,@vector          ;New vector place
846                                             mov     #rsppak,rsprng          ;load response packet area into ring
847                                             mov     #cmdpak,cmdrng          ;load command packet area into ring
848                                             mov     #140000,RSPRNG+2        ;Port ownership bit.
849                                             mov     #bit15,CMDRNG+2
850                                             jsr     pc,POLLWT               ;Go to poll and wait routine.
851
852                                     ;***************************************************************************
853
854                                         RFD'B:                              ;Intr to here.
855                                             add     #6,sp                   ;fix stack for interrupt (4), pollwt
856                                                                             ;subrtn (2)
857                                             mov     #intsrv,@vector         ;Change vector
858                                             jsr     pc,RSPCHK               ;Go to routine that will check on
859                                                                             ;the response recvd from the mut.
860                                                                             ;it will check the cmd ref
861                                                                             ;num  the endcode and status.
862                                             .nlist
863                                     .ENDM
864
```

Macro Definitions

```
                        ;+
                        ;       Execute a Eexcute Suppl'ed Program command and the check the response.
                        ;-


                        .MACRO  EXCSUPPRG Progname,Proqsize     ;Execute a Suppl'ed program command
                        B=B+1                                   ;increment the CRN number
                        esptmp  \B,Progname,Progsi.e            ;call variable A,B as 'f it were a
                                                                ;number (\)
                        .ENDM

                        .MACRO  ESPTMP  B,Progname,Progs'ze
                                .l'st
                        ESP B:  b't     #bit15,cmdrng+2         ;test ownersh'p of ring make sure we
                                                                ;own 't
                                bne     ESP'B                    ;f we don't own it wait unt'l we do
                                mov     #50,cmdlen              ;load length of packet to be sent
                                movb    #0,cmdlen+2             ;load msg type and cred't value
                                movb    #dup.'d,cmdlen+3        ;load DUP connect'on ID
                                inc     cmdpak
                                clr     CMDpak+2
                                clr     CMDpak+4
                                clr     CMDpak+6
                                mov     #op.esp,CMDpak+10       ;load up opcode
                                mov     #0,CMDpak+12            ;no stand alone modifier
                                mov     Progs ze,cmdpak+14      ;load length of prg into buffer
                                clr     cmdpak+16
                                mov     Progname,cmdpak+20      ;starting address of cownl'ne load prg
                                clr     CMDpak+22
                                clr     CMDpak+24
                                clr     CMDpak+26
                                clr     CMDpak+30
                                clr     CMDpak+32
                                clr     CMDpak+34               ;overlay buffer descr'ptor
                                clr     CMDpak+36
                                clr     CMDpak+40
                                clr     CMDpak+42
                                clr     CMDpak+44
                                clr     CMDpak+46
                                mov     #RFD'B,@vector          ;New vector place
                                mov     #rsppak,rsprng          ;load response packet area 'nto r'ng
                                mov     #cmdpak,cmdrng          ;load command packet area 'nto ring
                                mov     #140000,RSPRNG+2        ;Port ownership bit.
                                mov     #b't15,CMDRNG+2
                                jsr     pc,POLLWT               ;Go to poll and wa't rout'ne.
                        ;*****************************************************************************
                            RFD'B:                              ;Intr to here.
                                add     #6,sp                   ;fix stack for interrupt (4), pollw.
                                                                ;subrtn (2)
                                mov     #'ntsrv,@vector         ;Change vector
                                jsr     pc,RSPCHK               ;Go to routine that will check on
                                                                ;the response recvd from the mut.
                                .nl'st                          ;it will check the cmd ref
                        .ENDM                                   ;num, the endcode and status.

```

Macro Definitions

```
924
925
926                             ;+
927                             ;       Execute an MSCP SET CONTROLLER CHARACTERISTICS command and
928                             ;                               check the response
929                             ;-
930
931
932                             .MACRO  SCC             ;Execute an MSCP SET CONTROLLER CHARACTERISTICS command
933                             B=B+1                   ;increment the CRN number
934                             scctmp  \B              ;Call variable B as if it were a number (\)
935                             .ENDM
936
937                             .MACRO  SCCTMP  B
938                                     .list
939                             SCC'B:  bit     #bit15,cmdrng+2         ;test ownership of ring to make sure
940                                                                     ;we own it
941                                     bne     SCC'B                   ;if we don't, wait until we do
942                                     mov     #40,cmdlen              ;load length of packet to be sent
943                                     movb    #0,cmdlen+2             ;load message type and credit value
944                                     mov     #MSCP.id,cmdlen+3       ;load MSCP connection ID
945                                     ir      cmdpak                  ;load new CRN
946                                     clr     cmdpak+2
947                                     clr     cmdpak+4
948                                     clr     cmdpak 6
949                                     mov     #op.scc,cmdpak+10       ;load opcode
950                                     clr     cmdpak+12               ;load modifiers
951                                     clr     cmdpak 14               ;NO MODIFIERS
952                                     clr     cmdpak+16               ;load controller flags
953                                     clr     cmdpak+20               ;load default MSCP timeout value
954                                     clr     cmdpak+22
955                                     clr     cmdpak+24
956                                     clr     cmdpak+26
957                                     clr     cmdpak+30
958                                     clr     cmdpak+32
959                                     clr     cmdpak+34
960                                     cl.     cmdpak+36
961
962                                     mov     #RFD'B,@vector          ;NEW VECTOR PLACE
963                                     mov     #rsppak,rsprng          ;load response packet area into ring
964                                     mov     #cmdpak,cmdrng          ;load command packet area into ring
965                                     mov     #140000,rsprng+2        ;PORT OWNERSHIP BIT.
966                                     mov     #bit15,cmdrng+2
967                                     jsr     pc,POLLWT               ;GO TO POLL AND WAIT ROUTINE.
968                             ;*******************************************************************
969                                 RFD'B:                              ;INTR TO HERE.
970                                     add     #6,sp                   ;fix stack for interrupt (4),
971                                                                     ;pollwt subrtn (2)
972                                     mov     #intsrv,@vector         ;CHANGE VECTOR
973                                     jsr     pc,RSPCHK               ;Go to routine that will check on
974                                                                     ;the response recvd from the mut.
975                                                                     ;it will check the cmd ref
976                                                                     ;num, the endcode, and status.
977                                     .nlist
978                             .ENDM
```

Macro Definitions

```
                        ;+
                        ;       Execute an MSCP GET UNIT STATUS command and check the response
                        ;-


                        .MACRO  GUS             ;Execute an MSCP GET UNIT STATUS command
                        B=B+1                   ;increment the CRN number
                        gustmp  \B              ;Call variable B as if it were a number (\)
                        .ENDM

                        .MACRO  GUSTMP  B
                                .list
                        GUS'B:  b't     #bit15,cmdrng+2         ;test ownership of ring to make sure
                                                                ;we own it
                                bne     GUS'B                   ;if we don't, wait until we do
                                mov     #14,cmdlen              ;load length of packet to be sent
                                movb    #0,cmdlen+2             ;load message type and credit value
                                movb    #MSCP.id,cmdlen+3       ;load MSCP connection ID
                                inc     cmdpak                  ;load new CRN
                                clr     cmdpak+2
                                mov     UNIT,cmdpak+4           ;unit number
                                clr     cmdpak+6
                                mov     #op.gus,cmdpak+10       ;load opcode
                                clr     cmdpak+12               ;load modifiers
                                clr     cmdpak+14               ;NO MODIFIERS

                                mov     #RFD'B,@vector          ;NEW VECTOR PLACE
                                mov     #rsppak,rsprng          ;load response packet area into ring
                                mov     #cmdpak,cmdrng          ;load command packet area into ring
                                mov     #140000,rsprng+2        ;PORT OWNERSHIP BIT.
                                mov     #b't15,cmdrng+2
                                jsr     pc,POLLWT               ;GO TO POLL AND WAIT ROUTINE.
                        ;*************************************************************************
                            RFD'B:                              ;INTR TO HERE.
                                add     #6,sp                   ;fix stack for interrupt (4),
                                                                ;pollwt subrtn (2)
                                mov     #intsrv,@vector         ;CHANGE VECTOR
                                mov     rsppak+44,trksiz
                                mov     trksiz,bytsiz           ;Calculate bytes per track
                                swab    bytsiz
                                asl     bytsiz                  ;BYTSIZ = TRKSIZ * 1000 Octal
                                jsr     pc,RSPCHK               ;Go to routine that will check on
                                                                ;the response recvd from the mut.
                                                                ;It will check the cmd ref
                                                                ;num, the endcode, and status.
                                .nlist
                        .ENDM
```

Macro Definitions

```
1030
1031
1032                            ;+
1033                            ;       Execute an MSCP ONLINE command and check the response
1034                            ;-
1035
1036
1037                            .MACRO  ONLINE          ;Execute an MSCP ONLINE command
1038                            B=B+1                   ;increment the CRN number
1039                            onltmp  \B              ;Call var'able B as if it were a number (\)
1040                            .ENDM
1041
1042                            .MACRO  ONLTMP  B
1043                                    .list
1044                            ONL'B:  bit     #bit15,cmdrng+2         ;test ownership of ring to make sure
1045                                                                    ;we own it
1046                                    bne     ONL'B                   ;if we don't, wait until we do
1047                                    mov     #44,cmdlen              ;load length of packet to be sent
1048                                    movb    #0,cmdlen+2             ;load message type and cred't value
1049                                    movb    #MSCP.id,cmdlen+3       ;load MSCP connection ID
1050                                    inc     cmdpak                  ;load new CRN
1051                                    clr     cmdpak+2
1052                                    mov     UNIT,cmdpak+4           ;unit number
1053                                    clr     cmdpak+6
1054                                    mov     #op.onl,cmdpak+10       ;load opcode
1055                                    clr     cmdpak+12               ;load modifiers
1056                                    clr     cmdpak+14               ;reserved
1057                                    clr     cmdpak+16               ;flags
1058                                    clr     cmdpak+20
1059                                    clr     cmdpak+22
1060                                    clr     cmdpak+24
1061                                    clr     cmdpak+26
1062                                    clr     cmdpa +30
1063                                    clr     cmdpak+32
1064                                    clr     cmdpak+34               ;use default tuning parameters
1065                                    clr     cmdpak+36
1066
1067                                    mov     #RFD'B,@vector          ;NEW VECTOR PLACE
1068                                    mov     #rsppak,rsprng          ;load response packet area into ring
1069                                    mov     #cmdpak,cmdrng          ;load command packet area into r'ng
1070                                    mov     #140000,rsprng+2        ;PORT OWNERSHIP BIT.
1071                                    mov     #b't15,cmdrng+2
1072                                    jsr     pc,POLLWT               ;GO TO POLL AND WAIT ROUTINE.
1073                            ;*******************************************************************
1074                                RFD'B:                              ;INTR TO HERE.
1075                                    add     #6,sp                   ;fix stack for interrupt (4),
1076                                                                    ;pollwt subrtn (2)
1077                                    mov     rsppak+44,MAXLLBN       ;save low word of Max Available LBNs
1078                                    mov     rsppak+46,MAXHLBN       ;save high word of Max Available LBNs
1079                                    sub     #1,maxllbn              ;get max lbn versus size
1080                                    sbc     maxhlbn                 ;
1081                                    mov     #intsrv,@vector         ;CHANGE VECTOR
```

Macro Definitions

```
1082                                    jsr     pc,RSPCHK       ;Go to routine that will check on
1083                                                            ;the response recvd from the mut.
1084                                                            ;it will check the cmd ref
1085                                                            ;num, the endcode, and status.
1086                                    .nlist
1087                            .ENDM
```

Macro Definitions

```
1089
1090
1091                                    ;+
1092                                    ;       Execute an MSCP READ command and check the response
1093                                    ;-
1094
1095
1096                                    .MACRO  READ            ;Execute an MSCP READ command
1097                                    B=B+1                   ;increment the CRN number
1098                                    readtmp \B              ;Call variable B as if it were a number (\)
1099                                    .ENDM
1100
1101                                    .MACRO  READTMP B       ;UNIT carries the Unit Number, LOLBN carries
1102                                                            ;the low word of lbn, and HILBN carries the high
1103                                                            ;word of lbn
1104                                            .list
1105                                    READ'B: bit     #bit15,cmdrng+2         ;test ownership of ring to make sure
1106                                                                            ;we own it
1107                                            bne     READ'B                  ;if we don't, wait until we do
1108                                            mov     #40,cmdlen              ;load length of packet to be sent
1109                                            movb    #0,cmdlen+2             ;load message type and credit value
1110                                            movb    #MSCP.id,cmdlen+3       ;load MSCP connection ID
1111                                            inc     cmdpak                  ;load new CRN
1112                                            clr     cmdpak+2
1113                                            mov     UNIT,cmdpak+4           ;unit number
1114                                            clr     cmdpak+6
1115                                            mov     #op.RD,cmdpak+10        ;load opcode
1116                                            clr     cmdpak+12               ;load modifiers
1117                                            mov     BYTSIZ,cmdpak+14        ;byte count
1118                                            clr     cmdpak+16
1119                                            mov     #RCVBUF,cmdpak+20       ;address of buffer
1120                                            clr     cmdpak+22
1121                                            clr     cmdpak+24
1122                                            clr     cmdpak+26
1123                                            clr     cmdpak+30
1124                                            clr     cmdpak+32
1125                                            mov     LOLBN,cmdpak+34         ;lo word of lbn
1126                                            mov     HILBN,cmdpak+36         ;hi word of lbn
1127
1128                                            mov     #RFD B,@vector          ;NEW VECTOR PLACE
1129                                            mov     #rsppak,rsprng          ;load response packet area into ring
1130                                            mov     #cmdpak,cmdrng          ;load command packet area into ring
1131                                            mov     #140000,rsprng+2        ;PORT OWNERSHIP BIT.
1132                                            mov     #bit15,cmdrng+2
1133                                            jsr     pc,POLLWT               ;GO TO POLL AND WAIT ROUTINE.
1134                                    ;*******************************************************************
1135                                        RFD B:                              ;INTR TO HERE.
1136                                            add     #6,sp                   ;fix stack for interrupt (4),
1137                                                                            ;pollwt subrtn (2)
1138                                            mov     #intsrv,@vector         ;CHANGE VECTOR
1139                                            jsr     pc,RSPCHK               ;Go to routine that will check on
1140                                                                            ;the response recvd from the mut.
1141                                                                            ;it will check the cmd ref
1142                                                                            ;num, the endcode, and status.
1143                                            .nlist
1144                                    .ENDM
```

Macro Definitions

```
1146
1147
1148                                    ;+
1149                                    ;       Execute an MSCP WRITE command and check the response
1150                                    ;-
1151
1152
1153                                    .MACRO  WRITE           ;Execute an MSCP WRITE command
1154                                    B=B+1                   ;increment the CRN number
1155                                    wrttmp  \B              ;Call variables B, C, and D as if they are numbers (\)
1156                                    .ENDM
1157
1158                                    .MACRO  WRTTMP  B
1159                                    .list
1160                                    WRT B:  bit     #bit15,cmdrng+2         ;test ownership of ring to make sure
1161                                                                            ;we own it
1162                                            bne     WRTB                    ;if we don't, wait until we do
1163                                            mov     #40,cmdlen              ;load length of packet to be sent
1164                                            movb    #0,cmdlen+2             ;load message type and credit value
1165                                            movb    #MSCP.id,cmdlen+3       ;load MSCP connection ID
1166                                            inc     cmdpak                  ;load new CRN
1167                                            clr     cmdpak+2
1168                                            mov     UNIT,cmdpak+4           ;unit number
1169                                            clr     cmdpak+6
1170                                            mov     #op.wr,cmdpak+10        ;load opcode
1171                                            clr     cmdpak+12               ;load modifiers
1172                                            mov     BYTSIZ,cmdpak+14        ;byte count
1173                                            clr     cmdpak+16
1174                                            mov     #SNDBUF,cmdpak+20       ;address of buffer
1175                                            clr     cmdpak+22
1176                                            clr     cmdpak+24
1177                                            clr     cmdpak+26
1178                                            clr     cmdpak+30
1179                                            clr     cmdpak+32
1180                                            mov     LOLBN,cmdpak+34         ;low word of lbn
1181                                            mov     HILBN,cmdpak+36         ;high word of lbn
1182
1183                                            mov     #RFDB,@vector           ;NEW VECTOR PLACE
1184                                            mov     #rsppak,rsprng          ;load response packet area into ring
1185                                            mov     #cmdpak,cmdrng          ;load command packet area into ring
1186                                            mov     #140000,rsprng+2        ;PORT OWNERSHIP BIT.
1187                                            mov     #bit15,cmdrng+2
1188                                            jsr     pc,POLLWT               ;GO TO POLL AND WAIT ROUTINE.
1189                                    ;*******************************************************************************
1190                                        RFDB:                               ;INTR TO HERE.
1191                                            add     #6,sp                   ;fix stack for interrupt (4),
1192                                                                            ;pollwt subrtn (2)
1193                                            mov     #intsrv,@vector         ;CHANGE VECTOR
1194                                            jsr     pc,RSPCHK               ;Go to routine that will check on
1195                                                                            ;the response recvd from the mut.
1196                                                                            ;it will check the cmd ref
1197                                                                            ;num, the endcode, and status.
1198                                            .nlist
1199                                    .ENDM
```

Macro Definitions

```
1201
1202
1203                            .MACRO  CMPR                    ;This macro will read the data written onto
1204                                                            ;the disk, store it in RCVBUF, compare the data
1205                                                            ;with the data in SNDBUF, and report all
1206                                                            ;discrepancies as bad bytes in the logical block
1207
1208                            B=B+1
1209                            cmprtmp \B
1210                            .ENDM
1211
1212                            .MACRO  CMPRTMP B
1213                                    .list
1214                                    clr     r1              ;make sure bits 8-15 are zero in r1 and r2
1215                                    clr     r2
1216                                    clr     LOLBN           ;Clear low and high words of LBN counter
1217                                    clr     HILBN
1218                                    clr     ERRCNT          ;Clear cumulative error counter
1219                                    clr     TRKCNT          ;Clear track counter
1220                            NUTRK'B:
1221                                    clr     r0              ;Set offset = 0
1222                                    clr     r3              ;Clear bad byte counter
1223                                    WRITE                   ;Send data from SNDBUF to disk
1224                                    READ                    ;Get data from disk and place it in RCVBUF
1225
1226                            CMP'B:  cmpb    RCVBUF(r0),SNDBUF(r0)
1227                                                            ;Is the data in SNDBUF equal to data in RCVBUF?
1228                                    beq     UPDT'B          ;If so, skip bad byte counter update
1229                                    inc     r3              ;Update bad byte counter
1230                            UPDT'B: inc     r0              ;Increment offset
1231                                    cmp     BYTSIZ,r0
1232                                    bne     CMP'B           ;If not at the end of buffers, compare next byte
1233                                    tst     r3
1234                                    beq     CNTR'B          ;Branch over Bad Byte Report if none found
1235                            .nlist
1236                                    printb  #BTRPT,TRKCNT,r3
1237                                                            ;XXX bad bytes found in LBN:  YYYYYY ZZZZZZ
1238                            .list
1239
1240                            CNTR'B: add     TRKSIZ,LOLBN    ;Update track counters
1241                                    adc     HILBN           ;Add carry from LOLBN to HILBN
1242
1243                            OVER'B: cmp     HILBN,MAXHLBN   ;If high word of LBN <> Maximum high word
1244                                                            ;of LBN, update counters
1245                                    bne     JMP'B
1246                                    cmp     LOLBN,MAXLLBN   ;If high word of LBN = maximum high word
1247                                                            ;of LBN and low word
1248                                                            ;of LBN <= Maximum low word of LBN,
1249                                                            ;go to next block
1250                                    bge     END'B
1251                                    cmp     #0,r3           ;Check to see if any bad bytes found
1252                                    beq     JMP'B           ;If none, go to next track
1253                                    inc     ERRCNT          ;Otherwise, update error count
```

F3

Macro Definitions

```
1254                            JMP'B:  inc     TRKCNT          ;Update track counter
1255                                    jmp     NUTRK'B         ;Go to next track
1256                            .nlist
1257                            END'B:
1258                            .ENDM
```

```
                                .sbttl  Word & Buffer defintions

                        LOGUNIT: .WORD                  ;logunit number
                        LOCAL:   .WORD                  ;
                        PLOC:    .WORD                  ;p table address
                        ptbl:    .WORD                  ;p table address
                        UITadr:  .word
                        BOOT:    .word                  ;bootable media

                        ;+
                        ; These next locations may be altered to supply the correct IP & SA address
                        ; If only 1 jumper is to be placed on the MUT the locations should be filled
                        ;  with addresses 177770 and 177772 respectively.
                        ;-

                        IPreg:   .WORD   0              ;Address of the SA and IP registers
                        Vector:  .word   0
                        Unit:    .word   0              ;unit number
                                 .word   123
                        sernbr:  .word   177777         ;serial number
                        UNTflgs: .word   0              ;flags, bit 15 = auto mode
                                                        ;bit 13 = unknown model number
                                                        ;bit 12 = test floppy only
                                                        ;bit 11 = Format Progress Report title has
                                                        ;         already been printed
                        mdlnbr:  .word   0              ;model number of the controller as returned in
                                                        ;step 4
                        mcdnbr:  .word   0              ;microcode number of the controller as returned
                                                        ;in step 4
                        UIN:     .word   0              ;this is a pointer to the correct UIT table

                        RSP1:    .BLKW   2              ;Response packet length
                        RSPPAK:  .BLKW   30.            ;Response packet
                        CMDLEN:  .BLKW   2              ;Command packet length
                        CMDPAK:  .BLKW   20.            ;Command packet

                        CINTR:   .WORD   0              ;Command interrupt indicator
                        RINTR:   .WORD   0              ;Response interrupt indicator
                        RSPRNG:  .word   rsppak         ;Message ring
                                 .word   140000
                        CMDRNG:  .word   cmdpak         ;Command ring
                                 .word   100000
                                 .WORD    1

                        LSTCRN:  .word   0              ;storage for unreturned command CRN
                        LSTCMD:  .word   0              ;storage for unreturned command opcode
                        LSTVCT:  .word   0              ;storage for unreturned command interrupt
                                                        ;vector address
                        LOPRGI:  .word   0              ;Low word of the progress indicator
                        HIPRGI:  .word   0              ;High word of progress indicator
                        LOLBN:   .word   0              ;Low word of Logical Block Number
                                                        ;(MSCP Read/Write Commands)       GJK
                        HILBN:   .word   0              ;High word of Logical Block Number
                                                        ;(MSCP Read/Write Commands)       GJK
                        MAXLLBN:         .word 0        ;Low word of long word containing number of
                                                        ;LBNs available - GJK
                        MAXHLBN:         .word 0        ;High word of long word containing number of
```

Word & Buffer defintions

```
1317                                                                    ;LBNs available - GJK
1318
1319 002562  000000                BYTSIZ: .word   0                   ;(# of LBNs per track) * (# of bytes per LBN)
1320 002564  000000                TRKSIZ: .word   0                   ;# of lbns on a track
1321 002566  000000                ERRCNT: .word   0                   ;Word used to keep track of number of bad
1322                                                                    ;blocks found           - GJK
1323 002570  000000                TRKCNT: .word   0                   ;Track counter
1324 002572  000000                ENDIT:  .word   0                   ;Storage for GMANIL in tstdrv routine
1325 002574  000000                DELAY:  .word   0                   ;Storage for delay in TRP100
1326 002576  000001                NXTTIM: .word   1                   ;Used to keep track of one second delays
1327
1328                               ; L'ne time clock var'ables (Used in HRDINT routine)
1329
1330 002600                        TIMER:
1331 002600                                .blkw   1
1332 002602                        TIMEOUT:
1333 002602                                .blkw   1
1334 002604                        HERZ:
1335 002604  007020                        .word   3600.
1336 002606                        RECV.DONE:
1337 002606                                .blkw   1
1338
1339         .nlist bin                                 ;data area
1340 002610  DATARE: .asciz  /%A12345678901234567890123456789012345678901234567890123456789012345678901234567890/
1341         .even
1342
1343 002734  PRGnam: .ascii  /FORMAT/          ;address of local format program name
1344 002742          .byte   0                 ;null for asciz
1345 002743  XBN:    .ASCIZ  /0123456789/
1346 002756  DBN:    .ASCIZ  /0123456789/
1347 002771  LBN:    .ASCIZ  /0123456789/
1348 003004  RBN:    .ASCIZ  /0123456789/
1349
1350         .even
1351                                       .l'st bin
```

```
                                .sbttl  DISK UNIT INFORMATION TABLES
                                ;+
                                ; The following tables are made up of disk drive parameters which will be
                                ; fed to the FORMAT controller local program which will then use the
                                ; information to format the drives.
                                ;-
        002776                  .=2776
002776  177777                      .word       1               ;back door for custom table build
        003000                  .=3000

                                ;+
                                ;       Unit Information table RD51 Seagate
                                ;-

003000                          SNDBUF:                         ;Use UITs as data sent to disk to test the
                                                                ;integrity of the LBNs
003000                          UIT0:
                                                                ;/*Top of Unit Information table (UIT)
003000  000071                          .word   57.             ;/XBN size (lo wrd)
                                                                ;XBN size = 3*(1+sectors_per_track)/
003002  000000                          .word   0               ;/XBNs size (hi wrd)/
003004  000127                          .word   87.             ;/DBN size (lo wrd)/
003006  000000                          .word   0               ;/DBN size (hi wrd)/
003010  052360                          .word   21744.          ;/LBN size (lo wrd)/
003012  000000                          .word   0               ;/LBN size (hi wrd)/
003014  000220                          .word   144.            ;/RBN size (lo wrd)/
003016  000000                          .word   0               ;/RBN size (hi wrd)/
003020  000022                          .word   18.             ;/Sectors per track/
003022  000004                          .word   4.              ;/Surfaces per unit/
003024  000463                          .word   307.            ;/Cylinders per unit/
003026  000156                          .word   110.            ;/Write precomp cylinder/
003030  000462                          .word   306.            ;/Reduce write current cylinder /
003032  001006                          .word   518.            ;/Drive Type/
003034  000001                          .word   1               ;/Use CRC or ECC/
003036  000044                          .word   36.             ;/RCT Size/
003040  000004                          .word   4.              ;/Number of RCT copies/
003042  040063                          .word   ^B0100000000110011 ;^H4033;/Media (lo wrd)/
003044  022544                          .word   ^B0010010101100100 ;^H2564;/Media (hi wrd)/
003046  000002                          .word   2               ;/Sector Interleave (n-to-1)/
003050  000002                          .word   2               ;/Surface to Surface Skew/
003052  000001                          .word   1               ;/Cylinder to Cylinder Skew/
003054  000020                          .word   16.             ;/Gap size 0/
003056  000020                          .word   16.             ;/Gap size 1/
003060  000005                          .word   5.              ;/Gap size 2/
003062  000020                          .word   16.             ;/Gap size 3/
003064  000015                          .word   13.             ;/Sync size/
003066  000001                          .word   1               ;/MSCP cylinders per Unit/
003070  000001                          .word   1               ;/MSCP Groups per Cylinder/
003072  000001                          .word   1               ;/MSCP Tracks per Group/
003074  000002                          .word   2               ;/Max allowed bad spots per surface/
003076  000151                          .word   105.            ;/Bad spot tolerance (bytes)/
003100  000463                          .word   307.            ;/auto recal cylinder
003102  000463                          .word   307.            ;/auto recal cylinder
        000104                  UITsiz = .-UIT0
        003104                  .=3000+UITsiz
```

J3

DISK UNIT INFORMATION TABLES

```
                                ;+
                                ;       Unit Information table    RD52 Quantum drive
                                ;-


003104                          UIT1:
                                                                ;/*Top of Unit Information table (UIT)
003104  000066                          .word   54.             ;/XBN size (lo wrd)
                                                                ;XBN size = 3*(1+sectors_per_track)/
003106  000000                          .word   0               ;/XBN size (hi wrd)/
003110  000122                          .word   82.             ;/DBN size (lo wrd)/
003112  000000                          .word   0               ;/DBN size (hi wrd)/
003114  166140                          .word   60512.          ;/LBN size (lo wrd)/
003116  000000                          .word   0               ;/LBN size (hi wrd)/
003120  000250                          .word   168.            ;/RBN size (lo wrd)/
003122  000000                          .word   0               ;/RBN size (hi wrd)/
003124  000021                          .word   17.             ;/Sectors per track/
003126  000010                          .word   8.              ;/Surfaces per unit/
003130  001000                          .word   512.            ;/Cylinders per unit/
003132  000400                          .word   256.            ;/Write precomp cylinder/
003134  001000                          .word   512.            ;/Reduce write current cylinder /
003136  001010                          .word   520.            ;/Drive Type/
003140  000001                          .word   1               ;/Use CRC or ECC/
003142  000004                          .word   4               ;/RCT Size/
003144  000010                          .word   8.              ;/Number of RCT copies/
003146  040064                          .word   ^B0100000000110100 ;^H4034;/Media (lo wrd)/
003150  022544                          .word   ^B0010010001100100 ;^H2564;/Media (hi wrd)/
003152  000001                          .word   1               ;/Sector Interleave (n-to-1)/
003154  000002                          .word   2               ;/Surface to Surface Skew/
003156  000015                          .word   13.             ;/Cylinder to Cylinder Skew/
003160  000020                          .word   16.             ;/Gap size 0/
003162  000020                          .word   16.             ;/Gap size 1/
003164  000005                          .word   5.              ;/Gap size 2/
003166  000050                          .word   40.             ;/Gap size 3/
003170  000015                          .word   13.             ;/Sync size/
003172  000001                          .word   1               ;/MSCP cylinders per Unit/
003174  000001                          .word   1               ;/MSCP Groups per Cylinder/
003176  000001                          .word   1               ;/MSCP Tracks per Group/
003200  000012                          .word   10.             ;/Max allowed bad spots per surface/
003202  000151                          .word   105.            ;/Bad spot tolerance (bytes)/
003204  001000                          .word   512.            ;/auto recal cylinder
003206  001000                          .word   512.            ;/auto recal cylinder

003210                          .=3000+UIT1+UIT2

                                ;+
                                ;       Unit Information table  RD52 Gtasi
                                ;-


003210                          UIT2:
                                                                ;/*Top of Unit Information table (UIT)
003210  000066                          .word   54.             ;/XBN size (lo wrd)
                                                                ;XBN size = 3*(1+sectors_per_track)/
003212  000000                          .word   0               ;/XBN size (hi wrd)/
```

K3

DISK UNIT INFORMATION TABLES

```
1467 003214  000001                .word   65.             ;/DBN size (lo wrd)/
1468 003216  000000                .word   0               ;/DBN size (hi wrd)/
1469 003220  166140                .word   60512.          ;/LBN size (lo wrd)/
1470 003222  000000                .word   0               ;/LBN size (hi wrd)/
1471 003224  000250                .word   168.            ;/RBN size (lo wrd)/
1472 003226  000000                .word   0               ;/RBN size (hi wrd)/
1473 003230  000021                .word   17.             ;/Sectors per track/
1474 003232  000007                .word   7.              ;/Surfaces per unit/
1475 003234  001205                .word   645.            ;/Cylinders per unit/
1476 003236  000500                .word   320.            ;/Write precomp cylinder/
1477 003240  001205                .word   645.            ;/Reduce write current cylinder /
1478 003242  001010                .word   520.            ;/Drive Type/
1479 003244  000001                .word   1               ;/Use CRC or ECC/
1480 003246  000004                .word   4               ;/RCT Size/
1481 003250  000010                .word   8.              ;/Number of RCT copies/
1482 003252  040064                .word   ^B0100000000110100 ;^H4034;/Media (lo wrd)/
1483 003254  022544                .word   ^B0010010101100100 ;^H2564;/Media (hi wrd)/
1484 003256  000001                .word   1               ;/Sector Interleave (n-to-1)/
1485 003260  000002                .word   2               ;/Surface to Surface Skew/
1486 003262  000007                .word   7.              ;/Cylinder to Cylinder Skew/
1487 003264  000020                .word   16.             ;/Gap size 0/
1488 003266  000020                .word   16.             ;/Gap size 1/
1489 003270  000005                .word   5.              ;/Gap size 2/
1490 003272  000050                .word   40.             ;/Gap size 3/
1491 003274  000015                .word   13.             ;/Sync size/
1492 003276  000001                .word   1               ;/MSCP cylinders per Unit/
1493 003300  000001                .word   1               ;/MSCP Groups per Cylinder/
1494 003302  000001                .word   1               ;/MSCP Tracks per Group/
1495 003304  000024                .word   20.             ;/Max allowed bad spots per surface/
1496 003306  000151                .word   105.            ;/Bad spot tolerance (bytes)/
1497 003310  001206                .word   646.            ;/auto recal cylinder
1498 003312  001206                .word   646.            ;/auto recal cylinder
1499
1500         003314        .=3000+UITsiz+UITsiz+UITsiz
1501
1502                       ;+
1503                       ;       Unit Information table  RD53 Micropolis
1504                       ;-
1505
1506
1507 003314                UIT3:
1508                                                       ;/^Top of Unit Information table (UIT)
1509 003314  000066                .word   54.             ;/XBN size (lo wrd)
1510                                                       ;XBN size = 3^(1+sectors_per_track)/
1511 003316  000000                .word   0               ;/XBN size (hi wrd)/
1512 003320  000122                .word   82.             ;/DBN size (lo wrd)/
1513 003322  000000                .word   0               ;/DBN size (hi wrd)/
1514 003324  016730                .word   7640.           ;/LBN size (lo wrd)/
1515 003326  000002                .word   2.              ;/LBN size (hi wrd)/
1516 003330  000430                .word   280.            ;/RBN size (lo wrd)/
1517 003332  000000                .word   0               ;/RBN size (hi wrd)/
1518 003334  000021                .word   17.             ;/Sectors per track/
1519 003336  000010                .word   8.              ;/Surfaces per unit/
1520 003340  002000                .word   1024.           ;/Cylinders per unit/
1521 003342  002000                .word   1024.           ;/Write precomp cylinder/
1522 003344  002000                .word   1024.           ;/Reduce write current cylinder /
1523 003346  001011                .word   521.            ;/Drive Type/
```

DISK UNIT INFORMATION TABLES

```
1524 003350  000001                        .word   1               ;/Use CRC or ECC/
1525 003352  000005                        .word   5               ;/RCT Size/
1526 003354  000010                        .word   8.              ;/Number of RCT copies/
1527 003356  040065                        .word   ^B0100000000110101 ;^H4035;/Media (lo wrd)/
1528 003360  022544                        .word   ^B0010010101100100 ;^H2564;/Media (hi wrd)/
1529 003362  000001                        .word   1               ;/Sector Interleave (n-to-1)/
1530 003364  000002                        .word   2               ;/Surface to Surface Skew/
1531 003366  000010                        .word   8.              ;/Cylinder to Cylinder Skew/
1532 003370  000020                        .word   16.             ;/Gap size 0/
1533 003372  000020                        .word   16.             ;/Gap size 1/
1534 003374  000005                        .word   5.              ;/Gap size 2/
1535 003376  000050                        .word   40.             ;/Gap size 3/
1536 003400  000015                        .word   13.             ;/Sync size/
1537 003402  000001                        .word   1               ;/MSCP cylinders per Unit/
1538 003404  000001                        .word   1               ;/MSCP Groups per Cylinder/
1539 003406  000001                        .word   1               ;/MSCP Tracks per Group/
1540 003410  000040                        .word   32.             ;/Max allowed bad spots per surface/
1541 003412  000156                        .word   110.            ;/Bad spot tolerance (bytes)/
1542 003414  002000                        .word   1024.           ;/auto recal cylinder
1543 003416  002000                        .word   1024.           ;/auto recal cylinder
1544
1545         003420                .=3000+UITsiz+UITsiz+UITsiz+UITsiz
1546
1547
1548                               ;+
1549                               ;       Unit Information table  RD31 Seagate
1550                               ;-
1551
1552
1553 003420                        UIT4:
1554                                                               ;/*Top of Unit Information table (UIT)
1555 003420  000066                        .word   54.             ;/XBN size (lo wrd)
1556                                                               ;XBN size = 3*(1+sectors_per_track)/
1557 003422  000000                        .word   0               ;/XBN size (hi wrd)/
1558 003424  000016                        .word   14.             ;/DBN size (lo wrd)/
1559 003426  000000                        .word   0               ;/DBN size (hi wrd)/
1560 003430  121160                        .word   41584.          ;/LBN size (lo wrd)/
1561 003432  000000                        .word   0               ;/LBN size (hi wrd)/
1562 003434  000144                        .word   100.            ;/RBN size (lo wrd)/
1563 003436  000000                        .word   0               ;/RBN size (hi wrd)/
1564 003440  000021                        .word   17.             ;/Sectors per track/
1565 003442  000004                        .word   4.              ;/Surfaces per unit/
1566 003444  001147                        .word   615.            ;/Cylinders per unit/
1567 003446  000400                        .word   256.            ;/Write precomp cylinder/
1568 003450  001147                        .word   615.            ;/Reduce write current cylinder /
1569 003452  001014                        .word   524.            ;/Drive Type/
1570 003454  000001                        .word   1               ;/Use CRC or ECC/
1571 003456  000003                        .word   3               ;/RCT Size/
1572 003460  000010                        .word   8.              ;/Number of RCT copies/
1573 003462  040037                        .word   ^B0100000000011111 ;^H401F;/Media (lo wrd)/
1574 003464  022544                        .word   ^B0010010101100100 ;^H2564;/Media (hi wrd)/
1575 003466  000001                        .word   1               ;/Sector Interleave (n-to-1)/
1576 003470  000002                        .word   2               ;/Surface to Surface Skew/
1577 003472  000004                        .word   4.              ;/Cylinder to Cylinder Skew/
1578 003474  000020                        .word   16.             ;/Gap size 0/
1579 003476  000020                        .word   16.             ;/Gap size 1/
1580 003500  000005                        .word   5.              ;/Gap size 2/
```

DISK UNIT INFORMATION TABLES

```
1581 003502  000050                        .word   40.             ;/Gap size 3/
1582 003504  000015                        .word   13.             ;/Sync size/
1583 003506  000001                        .word   1               ;/MSCP cylinders per Unit/
1584 003510  000001                        .word   1               ;/MSCP Groups per Cylinder/
1585 003512  000001                        .word   1               ;/MSCP Tracks per Group/
1586 003514  000010                        .word   8.              ;/Max allowed bad spots per surface/
1587 003516  000151                        .word   105.            ;/Bad spot tolerance (bytes)/
1588 003520  001147                        .word   615.            ;/auto recal cylinder
1589 003522  001150                        .word   616.            ;/auto recal cylinder
1590
1591         003524                .=3000+UITsiz+UITsiz+UITsiz+UITsiz+UITsiz
1592
1593
1594                               ;+
1595                               ;       Unit Information table  RD54 Maxtor Drive
1596                               ;-
1597
1598
1599 003524                       UIT5:
1600                                                               ;/*Top of Unit Information table (UIT)
1601 003524  000066                        .word   54.             ;/XBN size (lo wrd)
1602                                                               ;XBN size = 3*(1+sectors_per_track)/
1603 003526  000000                        .word   0               ;/XBN size (hi wrd)/
1604 003530  000311                        .word   201.            ;/DBN size (lo wrd)/
1605 003532  000000                        .word   0               ;/DBN size (hi wrd)/
1606 003534  137730                        .word   137730          ;/LBN size (lo wrd)/
1607 003536  000004                        .word   4               ;/LBN size (hi wrd)/
1608 003540  001141                        .word   609.            ;/RBN size (lo wrd)/
1609 003542  000000                        .word   0               ;/RBN size (hi wrd)/
1610 003544  000021                        .word   17.             ;/Sectors per track/
1611 003546  000017                        .word   15.             ;/Surfaces per unit/
1612 003550  002311                        .word   1225.           ;/Cylinders per unit/
1613 003552  002311                        .word   1225.           ;/Write precomp cylinder/
1614 003554  002311                        .word   1225.           ;/Reduce write current cylinder /
1615 003556  001015                        .word   525.            ;/Drive Type/
1616 003560  000001                        .word   1               ;/Use CRC or ECC/
1617 003562  000007                        .word   7               ;/RCT Size/
1618 003564  000010                        .word   8.              ;/Number of RCT copies/
1619 003566  040066                        .word   ^B0100000000110110 ;^H4036;/Media (lo wrd)/
1620 003570  022544                        .word   ^B0010010101100100 ;^H2564;/Media (hi wrd)/
1621 003572  000001                        .word   1               ;/Sector Interleave (n-to-1)/
1622 003574  000002                        .word   2               ;/Surface to Surface Skew/
1623 003576  000010                        .word   8.              ;/Cylinder to Cylinder Skew/
1624 003600  000020                        .word   16.             ;/Gap size 0/
1625 003602  000020                        .word   16.             ;/Gap size 1/
1626 003604  000005                        .word   5.              ;/Gap size 2/
1627 003606  000050                        .word   40.             ;/Gap size 3/
1628 003610  000015                        .word   13.             ;/Sync size/
1629 003612  000001                        .word   1               ;/MSCP cylinders per Unit/
1630 003614  000001                        .word   1               ;/MSCP Groups per Cylinder/
1631 003616  000001                        .word   1               ;/MSCP Tracks per Group/
1632 003620  000040                        .word   32.             ;/Max allowed bad spots per surface/
1633 003622  000151                        .word   105.            ;/Bad spot tolerance (bytes)/
1634 003624  002311                        .word   1225.           ;/auto recal cylinder
1635 003626  002312                        .word   1226.           ;/auto recal cylinder possible on this vendor's
1636                                                               ;/drive mmm
1637
```

DISK UNIT INFORMATION TABLES

```
1638          003630                  .=3000+UITsiz+UITsiz+UITsiz+UITsiz+UITsiz+UITsiz
1639
1640
1641                                  ;+
1642                                  ;       Unit Information table  RD32
1643                                  ;-
1644
1645
1646 003630                           UIT6:
1647                                                                  ;/*Top of Unit Information table (UIT)
1648 003630  000066                           .word   54.             ;/XBN size (lo wrd)
1649                                                                  ;XBN size = 3*(1+sectors_per_track)/
1650 003632  000000                           .word   0               ;/XBN size (hi wrd)/
1651 003634  000060                           .word   48.             ;/DBN size (lo wrd)/
1652 003636  000000                           .word   0               ;/DBN size (hi wrd)/
1653 003640  042444                           .word   042444          ;/LBN size (lo wrd)/
1654 003642  000001                           .word   1               ;/LBN size (hi wrd)/
1655 003644  000310                           .word   200.            ;/RBN size (lo wrd)/
1656 003646  000000                           .word   0               ;/RBN size (hi wrd)/
1657 003650  000021                           .word   17.             ;/Sectors per track/
1658 003652  000006                           .word   6.              ;/Surfaces per unit/
1659 003654  001464                           .word   820.            ;/Cylinders per unit/
1660 003656  001464                           .word   820.            ;/Write precomp cylinder/
1661 003660  001464                           .word   820.            ;/Reduce write current cylinder /
1662 003662  001017                           .word   527.            ;/Drive Type/
1663 003664  000001                           .word   1               ;/Use CRC or ECC/
1664 003666  000004                           .word   4               ;/RCT Size/
1665 003670  000010                           .word   8.              ;/Number of RCT copies/
1666 003672  040040                           .word   ^B0100000000100000 ;^H40040;/Media (lo wrd)/
1667 003674  022544                           .word   ^B0010010101100100 ;^H22544;/Media (hi wrd)/
1668 003676  000001                           .word   1               ;/Sector Interleave (n-to-1)/
1669 003700  000002                           .word   2               ;/Surface to Surface Skew/
1670 003702  000011                           .word   9.              ;/Cylinder to Cylinder Skew/
1671 003704  000020                           .word   16.             ;/Gap size 0/
1672 003706  000020                           .word   16.             ;/Gap size 1/
1673 003710  000005                           .word   5.              ;/Gap size 2/
1674 003712  000050                           .word   40.             ;/Gap size 3/
1675 003714  000015                           .word   13.             ;/Sync size/
1676 003716  000001                           .word   1               ;/MSCP cylinders per Unit/
1677 003720  000001                           .word   1               ;/MSCP Groups per Cylinder/
1678 003722  000001                           .word   1               ;/MSCP Tracks per Group/
1679 003724  000010                           .word   8.              ;/Max allowed bad spots per surface/
1680 003726  000151                           .word   105.            ;/Bad spot tolerance (bytes)/
1681 003730  001465                           .word   821.            ;/auto recal cylinder
1682 003732  001465                           .word   821.            ;/auto recal cylinder
1683
1684          003734                  .=3000+UITsiz+UITsiz+UITsiz+UITsiz+UITsiz+UITsiz+UITsiz
1685
1686
1687                                  ;+
1688                                  ;       Unit Information table
1689                                  ;-
1690
1691
1692 003734                           UIT7:
1693                                                                  ;/*Top of Unit Information table (UIT)
1694 003734  000066                           .word   54.             ;/XBN size (lo wrd)
```

DISK UNIT INFORMATION TABLES

```
1695                                                    ;XBN size = 3*(1+sectors_per_trac
1696 003736  000000                        .word   0   ; XBN size (hi wrd)/
1697 003740  000057                        .word   47. ;/DBN size (lo wrd)/
1698 003742  000000                        .word   0   ;/DBN size (hi wrd)/
1699 003744  016677                        .word   016677      ;/LBN size (lo wrd)/
1700 003746  000002                        .word   2           ;/LBN size (hi wrd)/
1701 003750  000524                        .word   340.        ;/RBN size (lo wrd)/
1702 003752  000000                        .word   0           ;/RBN size (hi wrd)/
1703 003754  000021                        .word   17.         ;/Sectors per track/
1704 003756  000010                        .word   8.          ;/Surfaces per unit/
1705 003760  002000                        .word   1024.       ;/Cylinders per unit/
1706 003762  002000                        .word   1024.       ;/Write precomp cylinder/
1707 003764  002000                        .word   1024.       ;/Reduce write current cylinder /
1708 003766  000000                        .word   0           ;/Drive Type/
1709 003770  000001                        .word   1           ;/Use CRC or ECC/
1710 003772  000005                        .word   5           ;/RCT Size/
1711 003774  000003                        .word   3           ;/Number of RCT copies/
1712 003776  040065                        .word   ^B0100000000110101 ;^H4035;/Media (lo wrd)/
1713 004000  022544                        .word   ^B0010010101100100 ;^H2564;/Media (hi wrd)/
1714 004002  000001                        .word   1           ;/Sector Interleave (n-to-1)/
1715 004004  000002                        .word   2           ;/Surface to Surface Skew/
1716 004006  000010                        .word   8.          ;/Cylinder to Cylinder Skew/
1717 004010  000020                        .word   16.         ;/Gap size 0/
1718 004012  000020                        .word   16.         ;/Gap size 1/
1719 004014  000005                        .word   5.          ;/Gap size 2/
1720 004016  000050                        .word   40.         ;/Gap size 3/
1721 004020  000015                        .word   13.         ;/Sync size/
1722 004022  000001                        .word   1           ;/MSCP cylinders per Unit/
1723 004024  000001                        .word   1           ;/MSCP Groups per Cylinder/
1724 004026  000001                        .word   1           ;/MSCP Tracks per Group/
1725 004030  000040                        .word   32.         ;/Max allowed bad spots per surface/
1726 004032  000156                        .word   110.        ;/Bad spot tolerance (bytes)/
1727 004034  002000                        .word   1024.       ;/auto recal cylinder
1728 004036  002000                        .word   1024.       ;/auto recal cylinder
1729
1730         004040             .=3000+UITsiz+UITsiz+UITsiz+UITsiz+UITsiz+UITsiz+UITsiz+UITsiz
1731
1732
1733                            ;+
1734                            ;       DEFAULT Unit Information table
1735                            ;-
1736
1737
1738 004040                     UITdf:
1739                                                    ;/*Top of Unit Information table (UIT)
1740 004040  000066                        .word   54.         ;/XBN size (lo wrd)
1741                                                    ;XBN size = 3*(1+sectors_per_track)/
1742 004042  000000                        .word   0           ;/XBN size (hi wrd)/
1743 004044  000311                        .word   201.        ;/DBN size (lo wrd)/
1744 004046  000000                        .word   0           ;/DBN size (hi wrd)/
1745 004050  137710                        .word   137710      ;/LBN size (lo wrd)/
1746 004052  000004                        .word   4           ;/LBN size (hi wrd)/
1747 004054  001161                        .word   625.        ;/RBN size (lo wrd)/
1748 004056  000000                        .word   0           ;/RBN size (hi wrd)/
1749 004060  000021                        .word   17.         ;/Sectors per track/
1750 004062  000017                        .word   15.         ;/Surfaces per unit/
1751 004064  002311                        .word   1225.       ;/Cylinders per unit/
```

DISK UNIT INFORMATION TABLES

```
1752 004066  002311                .word   1225.           ;/Write precomp cylinder/
1753 004070  002311                .word   1225.           ;/Reduce write current cylinder /
1754 004072  000000                .word   0               ;/Drive Type/
1755 004074  000001                .word   1               ;/Use CRC or ECC/
1756 004076  000007                .word   7               ;/RCT Size/
1757 004100  000010                .word   8.              ;/Number of RCT copies/
1758 004102  040066                .word   ^B0100000000110110 ;^H4036;/Media (lo wrd)/
175? 004104  022544                .word   ^B0010010101100100 ;^H2564;/Media (hi wrd)/
1760 004106  000001                .word   1               ;/Sector Interleave (n-to-1)/
1761 004110  000002                .word   2               ;/Surface to Surface Skew/
1762 004112  000015                .word   13.             ;/Cylinder to Cylinder Skew/
1763 004114  000020                .word   16.             ;/Gap size 0/
1764 004116  000020                .word   16.             ;/Gap size 1/
1765 004120  000005                .word   5.              ;/Gap size 2/
1766 004122  000050                .word   40.             ;/Gap size 3/
1767 004124  000015                .word   13.             ;/Sync size/
1768 004126  000001                .word   1               ;/MSCP cylinders per Unit/
1769 004130  000001                .word   1               ;/MSCP Groups per Cylinder/
1770 004132  000001                .word   1               ;/MSCP Tracks per Group/
1771 004134  000012                .word   10.             ;/Max allowed bad spots per surface/
1772 004136  000151                .word   105.            ;/Bad spot tolerance (bytes)/
1773 004140  002000                .word   1024.           ;/auto recal cylinder
1774 004142  002000                .word   1024.           ;/auto recal cylinder
1775
```

DISK PARAMETER QUESTIONS

```
                                        .sbttl DISK PARAMETER QUESTIONS
                .nlist  bin

                ;+
                ; P table Questions
                ;-

004144  IP.adr:   .ASCIZ  /IP Address/
004157  vec.adr:  .ASCIZ  /Vector Address/
004176  drv.nbr:  .ASCIZ  /Logical Drive (0-255)/
004224  ser.nbr:  .ASCIZ  /Drive Serial Number(1-32000)/
004261  tst.dsk:  .ASCIZ  /Just test floppy/
004302  do.agn:   .ASCIZ  /Test another floppy/
004326  auto.md:  .ASCIZ  /Auto Format Mode/
004347  warning:  .ASCIZ /***** WARNING all the data on this drive will be DESTROYED ***/
004446            .byte  0

004447  do.cont:  .ASCIZ  /Proceed to format the drive/

004503  DrvTxa:  .asciz /%N%AUIT#  Drive Name%N/
004532  DrvTxb:  .asciz /%A-------------------------------------------------%N/
004626  DrvTx0:  .asciz /%A  0   RD51                                        %N/
004722  DrvTx1:  .asciz /%A  1   RD52 part # 30-21721-02 (1 light on front panel) %N/
005016  DrvTx2:  .asciz /%A  2   RD52 part # 30-23227-02 (2 lights on front panel)%N/
005112  DrvTx3:  .asciz /%A  3   RD53                                        %N/
005206  DrvTx4:  .asciz /%A  4   RD31                                        %N/
005302  DrvTx5:  .asciz /%A  5   RD54                                        %N/
005376  DrvTx6:  .asciz /%A  6   RD32                                        %N/
005472  DrvTx7:  .asciz /%A  7                                               %N/
005565  DrvTxc:  .asciz /%A 10                                               %N/
005661  ASMSGr:  .ASCIZ /%A      Unrecognized Drive                          %N/

005755  ASMSG1:  .ASCII  /%N%AAUTOSIZER FOUND:/
006001           .ASCIZ  /%N%AUnt  Cyls  UIT#  Drive Name%N/
006043  ASMSG7:  .ASCIZ  /%A  %D1%A              Nonexistent%N/
006110  ASMSG8:  .ASCIZ  /%A  %D1%A              RX50 Floppy (UNFORMATABLE)%N/
006174  ASMSG9:  .ASCIZ  /%A  %D1%A              RX33 Floppy (FORMATABLE)%N/
006256  ASMSG2:  .ASCIZ  /%A  %D1%A  %D4%A  /
006301  ASMSG3:  .ASCIZ  /%N%AAUTOSIZER RETURNED FAILURE STATUS CODE %D1%A:/
006363  ASMSG4:  .ASCIZ  /%N%A    CONTROLLER CHIP NEVER WENT DONE/
006433  ASMSG5:  .ASCIZ  /%N%A    CONTROLLER CHIP NEVER INTERRUPTED/
006505  ASMSG6:  .ASCIZ  /%N%A    SEEK FAILED/
006531  ASMSGT:  .ASCIZ  /%N/

006534  Unt.nbr:  .ASCIZ  /Enter Unit Identifier Table (UIT)/
006576  ask.prg:  .ASCIZ  /What local program do you want to run/
006644  ask.xbn:  .ASCIZ  /Enter XBN size in decimal (upto 10 digits)/
006717  ask.dbn:  .ASCIZ  /Enter DBN size in decimal (upto 10 digits)/
006772  ask.lbn:  .ASCIZ  /Enter LBN size in decimal (upto 10 digits)/
007045  ask.rbn:  .ASCIZ  /Enter RBN size in decimal (upto 10 digits)/


                ;+
                ;               FORMAT PROGRESS REPORT MESSAGES
                ;-

007120  FRPTB:   .ASCIZ  /%N%A    ------- - -  FORMAT PROGRESS REPORT ------- ---- %N/
```

DISK PARAMETER QUESTIONS

```
1834 007214  FMTTRK: .ASCIZ  /%N%AFormatting tracks, lbn #: /
1835 007253  RPDFCT: .ASCIZ  /%N%AReplacing defect #: %D5%A on head #: %D3/
1836 007330  RDDFCT: .ASCIZ  /%N%AReading defect list/
1837 007360  FCPW:   .ASCIZ  /%N%AFirst check pass, writing lbn #: /
1838 007426  FCPR:   .ASCIZ  /%N%AFirst check pass, reading lbn #: /
1839 007474  SCPW:   .ASCIZ  /%N%ASecond check pass, writing lbn #: /
1840 007543  SCPR:   .ASCIZ  /%N%ASecond check pass, reading lbn #: /
1841 007612  TCPW:   .ASCIZ  /%N%AThird check pass, writing lbn #: /
1842 007660  TCPR:   .ASCIZ  /%N%AThird check pass, reading lbn #: /
1843
1844 007726  bot.dev: .ASCII <15><12>/WARNING - If RX33 remove boot diskette if in drive to be formatted and/
1845 010036           .ASCII <15><12>/                        insert a diskette to be formatted./
1846 010126           .ASCII <15><12>/              If WINCHESTER check if wrt protect switch (off) & ready switch (on)./
1847 010246           .ASCIZ <15><12>/WARNING - All data on drive will be DESTROYED, ?/
1848 010331  bot.rep: .ASCIZ /If boot drive, reinsert boot diskette & press <RETURN>./
1849 010421  bot.con: .ASCIZ <15><12>/Do you want to format another diskette?/
1850
1851         ; Top of Unit Information table (UIT)
1852
1853 010473  TBQ0:   .ASCIZ  /XBN size (lo wrd) XBN size = 3*(1+sectors_per_track)/
1854 010560  TBQ1:   .ASCIZ  /XBN size (hi wrd)/
1855 010602  TBQ2:   .ASCIZ  /DBN size (lo wrd)/
1856 010624  TBQ3:   .ASCIZ  /DBN size (hi wrd)/
1857 010646  TBQ4:   .ASCIZ  /LBN size (lo wrd)/
1858 010670  TBQ5:   .ASCIZ  /LBN size (hi wrd)/
1859 010712  TBQ6:   .ASCIZ  /RBN size (lo wrd)/
1860 010734  TBQ7:   .ASCIZ  /RBN size (hi wrd)/
1861 010756  TBQ8:   .ASCIZ  /Sectors per track/
1862 011000  TBQ9:   .ASCIZ  /Surfaces per unit/
1863 011022  TBQ10:  .ASCIZ  /Cylinders per unit/
1864 011045  TBQ11:  .ASCIZ  /Write precomp cylinder/
1865 011074  TBQ12:  .ASCIZ  /Reduce write current cylinder /
1866 011133  TBQ13:  .ASCIZ  /Drive Type/
1867 011146  TBQ14:  .ASCIZ  /Use CRC or ECC/
1868 011165  TBQ15:  .ASCIZ  /RCT Size/
1869 011176  TBQ16:  .ASCIZ  /Number of RCT copies/
1870 011223  TBQ17:  .ASCIZ  /Media (lo wrd)/
1871 011242  TBQ18:  .ASCIZ  /Media (hi wrd)/
1872 011261  TBQ19:  .ASCIZ  /Sector Interleave (n-to-1)/
1873 011314  TBQ20:  .ASCIZ  /Surface to Surface Skew/
1874 011344  TBQ21:  .ASCIZ  /Cylinder to Cylinder Skew/
1875 011376  TBQ22:  .ASCIZ  /Gap size 0/
1876 011411  TBQ23:  .ASCIZ  /Gap size 1/
1877 011424  TBQ24:  .ASCIZ  /Gap size 2/
1878 011437  TBQ25:  .ASCIZ  /Gap size 3/
1879 011452  TBQ26:  .ASCIZ  /Sync size/
1880 011464  TBQ28:  .ASCIZ  /MSCP cylinders per Unit/
1881 011514  TBQ29:  .ASCIZ  /MSCP Groups per Cylinder/
1882 011545  TBQ30:  .ASCIZ  /MSCP Tracks per Group/
1883 011573  TBQ31:  .ASCIZ  /Max allowed bad spots per surface/
1884 011635  TBQ32:  .ASCIZ  /Bad spot tolerance (bytes)/
1885
1886 011670  DF1:    .ASCIZ  /Controller Initialization Timeout/
1887 011732  DF2:    .ASCIZ  /Controller never advanced to next step/
1888 012001  DF3:    .ASCIZ  /Controller can not execute local programs or non STD DUP dialog program/
1889 012111  DF4:    .ASCIZ  /NXM Trap at controllers IP address/
1890         ;DF10:  .ASCIZ  /No Interrupt occurred after SA polled/
```

F4

DISK PARAMETER QUESTIONS

```
1891 012154  DF11:    .ASCIZ  /Bad Response Packet returned/
1892 012211  DF12:    .ASCIZ  /Fatal SA error ctlr offl'ne/
1893 012245  DF13:    .ASCIZ  /No progress shown after a cmd had t'med out/
1894 012321  DF14:    .ASCIZ  /GET DUST CMD t'me_out after another CMD t'me out/
1895 012402  DF15:    .ASCIZ  /%N%AFatal error was reported when runn'ng local program/
1896 012472  DF16:    .ASCIZ  /%N%AA Spec'al was reported when runn'ng local program don't know how to handle 't/
1897 012614  SF0:     .ASCIZ  /DUP protocol Error, unexpected message/
1898 012663  SF1:     .ASCIZ  /%N%ASYSTEM 's NOT in manual mode/
1899 012724  SF100:   .ASCIZ  /Unexpected or delayed Controller Interrupt/
1900 012777  HRD0:    .ASCIZ  /Fatal Format error/
1901 013022  SFT0:    .ASCIZ  /Controller 'n an unexpected ACTIVE state/
1902 013073  SFT1:    .ASCIZ  /Wrong Model Number on controller/
1903 013134  PB0:     .ASCIZ  /%N%AModel # listed %06/
1904 013163  PB1:     .ASCIZ  /%N%AExpected SA step b't %06%A,Received in SA %06/
1905 013245  PB3:     .ASCIZ  /%N%AAsking for Format Parameter table/
1906 013313  PB4:     .ASCIZ  /%N%ARece'ved val'd Format Parameter table/
1907 013365  PB5:     .ASCIZ  /%N%AOn UNIT %06%A, %06 Bad Blks were found during Format/
1908 013456  PB6:     .ASCIZ  /%N%AOn UNIT %06%A, %06 Bad Blks were found dur'ng Ver'fy pass %06/
1909 013560  PB7:     .ASCIZ  /%N%ADUP Message Type: %06/
1910 013612  PB8:     .ASCIZ  /%N%ADUP message number: %06/
1911 013646  PB9:     .ASCIZ  /%N%AMSCP Controller model # : %D3/
1912 013710  PB10:    .ASCIZ  /%N%A    M'crocode version # : %D3/
1913 013752  PB11:    .ASCIZ  /%N%AController 's IDLE when 't should be ACTIVE running format program/
1914 014061  PB13:    .ASCIZ  /%N/
1915 014064  PF2:     .ASCIZ  /%N%N%AFinished local program w'thout procedure error/
1916 014151  PBF0:    .ASCIZ  /%N%AFormat Parameter table entry at byte %06%N%Ais out of range/
1917 014251  PBF1:    .ASCIZ  /%N%AFormat Parameter table entry at byte %06%N%Ais 'ncompatible with entry at byte %06/
1918 014400  PBF2:    .ASCIZ  /%N%AUNIT %06%A does not ex'st on controller/
1919 014454  PBF3:    .ASCIZ  /%N%AUNIT %06%A does exist but ooesn't respond on controller/
1920 014550  PBF4:    .ASCIZ  /%N%AUNIT %06%A is wr'te protected /
1921 014613  PBF5:    .ASCIZ  /%N%AWrite Fault detected on UNIT %06/
1922 014660  PBF6:    .ASCIZ  /%N%AAttempt to step hd %03%A at cyl %03%A fa'led on UNIT %06/
1923 014755  PBF7:    .ASCIZ  /%N%AAttempt to format hd %03%A at cyl %03%A fa'led on UNIT %06/
1924 015054  PBF8:    .ASCIZ  /%N%ATo many Bad Blocks                total Bad Blocks %06/
1925 015144  PBF9:    .ASCIZ  /%N%ADisk Controller model : %D3/
1926 015204  PBF10:   .ASCIZ  /%N%A    M'crocode vers'on : %D3/
1927 015244  PB11crr: .ASCIZ  /%N%AExpected CRN %06%A,Received CRN %06/
1928 015314  PB11op:  .ASCIZ  /%N%ACMDpkt Opcode %06%A,RSPpkt Opcode %06/
1929 015366  PB11sts: .ASCIZ  /%N%AResponse pkt status %06/
1930 015422  PB11end: .ASCIZ  /%N%ANo end bit(200) 'n response packet endcode/
1931 015501  PB11GDS: .ASCIZ  /%N%AGet Dust Status cmd/
1932 015531  PB11ESP: .ASCIZ  /%N%AExecute Supplied Prg cmd/
1933 015566  PB11ELP: .ASCIZ  /%N%AExecute Local Prg cmd/
1934 015620  PB11SD:  .ASCIZ  /%N%ASend Data cmd/
1935 015642  PB11RD:  .ASCIZ  /%N%ARece've Data cmd/
1936 015667  PB11AP:  .ASCIZ  /%N%AAbort Prg cmd/
1937 015711  pb11s0:  .ASCIZ  /%N%Asts:  successful/
1938 015736  pb11s1:  .ASCIZ  /%N%Asts:  Invalid Command/
1939 015770  pb11s2:  .ASCIZ  /%N%Asts:  No Region Available/
1940 016026  pb11s3:  .ASCIZ  /%N%Asts:  No Region Su'table/
1941 016063  pb11s4:  .ASCIZ  /%N%Asts:  Program Not Known/
1942 016117  pb11s5:  .ASCIZ  /%N%Asts:  Load Failure/
1943 016146  pb11s6:  .ASCIZ  /%N%Asts:  Standalone/
1944 016173  pb11s9:  .ASCIZ  /%N%Asts:  Host Buffer Access error/
1945 016236  pb11w0:  .ASCIZ  /%N%AUnknown command OPCODE rece'ved 'n t'meout loop/
1946 016322  pb11w1:  .ASCIZ  /%N%AUnknown command CRN rece'ved in command t'meout loop/
1947 016413  pb1201:  .ASCIZ  /%N%ASA er: Envelope\packet Read (par'ty or t'meout)/
```

DISK PARAMETER QUESTIONS

```
1948 016477  pb1202: .ASCIZ  /%N%ASA er: Envelope\packet Write (parity or timeout)/
1949 016564  pb1203: .ASCIZ  /%N%ASA er: Controller ROM and RAM parity/
1950 016635  pb1204: .ASCIZ  /%N%ASA er: Controller RAM parity/
1951 016676  pb1205: .ASCIZ  /%N%ASA er: Controller ROM parity/
1952 016737  pb1206: .ASCIZ  /%N%ASA er: Queue Read (parity or timeout)/
1953 017011  pb1207: .ASCIZ  /%N%ASA er: Queue Write (parity or timeout)/
1954 017064  pb1208: .ASCIZ  /%N%ASA er: Interrupt Master/
1955 017120  pb1209: .ASCIZ  /%N%ASA er: Host Access Timeout (h'gher level protocol dependent)/
1956 017221  pb1210: .ASCIZ  /%N%ASA er: Credit Limit Exceeded /
1957 017263  pb1211: .ASCIZ  /%N%ASA er: Bus Master Error/
1958 017317  pb1212: .ASCIZ  /%N%ASA er: D'agnostic Controller Fatal error/
1959 017374  pb1213: .ASCIZ  /%N%ASA er: Instruct'on Loop Timeout/
1960 017440  pb1214: .ASCIZ  /%N%ASA er: Invalid Connection Identifier/
1961 017511  pb1215: .ASCIZ  /%N%ASA er: Interrupt Wr'te Error/
1962 017552  pb1216: .ASCIZ  /%N%ASA er: MAINTENANCE READ\WRITE Invalid Region Identifier/
1963 017646  pb1217: .ASCIZ  /%N%ASA er: MAINTENANCE WRITE Load to non-loadable controller/
1964 017743  pb1218: .ASCIZ  /%N%ASA er: Controller RAM error (non-parity)/
1965 020020  pb1219: .ASCIZ  /%N%ASA er: INIT sequence error/
1966 020057  pb1220: .ASCIZ  /%N%ASA er: H'gh level protocol incompatibility error/
1967 020144  pb1221: .ASCIZ  /%N%ASA er: Purge\poll hardware failure/
1968 020213  pb1222: .ASCIZ  /%N%ASA er: Mapp'ng Register read error (parity or timeout)/
1969 020306  pb1223: .ASCIZ  /%N%ASA er: Attempt to set port data transfer mapping when option not present/
1970 020423  PB12:   .ASCIZ  /%N%ASA Value (oct) %06/
1971
1972 020452  PBsf0:  .ASCIZ  /%N%ADUP type %06%A message number %06/
1973 020520  DRPunt: .ASCIZ  /%N%N%ARQDX DRIVE %06%A f'nished./
1974 020561  TYPASC: .ASCIZ  /%N%APLEASE TYPE ANSWER to controller question or just <return>/
1975
1976          ;mmm
1977          ;
```

```
                        .sbttl  FORMAT Messages

                ; queries

020660  qfuit:  ;.byte  2,,,b.spl               ; Un't Info Table? (spl #2)
020660          .asciz  '%N%AEntering UIT%02%A: on drive number %D3%N'
020735  qfdat:  ;.byte  0,,,a.que               ; Date? (que #0)
020735          .asc'z  'Enter date <MM-DD-YYYY>:'
020766  dfunt:  ;.byte  1,,,a.def               ; Unit? (def #1)
020766          .asciz  'Enter unit number to format <0>:'
021027  dfbad:  ;.byte  4,,,a.def               ; Use Bad? (def #4)
021027          .asciz  'Use existing bad block 'nformation <N>:'
021077  dfdwn:  ;.byte  5,,,a.def               ; Downline? (def #5)
021077          .asc'z  'Use down-line load <Y>:'
021127  dfcon:  ;.byte  6,,,a.def               ; Continue? (def #6)
021127          .asciz  'Continue if bad block information is inaccessible <N>:'
021216  qfser:  ;.byte  7,,,a.que               ; Serial #? (que #7)
021216          .asciz  'Enter non zero serial number <8-10 digits>:'
021272  ASK.ANSWER:
021272          .asciz  'ans>'

                ; Informational Messages

021277  sfbegt: ;.byte  0,,,a.inf               ; Beg'n (inf #0)
021277          .asciz   %N%AFormat Begun'
021320  sfdont: ;.byte  1,,,a.inf               ; Complete (inf #1)
021320          .asc'z  '%N%AFormat complete'
021344  sfrevt: ;.byte  2,,,a.'nf               ; # of Revectored LBNs (inf #2)
021344          .asciz  '% Revectored LBNS'
021366  sfr1t:  ;.byte  3,,,a.inf               ; # of primary ... (inf #3)
021366          .asc'z  '% Pr'mary revectored LBNS'
021420  sfr2t:  ;.byte  4,,,a.'nf               ; # of secondary ... (inf #4)
021420          .asciz  '% Secondary/tert'ary revectored LBNS'
021465  sfrcbt: ;.byte  5,,,a.'nf               ; # of Bad RCT blocks ... (inf #5)
021465          .asciz  '% Bad blocks in the RCT area due to data errors'
021545  sfdbbt: ; b,te  7,,,a.'nf               ; # of Bad DBNs ... ('nf #7)
021545          .asciz  '% Bad blocks in the DBN area due to data errors'
021625  sfxbbt: ;.byte  9,,,a.inf               ; # of Bad XBNs ... (inf #9)
021625          .asciz  '% Bad blocks in the XBN area due to data errors'
021705  sftryt: ;.byte  11,,,a.inf              ; # of Retries (inf #11)
021705          .asciz  '% Blocks retr'ed on the check pass'
021750  sfrbbt: ;.byte  14,,,a.inf              ; # of Bad RBNs ... (inf #14)
021750          .asciz  '% Bad RBNS'
021763  sfcylt: ;.byte  15,,,a.'nf              ; Formatting Cyl (inf #15)
021763          .asciz  'Formatting Cyl %'
```

```
2025
2026                ; Successful Termination Messages
2027
2028                    ;.byte      12...a.ter              ; Reformat Worked (ter #12)
2029 022004  sffcut: .asciz  '%N%AFCT used successfully'
2030                    ;.byte  13...a.ter              ; Reconstruct Worked (ter #13)
2031 022036  sffcnt: .asc'i   '%N%AFCT was not used'
2032 022062          .asc'z   '%N%AFormat completed'
2033                ; Error messages
2034
2035 022107  efstat: ;.byte  1...a.fat                ; Status Error (fat #1)
2036 022107          .asciz   '%N%AGET STATUS fa'lure'
2037
2038 022136  efsndt: ;.byte  2...a.fat                ; Send Error (fat #2)
2039 022136          .asc'z    %N%AQ PORT send error'
2040
2041 022164  efcmdt: ;.byte  3...a.fat                ; Command Error (fat #3)
2042 022164          .asciz   '%N%AUnsuccessful command'
2043
2044 022215  efrcvt: ;.byte  4...a.fat                ; Receive Error (fat #4)
2045 022215          .asc'z   '%N%AQ-PORT rece've error'
2046
2047 022246  efbust: ;.byte  5...a.fat                ; Bus Error (fat #5)
2048 022246          .asc'z   '%N%AQ Bus I/O error'
2049
2050 022272  efinit: ;.byte  6...a.fat                ; Format Init Error (fat #6)
2051 022272          .asciz   '%N%AFormatter initialization error'
2052
2053 022335  efnut:  ;.byte  7...a.fat                ; Unit nonexistent error (fat #7)
2054 022335          .asciz   '%N%ANonexistent un't number'
2055
2056 022371  efdxft: ;.byte  8...a.fat                ; DBN/XBN Format error (fat #8)
2057 022371          .asciz   '%N%ADBN/XBN format error (drive FORMAT command failed)'
2058
2059 022460  effcct: ;.byte  9...a.fat                ; FCT copies error (fat #9)
2060 022460          .asciz   '%N%AFCT does not have enough good copies of each block'
2061
2062 022547  efsekt: ;.byte  10...a.fat               ; Seek error (fat #10)
2063 022547          .asciz   '%N%ASEEK error'
2064
2065 022566  efrcct: ;.byte  11...a.fat               ; RCT cop'es error (fat #11)
2066 022566          .asciz   '%N%ARCT does not have enough good cop'es of each block'
2067
2068 022655  eflbft: ;.byte  12...a.fat               ; LBN format error (fat #12)
2069 022655          .asciz   '%N%A_BN format error (dr've FORMAT command failed)'
2070
2071 022740  effcwt: ..byte  13...a.fat               ; FCT write error (fat #13)
2072 022740          .asciz   '%N%AFCT write error (check wr'te protect switch)'
2073
2074 023021  efrcrt: ;.byte  14...a.fat               ; RCT read error (fat #14)
2075 023021          .asciz   '%N%ARCT read error'
2076
2077 023044  efrcwt: ;.byte  15...a.fat               ; RCT wr'te error (fat #15)
2078 023044          .asciz   '%N%ARCT wr'te error'
2079
2080 023070  efrcft: ;.byte  16...a.fat               ; RCT full error (fat #16)
2081 023070          .asciz   '%N%ARCT full'
```

FORMAT Messages

```
2082
2083 023105  effcrt: ;.byte  17...a.fat              ; FCT read error (fat #17)
2084 023105           .asciz  '<N><A>FCT read error'
2085
2086 023130  effcnt: ;.byte  18...a.fat              ; FCT nonexistent error (fat #18)
2087 023130           .asciz  '<N><A>FCT nonexistent'
2088
2089 023154  effcdt: ;.byte  19...a.fat              ; FCT downline load error (fat #19)
2090 023154           .asciz  '<N><A>FCT Down-line load error'
2091
2092 023211  eftmot: ;.byte  20...a.fat              ; Drive timeout error (fat #20)
2093 023211           .asciz  '<N><A>Drive init timeout'
2094
2095 023240  efillt: ;.byte  21...a.fat              ; Illegal response error (fat #21)
2096 023240           .asciz  '<N><A>Illegal response to start-up question'
2097
2098 023312  efwart: ;.byte  22...a.fat              ; Head error (fat #22)
2099 023312           .asciz  '<N><A>WARNING - possible head addressing problem - run diagnostics'
2100
2101 023413  efinpt: ;.byte  23...a.fat              ; Input error (fat #23)
2102 023413           .asciz  '<N><A>INPUT Error '
2103
2104 023434  efmedt: ;.byte  24...a.fat              ; Media error (fat #24)
2105 023434           .asciz  '<N><A>Media degraded'
2106
2107 023457  efunrg: ;.byte  1...a.fat               ; Status Error (fat #1)
2108 023457           .asciz  '<N><A>Unrecogonized drive'
2109
```

FORMAT Messages

```
2111
2112
2113          ;**** ****************************************************************************
2114          ;
2115          ;       ASCII text added to allow a bad status returned by an MSCP command to
2116          ;       be reported.
2117          ;               The format of the message is MEXYYY.
2118          ;
2119          ;                       ME____  =>      Command issued was an MSCP Command
2120          ;                       __X___  =>      MSCP status code in hex
2121          ;                       ___YYY  =>      MSCP status code sub-code
2122          ;                                                               - GJK
2123          ;
2124          ;****************************************************************************
2125
2126 023507  ME10:    .asciz  '<N><A>Invalid Command'
2127 023533  ME20:    .asciz  '<N><A>Command Aborted'
2128 023557  ME30:    .asciz  '<N><A>UNIT <02><A> is unknown or online to another controller
2129 023651  ME31:    .asciz  '<N><A>UNIT <02><A> is disabled or no volume mounted'
2130 023731  ME32:    .asciz  '<N><A>UNIT <02><A> is inoperative'
2131 023767  ME34:    .asciz  '<N><A>UNIT <02><A> is a duplicate unit number'
2132 024041  ME38:    .asciz  '<N><A>UNIT <02><A> has been disabled by field service or diagnostic'
2133 024141  ME40:    .asciz  '<N><A>UNIT <02><A> is available'
2134 024175  ME55:    .asciz  '<N><A>UNIT <02><A> is not formatted with 512 byte sectors
2135 024263  ME56:    .asciz  '<N><A>UNIT <02><A> is not formatted or is FCT corrupted'
2136 024347  ME57:    .asciz  '<N><A>UNIT <02><A> FCT or RCT is unreadable due to an uncorrectable ECC Error
2137 024461  ME58:    .asciz  '<N><A>RCT search algorithm has encountered an invalid RCT entry'
2138 024557  ME59:    .asciz  '<N><A>No replacement block available.<N><A>Replacement was attempted for a bad block.
2139 024701  ME6128:  .asciz  '<N><A>UNIT <02><A> is software write protected
2140 024754  ME6256:  .asciz  '<N><A>UNIT <02><A> is hardware write protected'
2141 025027  ME70:    .asciz  '<N><A>Compare Error'
2142 025051  ME80:    .asciz  '<N><A>Sector was written with Force Error modifier'
2143 025132  ME82:    .asciz  '<N><A>Invalid Header'
2144 025155  ME83:    .asciz  '<N><A>Data Sync Timeout'
2145 025203  ME84:    .asciz  <N><A>Correctable error in ECC field'
2146 025246  ME87:    .asciz  '<N><A>Uncorrectable ECC Error'
2147 025302  ME88:    .asciz  <N><A>One Symbol ECC Error'
2148 025333  ME89:    .asciz  '<N><A>Two Symbol ECC Error'
2149 025364  ME810:   .asciz  <N><A>Three Symbol ECC Error
2150 025417  ME811:   .asciz  <N><A>Four Symbol ECC Error'
2151 025451  ME812:   .asciz  <N><A>Five Symbol ECC Error'
2152 025503  ME813:   .asciz  '<N><A>Six Symbol ECC Error'
2153 025534  ME814:   .asciz  <N><A>Seven Symbol ECC Error'
2154 025567  ME815:   .asciz  '<N><A>Eight Symbol ECC Error'
2155 025622  ME90:    .asciz  '<N><A>Host Buffer Access Error'
2156 025657  ME91:    .asciz  <N><A>Odd Transfer Address'
2157 025710  ME92:    .asciz  <N><A>Odd Byte Count'
2158 025733  ME93:    .asciz  <N><A>Non-Existent Memory Error'
2159 025771  ME94:    .asciz  '<N><A>Host Memory Parity Error'
2160 026026  ME95:    .asciz  '<N><A>Invalid Page Table Entry'
2161 026063  MEA1:    .asciz  '<N><A>SERDES overrun or underrun error'
2162 026130  MEA2:    .asciz  <N><A>EDC Error
2163 026146  MEA3:    .asciz  <N><A>Inconsistent internal control structure'
2164 026222  MEA4:    .asciz  <N><A>Internal EDC Error'
2165 026251  MEA5:    .asciz  <N><A>LESI Adapter Card parity error on input'
2166 026325  MEA6:    .asciz  <N><A>LESI Adapter Card parity error on output'
2167 026402  MEA7:    .asciz  <N><A>LESI Adapter Card 'cable in place' not asserted
```

L4

FORMAT Messages

```
   2168 026466  MEA8:    .asciz '%N%AController overrun or unoerrun'
   2169 026531  MEA9:    .asciz '%N%AController Memory Error'
   2170 026565  MEB1:    .asciz '%N%ADrive %02%A command t'me out'
   2171 026626  MEB2:    .asciz '%N%AController detected transm'ss'on error
   2172 026701  MEB3:    .asciz '%N%APositioner Error'
   2173 026726  MEB4:    .asciz  %N%ALost Read/Wr'te Ready dur'ng or between transfers'
   2174 027014  MEB5:    .asciz '%N%ADr've %02%A clock dropout'
   2175 027052  MEB6:    .asc'z '%N%ALost receiver ready for transfer
   2176 027117  MEB7:    .asciz '%N%ADrive %02%A detected error'
   2177 027156  MEB8:    .asc'z  %N%AController detected pulse or state par'ty error'
   2178 027242  MEB10:   .asc'z '%N%AController detected protocol error
   2179 027311  MEB11:   .asc'z '%N%ADr've %02%A fa'led initial'zat'on
   2180 027357  MEB12:   .asc'z '%N%ADrive %02%A 'gnored in't'al'zat'on'
   2181 027426  MEB13:   .asc'z '%N%AReceiver Ready Collision'
   2182
   2183                 ;                       End of MSCP Error Message Text
   2184                 ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
2186
2187
2188                ;*****************************************************************************
2189                ;
2190                ;       Messages that report which MSCP command was executed, MSCP status errors
2191                ;               and Bad Bytes found in a logical block.         - GJK
2192                ;
2193                ;*****************************************************************************
2194
2195 027463  MSCPsts:        .asciz '%N%AResponse Packet Status %O6%N'
2196 027524  MSCPend:        .asciz '%N%ANo end bit(200) in response packet endcode'
2197 027603  MSCPGUS:        .asciz '%N%AGet Unit Status command'
2198 027637  MSCPSCC:        .asciz '%N%ASet Controller Characteristics command'
2199 027712  MSCPONL:        .asciz '%N%AOn Line command'
2200 027736  MSCPRD:         .asciz '%N%ARead command'
2201 027757  MSCPWRT:        .asciz '%N%AWrite command'
2202 030001  MSCPOP:         .asciz '%N%ACMDpak Opcode %O6%A, RSPpak Opcode %O6'
2203
2204 030054  BTFND:  .asciz '%N%ATotal bad track(s) found:  %D4%N'
2205 030121  BTRPT:  .asciz '%N%ATrack %D4%A (decimal) has %D3%A (decimal) bad bytes'
2206 030211  DONE:   .ascii '%N%N%ADisk has been formatted and all available'
2207 030270          .asciz '%N%ALBNs have been tested for errors'
2208 030335  DSKUT:  .asciz '%N%N%ATesting LBNs on disk ...%N'
2209                ;*****************************************************************************
2210
2211 030376  RCVBUF: .BLKB   17000           ;Buffer to check data sent to disk      - GJK
2212                                .list   bin
2213                                .even
2214
```

Global subroutines

```
                        .sbttl global subroutines

                        ;*****************************************'************************
                        ;
                        ;       THIS ROUTINE WILL ALTER THE CPU INTERRUPT LEVEL TO THREE,
                        ;       INITIATE THE MUT POLLING AND WAIT A REASONABLE AMOUNT OF
                        ;       TIME HOPING THAT THE MUT WILL INTERRUPT.
                        ;There is really 2 sections to this routine. One section awaits the
                        ;normal course of a command by waiting for an iterrupt and then going
                        ;The other course of the routine is to handled t'meout commands. This is
                        ;
                        ;a little more difficult. If there is a command that is time-out the
                        ;program will do a GETDUST status command to f'nd out the status. IF
                        ;we get an interrupt there are 2 possibilties.
                        ; 1. a response to the Get Dust status command. Handled like any other
                        ;command.
                        ; 2. a response to the timeout command in which case we handle it instead
                        ;of the normal handler located in the program right after it was issued.
                        ;This means that there is a GET DUST response pending which we must handle
                        ;shortly after. As soor as we handle the GET DUST response we check to
                        ; make sure we handled the timed out command by checking the LSTCRN register
                        ; and return to the DUP dialog mode by checking the DUP message type and
                        ;responding to the intial timed out command
                        ;
                        ;  )>>>> waiting for initial or timed out command
                        ;  )     )interrupt received
                        ;  )     )just jump to vector address location
                        ;  )     )or on site response handler located right after cmd loader in the program
                        ;  )     )
                        ;  )     if no interrupt then save cmd info & submit GETDUST command
                        ;)>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>wa'ting for GETDUST status & timed out command
                        ;) )                              responses, if no interrupt then fatal error
                        ;) )                              )interrupt received
                        ;) )                              )test for response to GET DUST or TIMED_OUT cmd
                        ;) )                              )handle response here
                        ;) )                              )if response was GET DUST check if we handled
                        ;) )                              )the TIMED_OUT cmd already
                        ;) )     )<<<<<<<<<<<<<<<<<<<<<<<<)(yes, LSTCRN = 0) go back to DUP dialog mode
                        ;) )     )                          by checking DUP type and respond'ng
                        ;) )<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<)(no,  LSTCRN = #) load TIMED_OUT cmd Vector,
                        ;)       )                          Opcode, CRN goto top of routine
                        ;)<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<)if response was TIMED_OUT cmd clear LSTCRN
                        ;        )                          and wait for GET DUST response
                        ;        )
                        ;        ) go to DUP dialog routine
                        ;*************************************************************************

047376                  POLLWT:

047376  106427  000140  pollw:  mtps    #140                    ;Drop cpu level to three.
047402  005777  132716          tst     @IPreg                  ;Tell mut to start polling.

047406  005037  051162          clr     time                    ;reset timer
047412  012737  000001  002576  mov     #1,nxttim               ;Guarantee nxttim = time + 1
047420  012737  051172  000100  mov     #trp100,@#LKSvct        ;load the trap handler address
047426  012737  000340  000102  mov     #pri07,@#LKSvct+2       ;priority 7
```

```
2273
2274 047434  052737  000100  177546  2$:     bis     #bit6,@#LKS             ;Turn on line clock
2275 047442  023737  002576  051162  1$:     cmp     nxttim,time             ;Has 1 second delay expired ?
2276 047450  003374                          bgt     1$
2277 047452  042737  000100  177546          bic     #bit6,@#LKS             ;Turn off line clock
2278 047460  005237  002576                  inc     nxttim                  ;Update nxttim
2279 047464  106427  000340                  mtps    #340                    ;don't want interrupts while in other
2280                                                                         ;routines
2281 047470  004737  064316                  jsr     pc,BIT15T
2282 047474                                  BREAK                           ;check for control C
2283 047476  106427  000140                  mtps    #140                    ;turn on interrupts again after check
2284 047502  023737  002574  051162          cmp     delay,time              ;Has total delay been realized???
2285 047510  003351                          bgt     2$                      ;If so, then exit delay loop
2286
2287 047512  005737  002606                  tst     recv.done               ;is this the first time ?
2288 047516  001011                          bne     628$                    ;no, execute get dust command
2289 047520  013700  002462                  mov     cmdpak+10,r0            ;get opcode
2290 047524  022700  000002                  cmp     #op.esp,r0              ;If the command issued was a exec sup.
2291 047530  001403                          beq     627$
2292 047532  022700  000005                  cmp     #op.rec,r0              ;if the command issued was a recv. data
2293 047536  001001                          bne     628$
2294 047540  000777                  627$:   br      .
2295 047542  106427  000340          628$:   mtps    #340                    ;don't want interrupts while setting up
2296                                                                         ;for cmd
2297 047546  004737  064316                  jsr     pc,BIT15T               ;test SA make sure not a fatal error
2298 047552  013700  002462                  mov     cmdpak+10,r0            ;get opcode
2299 047556  022700  000001                  cmp     #op.gds,r0              ;if the command issued was a GETDUST
2300                                                                         ;STATUS and timeout big trouble
2301 047562  001006                          bne     GDS0                    ;if not go do a GET DUST to find out
2302                                                                         ;what the situation is
2303 047564                                  ERRDF   12,df14                 ;type no iterrupt after get dust status
2304                                                                         ;command controller dead
2305 047574  000137  074424                  jmp     dropunt                 ;drop unit and go on
2306
2307                              ;GETDUST                                   ;save timed out command information
2308
2309 047600  017737  132522  002544  GDS0:   mov     @vector,LSTVCT          ;store the vector address of timeout
2310                                                                         ;command
2311 047606  013737  002452  002540          mov     cmdpak,LSTCRN           ;store the CRN of the timed out command
2312 047614  013737  002462  002542          mov     cmdpak+10,LSTCMD        ;store the opcode of timed out command
2313
2314 047622  032737  100000  002534          bit     #bit15,cmdrng+2         ;test ownership of ring make sure we
2315                                                                         ;own it
2316 047630  001363                          bne     GDS0                    ;if we don't own it wait until we do
2317 047632  012737  000016  002446          mov     #14.,cmdlen             ;load length of packet to be sent
2318 047640  112737  000000  002450          movb    #0,cmdlen+2             ;load msg type and credit
2319 047646  112737  000002  002451          movb    #dup.id,cmdlen+3        ;load DUP connection ID
2320 047654  005237  002452                  inc     cmdpak                  ;load new CRN
2321 047660  005037  002454                  clr     cmdpak+2
2322 047664  005037  002456                  clr     cmdpak+4
2323 047670  005037  002460                  clr     cmdpak+6
2324 047674  012737  000001  002462          mov     #op.gds,cmdpak+10       ;load up opcode
2325 047702  005037  002464                  clr     cmdpak+12               ;no modifiers
2326
2327 047706  012777  047746  132412          mov     #RFDO,@vector           ;NEW VECTOR PLACE
2328 047714  012737  002352  002526          mov     #rsppak,rsprng          ;load response packet area into ring
2329 047722  012737  002452  002532          mov     #cmdpak,cmdrng          ;load command packet area into ring
```

Global subroutines

```
2330 047730  012737  140000  002530          mov     #140000,RSPRNG+2        ;PORT OWNERSHIP BIT.
2331 047736  012737  100000  002534          mov     #bit15,CMDRNG+2
2332 047744  000614                          br      POLLWT                  ;GO and wait for interrupt
2333
2334
2335                                 ;+
2336                                 ; There are only 3 ways out code.
2337                                 ;If GETDUST response and TIMED_OUT cmd response was handled
2338                                 ; if LSTCRN = 0 and RSPPAK+10 = OP.GDS+OP.END then
2339                                 ; back to DUP dialog mode.
2340                                 ;or
2341                                 ; (TIMED_OUT cmd still hasn't returned but GETDUST has returned)
2342                                 ; if LSTCRN = 0 and RSPPAK+10 = OP.GDS+OP.END then
2343                                 ; check if idle or active. if idle then error
2344                                 ; check for progress in progress indicator if no progress then error
2345                                 ; load LSTVCT into @vector,LSTCRN into cmdpak, LSTCMD into cmdpak+10
2346                                 ; set response ring ownership to Port Owned
2347                                 ; jmp to pollwt.
2348                                 ;or
2349                                 ; (TIMED_OUT cmd response received before GETDUST response returned)
2350                                 ; if LSTCRN = 0 and RSPPAK+10 not= OP.GDS+OP.END then
2351                                 ; clear LSTCRN and
2352                                 ; jmp to pollwt.
2353                                 ;+
2354
2355 047746                        RFDO:                                   ;INTR TO HERE if GETDUST or TIMED_OUT
2356                                                                         ;command
2357 047746  106427  000340                  mtps    #340                    ;No interrupts please
2358 047752  062706  000004                  add     #4,sp                   ;fix stack 4 for intrpt
2359 047756  013701  002452                  mov     cmdpak,r1               ;check command packet CRN
2360 047762  013700  002352                  mov     rsppak,r0               ;check response packet CRN
2361 047766  020001                          cmp     r0,r1                   ;Are they the SAME must be GETDUST cmd
2362 047770  001107                          bne     3$                      ;if not it must be the TIMED_OUT cmd
2363
2364 047772  023727  002362  000201          cmp     rsppak+10,#op.gds+op.end         ;it should be a GETDUST lets
2365                                                                                  ;make sure
2366 050000  001412                          beq     1$
2367 050002                                  printf  #pb11w0                 ;unexpected cmd response in time out loop
2368 050022  000137  074410                  jmp     unkwn                   ;error handler
2369
2370 050026  004737  060352          1$:     jsr     pc,RSPCHK               ;check the response
2371 050032  005737  002540                  tst     LSTCRN                  ;see if timed out command was already
2372                                                                         ;received (lstcrn = 0)
2373 050036  001004                          bne     2$
2374 050040  062706  000002                  add     #2,sp                   ;adjust stack for Timed Out cmd's
2375                                                                         ;initial call to POLLWT
2376 050044  000137  071104                  jmp     DUPDLG                  ;if Timed out cmd was already received
2377                                                                         ;then goto DUP dialog mode
2378
2379 050050                          2$:                                     ;if Timed out command was not received
2380                                                                         ;already (LSTCRN not= 0)
2381 050050  132737  000010  002371          bitb    #bit3,rsppak+17         ;if server idle then error
2382 050056  001010                          bne     1002$                   ;if not check for progress
2383 050060                                  printf  #pb11                   ;controller idle when it should be active
2384
2385 050100  013700  002372          1002$:  mov     rsppak+20,r0            ;check for progress in progress indicator
2386 050104  013701  002374                  mov     rsppak+22,r1
```

Global subroutines

```
2387 050110  020037  002546                cmp     r0,loprgi           ;see if low word of progress indicator
2388                                                                    ;is the same as older value
2389 050114  001011                        bne     1001$               ;if it is then continue
2390 050116  020137  002550                cmp     r1,hiprgi           ;see if high vaule is the same
2391 050122  001006                        bne     1001$
2392 050124                                ERRDF   11,DF13             ;no progress shown after cmd timeout
2393 050134  000137  074424                jmp     dropunt
2394 050140                        1001$:
2395 050140  010037  002546                mov     r0,loprg            ;update progress indicator
2396 050144  010137  002550                mov     r1,hiprg
2397 050150  004737  050300                jsr     pc,FPRPT            ;Call format progress report
2398 050154  013737  002540  002452        mov     LSTCRN,cmdpak       ;move TIMED_OUT cmd CRN into cmd
2399 050162  013737  002542  002462        mov     LSTCMD,cmdpak+10    ;move TIMED_OUT cmd Opcode into cmd
2400 050170  013777  002544  132130        mov     LSTVCT,@vector      ;load TIMED_OUT cmd interrupt handler
2401                                                                    ;address into vector
2402 050176  012737  140000  002530        mov     #140000,RSPRNG+2    ;Port owned
2403 050204  000137  047376                jmp     pollw               ;wait for TIMED_OUT cmd response
2404
2405
2406
2407 050210  020037  002540        3$:     cmp     r0,LSTCRN           ;check the crn with the last CRN from
2408                                                                    ;the timeout command
2409 050214  001412                        beq     4$                  ;
2410 050216                                printf  #pb11w1             ;Unexpected cmd response in time out loop
2411 050236  000137  074410                jmp     unkwn               ;error handler
2412
2413                                                                    ;Timed out command received but Get Dust
2414                                                                    ;Status is still in Queue
2415 050242  013737  002540  002452  4$:     mov     LSTCRN,cmdpak       ;load timed out command values for
2416                                                                    ;RSPCHK routine
2417 050250  013737  002542  002462        mov     LSTCMD,cmdpak+10    ;load timed out command values for
2418                                                                    ;RSPCHK routine
2419 050256  005037  002540                clr     LSTCRN              ;if it is the timeout command clear LAST
2420                                                                    ;CRN register
2421 050262  004737  060352                jsr     pc,RSPCHK           ;go check the command
2422 050266  012737  140000  002530        mov     #140000,RSPRNG+2    ;PORT OWNERSHIP BIT.
2423 050274  000137  047376                jmp     POLLW               ;go wait for GETDUST interrupt
```

Global subroutines

```
2425
2426
2427                                    ;*****************************************************************************
2428                                    ;
2429                                    ;    Format Progress Report  (Done Only for uCode Rev 2 or higher)
2430                                    ;
2431                                    ;*****************************************************************************
2432 050300                            FPRPT:
2433 050300  023727  002342  000002            cmp     mcdnbr,#2       ;check microcode rev number
2434 050306  002001                            bge     33$             ;If rev > or = 2 continue execution
2435 050310  000207                            return                  ;If not, don't output progress report
2436
2437 050312  032737  004000  002336  33$:     bit     #bit11,UNTflgs  ;Has title already been printed ??
2438 050320  001013                            bne     22$             ;If so, don't print it again
2439 050322                                    printf  #FRPTB
2440 050342  052737  004000  002336            bis     #bit11,UNTflgs  ;Set bit 11 in flag register so title only
2441                                                                    ;appears once
2442 050350  122737  000000  002551  22$:     cmpb    #0,hiprgi+1     ;Is pass = 0 ??
2443 050356  001022                            bne     1$              ;If not, check for pass = 1
2444 050360                                    printf  #FMTTRK         ;Print "Formatting Tracks, lbn #: "
2445 050400  004737  051220                    jsr     pc,DECasc       ;Convert counter to ASCII characters
2446 050404                                    printf  #tmpbuf         ;Print lbn number
2447
2448 050424  122737  000001  002551  1$:      cmpb    #1,hiprgi+1     ;Is pass = 1 ??
2449 050432  001032                            bne     2$              ;If not, check for pass = 2
2450 050434  105037  002551                    clrb    hiprgi+1        ;Make sure 8 MSBs are clear
2451 050440  005737  002546                    tst     loprgi          ;Are we just reading defect list ??
2452 050444  001011                            bne     11$
2453 050446                                    printf  #RDDFCT         ;Yes, print "Reading defect list"
2454 050466  000414                            br      2$              ;Continue with rest of routine
2455 050470                            11$:    printb  #RPDFCT,loprgi,hiprgi
2456                                                                    ;No, print "Replacing defect #: ___ on head #:
2457
2458 050520  122737  000002  002551  2$:      cmpb    #2,hiprgi+1     ;Is pass = 2 ??
2459 050526  001024                            bne     3$              ;If not, check for pass = 3
2460 050530  105037  002551                    clrb    hiprgi+1        ;Make sure 8 MSBs are clear
2461 050534                                    printf  #FCPW           ;Print "First check pass, writing lbn #:
2462 050554  004737  051220                    jsr     pc,DECasc       ;Convert counter to ASCII characters
2463 050560                                    printf  #tmpbuf         ;Print lbn number
2464
2465 050600  122737  000003  002551  3$:      cmpb    #3,hiprgi+1     ;Is pass = 3 ??
2466 050606  001024                            bne     4$              ;If not, check for pass = 4
2467 050610  105037  002551                    clrb    hiprgi+1        ;Make sure 8 MSBs are clear
2468 050614                                    printf  #FCPR           ;Print "First check pass, reading lbn #: "
2469 050634  004737  051220                    jsr     pc,DECasc       ;Convert counter to ASCII characters
2470 050640                                    printf  #tmpbuf         ;Print lbn number
2471
2472 050660  122737  000004  002551  4$:      cmpb    #4,hiprgi+1     ;Is pass = 4 ??
2473 050666  001024                            bne     5$              ;If not, check for pass = 5
2474 050670  105037  002551                    clrb    hiprgi+1        ;Make sure 8 MSBs are clear
2475 050674                                    printf  #SCPW           ;Print  Second check pass, writing lbn #:
2476 050714  004737  051220                    jsr     pc,DECasc       ;Convert counter to ASCII characters
2477 050720                                    printf  #tmpbuf         ;Print lbn number
2478
2479 050740  122737  000005  002551  5$:      cmpb    #5,hiprgi+1     ;Is pass = 5 ??
2480 050746  001024                            bne     6$              ;If not, check for pass = 6
2481 050750  105037  002551                    clrb    hiprgi+1        ;Make sure 8 MSBs are clear
```

F5

Global subroutines

```
2482 050754                                        printf  #SCPR           ;Print 'Second check pass, reading lbn #: "
2483 050774  004737  051220                        jsr     pc,DECasc      ;Convert counter to ASCII characters
2484 051000                                        printf  #tmpbuf        ;Print lbn number
2485
2486 051020  122737  000006  002551  6$:          cmpb    #6,hiprgi+1    ;Is pass = 6 ??
2487 051026  001024                                bne     7$             ;If not, then pass = 7
2488 051030  105037  002551                        clrb    hiprgi+1       ;Make sure 8 MSBs are clear
2489 051034                                        printf  #TCPW          ;Print "Third check pass, writing lbn #:
2490 051054  004737  051220                        jsr     pc,DECasc      ;Convert counter to ASCII characters
2491 051060                                        printf  #tmpbuf        ;Print lbn number
2492
2493 051100  122737  000007  002551  7$:          cmpb    #7,hiprgi+1    ;Is pass = 7 ??
2494 051106  001024                                bne     8$             ;If not, then return to calling program
2495 051110  105037  002551                        clrb    hiprgi+1       ;Make sure 8 MSBs are clear
2496 051114                                        printf  #TCPR          ;Print "Second check pass, reading lbn #:
2497 051134  004737  051220                        jsr     pc,DECasc      ;Convert counter to ASCII characters
2498 051140                                        printf  #tmpbuf        ;Print lbn number
2499
2500 051160  000207                        8$:     return
```

Global subroutines

```
                                ;*****************************************************************
                                ;
                                ;                       Line Clock Timer Routine
                                ;
                                ;*****************************************************************

051162  000000                  time:   .word                           ;Time in Seconds
051164  000000                          .word
                                
                                        .word                           ;Counter for Cycles per Second
051170  000074                          .word   60.                     ;Cycles per Second
                                
051172                          trp100:
051172  005237  051166                  inc     time+4                  ;Add a cycle
051176  023727  051166  051170          cmp     time+4,time+6           ;Compare to total cycle time
                                                                        ;50 Hz or 60 Hz
051204  003404                          ble     10$
051206  005037  051166                  clr     time+4                  ;Reinit the cycle timer
051212  005237  051162                  inc     time                    ;Add a second
                                ;               adc     time+2          ;Add carry to high word
051216  000002                  10$:    rti                             ;Return from interrupt
```

Global subroutines

```
2526
2527
2528                                    ;*****************************************************************
2529                                    ;
2530                                    ;       Octal number to ASCII Decimal number
2531                                    ;
2532                                    ;        r1 = address of ascii decimal data
2533                                    ;         r0 = address of octal data word
2534                                    ;
2535                                    ;*****************************************************************
2536
2537 051220                            DECasc:
2538 051220  010146                            mov     r1,-(sp)
2539 051222  013737  002546  051444            mov     loprg ,dtmp             ;octal number address
2540 051230  013737  002550  051446            mov     hiprgi,dtmp+2           ;octal number address
2541 051236  012701  051430                    mov     #tmpbuf+2,r1            ;asc z buffer address
2542 051242  010146                            mov     r1,-(sp)
2543 051244  010246                            mov     r2,-(sp)
2544 051246  010346                            mov     r3,-(sp)
2545 051250  005002                            clr     r2                      ;clear the decimal table pointer
2546 051252  005003                    1$:     clr     r3                      ;clear decimal digit
2547 051254  005203                    2$:     inc     r3                      ;ncrement decimal digit
2548 051256  166237  051450  051444            sub     dtbl(r2),dtmp           ;subtract power of ten from accumulator
2549 051264  005637  051446                    sbc     dtmp+2
2550 051270  166237  051452  051446            sub     dtbl+2(r2),dtmp+2
2551 051276  002366                            bge     2$                      ;if not negative subtract another
2552 051300  066237  051450  051444            add     dtbl(r2),dtmp           ;adjust accumulator so posit ve
2553 051306  005537  051446                    adc     dtmp+2
2554 051312  066237  051452  051446            add     dtbl+2(r2),dtmp+2
2555 051320  005303                            dec     r3                      ;adjust decimal d g t
2556 051322  062703  000060                    add     #60,r3                  ;convert dec mal to ascii
2557 051326  110321                            movb    r3,(r1)+                ;mov ascii d git text into buffer
2558 051330  062702  000004                    add     #4,r2                   ;increment table pointer
2559 051334  005762  051450                    tst     dtbl(r2)                ;check if thats all
2560 051340  001344                            bne     1$
2561 051342  105021                            clrb    (r1)+                   ;store null
2562
2563 051344                            PURL0:                                  ;Purge leading zeroes of counter convrsn
2564 051344  005001                            clr     r1                      ;Set up addresses, flags, and/or indices
2565 051346  005002                            clr     r2
2566 051350  012703  051430                    mov     #tmpbuf+2,r3
2567
2568 051354  122761  000060  051430    1$:     cmpb    #60,tmpbuf+2(r1)        ;Is byte a leading 0 ??
2569 051362  002404                            blt     3$                      ;Byte must be a digit
2570 051364  003011                            bgt     4$                      ;Byte must be a null
2571 051366  032702  000001                    b t     #b t0,r2                ;If no non-zero digits have been found,
2572 051372  001404                            beq     2$                      ;go to next byte
2573 051374  052702  000001            3$:     b s     #b t0,r2                ;flag first non-zero dig t found
2574 051400  116123  051430                    movb    tmpbuf+2(r1),(r3)+      ;move byte to proper location in buffer
2575 051404  005201                    2$:     nc      r1                      ;Update po nter
2576 051406  000762                            br      1$                      ;Check next byte
2577 051410  116123  051430            4$:     movb    tmpbuf+2(r1),(r3)+      ;move null and end
2578
2579 051414  012603                            mov     (sp)+,r3
2580 051416  012602                            mov     (sp)+,r2
2581 051420  012601                            mov     (sp)+,r1                ;address preserved
2582 051422  012601                            mov     (sp)+,r1                ;restore or ginal r1
```

Global subroutines

```
2583 051424  000207                        return
2584
2585 051426     045     101        tmpbuf: .ascii /%A/                ;Provide buffer for ascii data
2586 051430                                .blkb   11.                ;10 bytes for digits, 1 for null
2587                                .even
2588
2589 051444  000000                atmp:   0
2590 051446  000000                        0
2591 051450                        dtbl:
2592 051450  145000                        145000                     ; 1.0 E09
2593 051452  035632                        35632
2594 051454  160400                        160400                     ; 1.0 E08
2595 051456  002765                        2765
2596 051460  113200                        113200                     ; 1.0 E07
2597 051462  000230                        230
2598 051464  041100                        041100                     ; 1.0 E06
2599 051466  000017                        17
2600 051470  103240                        103240                     ; 1.0 E05
2601 051472  000001                        1
2602 051474  023420                        23420                      ; 1.0 E04
2603 051476  000000                        0
2604 051500  001750                        1750                       ; 1.0 E03
2605 051502  000000                        0
2606 051504  000144                        144                        ; 1.0 E02
2607 051506  000000                        0
2608 051510  000012                        12                         ; 1.0 E01
2609 051512  000000                        0
2610 051514  000001                        1                          ; 1.0 E00
2611 051516  000000                        0
2612 051520  000000                        0                          ; endflag
2613                               .even
2614
```

Global subroutines

```
2616
2617
2618                                    ;*****************************************************************************
2619                                    ;       HARD INITIALIZE
2620                                    ; This routine hard initialize the disk controller so that commands
2621                                    ; can be issued. This routine is governed by the UQSSP spec.
2622                                    ; This format starts by initializating
2623                                    ;       r1 = ptable address
2624                                    ;       r3 = step bit mask
2625                                    ;       r4 = SA address
2626
2627
2628 051522                            HRDINT:
2629 051522  106427  000140                    mtps    #140
2630 051526  013704  002324                    mov     ipreg,r4
2631 051532  005024                    1$:     clr     (r4)+                   ; start initialization IP and put SA
2632 051534  012703  004000                    mov     #bit11,r3
2633 051540  012746  000024                    mov     #20.,-(sp)
2634 051544  004737  052626                    jsr     pc,sleep                ;looking for step 1 >>>>>>>>>>>>>>>>>
2635 051550  005726                          tst     (sp)+
2636 051552  004737  064316            4$:     jsr     pc,bit15T               ;Look for fatal SAreg error
2637 051556  032714  004000                    bit     #bit11,(r4)             ;check sa reg for step 1        bit11 =0 >1
2638 051562  001004                          bne     6$
2639 051564  005737  002602                    tst     timeout
2640 051570  001370                          bne     4$
2641 051572  000571                          br      timout
2642 051574  013700  002326            6$:     mov     vector,r0               ;responding in step 1 <<<<<<<<<<<<<<<<
2643 051600  000241                          clc
2644 051602  006200                          asr     r0
2645 051604  006200                          asr     r0
2646 051606  052700  100200                    bis     #<bit15+bit7>,r0        ;= VECTOR/4
2647 051612  013701  002326                    mov     vector,r1
2648 051616  012721  052730                    mov     #saint,(r1)+            ;load up interrupt location into vector
2649 051622  012711  000140                    mov     #140,(r1)               ;lower the priority
2650 051626  010014                          mov     r0,(r4)                 ; Enable INTERRUPTS, set 1 cmd rng and
2651                                                                          ;1 rsp rng
2652 051630  012703  010000                    mov     #bit12,r3
2653 051634  012746  000024                    mov     #20.,-(sp)
2654 051640  004737  052626                    jsr     pc,sleep                ;looking for step 2 >>>>>>>>>>>>>>>>>
2655 051644  005726                          tst     (sp)+
2656 051646  004737  064316            44$:    jsr     pc,bit15T               ;look for sa error
2657 051652  032714  010000                    bit     #bit12,(r4)             ;check step
2658 051656  001007                          bne     12$
2659 051660  005737  002602                    tst     timeout
2660 051664  001370                          bne     44$
2661 051666  032714  004000                    bit     #bit11,(r4)             ;check step
2662 051672  001151                          bne     wrngstep
2663 051674  000530                          br      timout
2664
2665 051676  012700  002526            12$:    mov     #RSPRNG,R0              ;responding in step 2 <<<<<<<<<<<<<<<<
2666 051702  042700  000001                    bic     #bit0,r0                ;no adapter purge interrupts
2667 051706  010014                          mov     r0,(r4)                 ;load low ringbase address of the
2668                                                                          ;communications area
2669 051710  012703  020000                    mov     #bit13,r3
2670 051714  012746  000024                    mov     #20.,-(sp)
2671 051720  004737  052626                    jsr     pc,sleep                ;looking for step 3 >>>>>>>>>>>>>>>>>
2672 051724  005726                          tst     (sp)+
```

Global subroutines

```
2673 051726  004737  064316          444$:   jsr     pc,bit15T               ;look for sa error
2674 051732  032714  020000                  bit     #bit13,(r4)             ;check step
2675 051736  001012                          bne     13$
2676 051740  005737  002602                  tst     timeout
2677 051744  001370                          bne     444$
2678 051746  032714  004000                  bit     #bit11,(r4)             ;check step
2679 051752  001121                          bne     wrngstep
2680 051754  032714  010000                  bit     #bit12,(r4)             ;check step
2681 051760  001116                          bne     wrngstep
2682 051762  000475                          br      timout
2683
2684 051764  013701  002326          13$:    mov     vector,r1
2685 051770  012721  052730                  mov     #saint,(r1)+            ;load up interrupt loacation into vector
2686 051774  012711  000340                  mov     #340,(r1)               ;after step four we want no interrupts
2687                                                                         ;until expected.
2688 052000  005014                          clr     (r4)                    ;load low ringbase address of the
2689                                                                         ;communications area
2690 052002  012703  040000                  mov     #bit14,r3
2691 052006  012746  000024                  mov     #20.,-(sp)
2692 052012  004737  052626                  jsr     pc,sleep                ;looking for step 4 >>>>>>>>>>>>>>>>>
2693 052016  005726                          tst     (sp)+
2694 052020  004737  064316          4444$:  jsr     pc,bit15T               ;look for sa error
2695 052024  032714  040000                  bit     #bit14,(r4)             ;check step
2696 052030  001015                          bne     18$
2697 052032  005737  002602                  tst     timeout
2698 052036  001370                          bne     4444$
2699 052040  032714  004000                  bit     #bit11,(r4)             ;check step
2700 052044  001064                          bne     wrngstep
2701 052046  032714  010000                  bit     #bit12,(r4)             ;check step
2702 052052  001061                          bne     wrngstep
2703 052054  032714  020000                  bit     #bit13,(r4)             ;check step
2704 052060  001056                          bne     wrngstep
2705 052062  000435                          br      timout
2706
2707 052064  011401                  18$:    mov     (r4),r1                 ;identify the controller number and
2708                                                                         ;microcode version number
2709 052066  010102                          mov     r1,r2
2710 052070  006201                          asr     r1                      ;shift version # out
2711 052072  006201                          asr     r1
2712 052074  006201                          asr     r1
2713 052076  006201                          asr     r1
2714 052100  042701  177400                  bic     #177400,r1              ;clear top bits off
2715 052104  010137  002340                  mov     r1,mdlnbr               ;model number storage
2716 052110  042702  177760                  bic     #177760,r2              ;clear model number out
2717 052114  010237  002342                  mov     r2,mcdnbr
2718 052120  122701  000007                  cmpb    #Mrqdx1,r1              ;check for Model #
2719 052124  001454                          beq     gobit
2720 052126  122701  000023                  cmpb    #Mrqdx3,r1              ;check for Model #
2721 052132  001451                          beq     gobit
2722
2723 052134                                  ERRSOFT 6,SFT1                  ; DEVICE FATAL wrong model #, wrong
2724                                                                         ;controller
2725 052144  052737  020000  002336          bis     #bit13,UNTflgs          ;set unknown model number in unit flags
2726 052152  000137  052256                  jmp     gobit                   ;drop unit and go on
2727
2728 052156                          timout:
2729 052156                                  ERRDF   5,DF1                   ; DEVICE FATAL controller timeout
```

Global subroutines

```
2730                                                    ; during hard init
2731 052166                            Printf  #pb1,r3,(r4)     ; Expected SA step bit xxxxx set,
2732                                                    ; received yyyyyy
2733 052212  000137  074424            jmp     dropunt          ;drop unit and go on
2734
2735 052216                    wrngstep:
2736 052216                            ERRDF   4,DF2            ; DEVICE FATAL wrong step bit set
2737                                                    ; after interrupt
2738 052226                            Printf  #pb1,r3,(r4)     ; Expected SA step bit xxxxx, received
2739                                                    ; in SA yyyyyy
2740 052252  000137  074424            jmp     dropunt          ;drop unit and go on
2741
2742 052256                    GOBIT:
2743 052256  012714  000001            mov     #1,(r4)          ;Controller is NOW INITIALIZED
2744
2745 052262  012700  177777            mov     #-1,r0
2746 052266  000240            11$:    nop                      ;waste just a little time
2747 052270  077002                    sob     r0,11$
2748
2749 052272                    GDScmd:
2750 052272                    GETDUST
     052272  032737  100000  002534    GDS2:   bit     #bit15,cmdrng+2  ;Do a Get Dust Status command start
                                                                ;test ownership of ring make sure we own
                                                                ;it
     052300  001374                    bne     GDS2             ;if we don't own it wait until we do
     052302  012737  000016  002446    mov     #14.,cmdlen      ;load length of packet to be sent
     052310  112737  000000  002450    movb    #0,cmdlen+2      ;load msg type and credit
     052316  112737  000002  002451    movb    #dup.id,cmdlen+3 ;load DUP connection ID
     052324  005237  002452            inc     cmdpak           ;load new CRN
     052330  005037  002454            clr     cmdpak+2
     052334  005037  002456            clr     cmdpak+4
     052340  005037  002460            clr     cmdpak+6
     052344  012737  000001  002462    mov     #op.gds,cmdpak+10 ;load up opcode
     052352  005037  002464            clr     cmdpak+12        ;no modifiers

     052356  012777  052420  127742    mov     #RFD2,@vector    ;New vector place
     052364  012737  002352  002526    mov     #rsppak,rsprng   ;load response packet area into ring
     052372  012737  002452  002532    mov     #cmdpak,cmdrng   ;load command packet area into ring
     052400  012737  140000  002530    mov     #140000,RSPRNG+2 ;Port ownership bit.
     052406  012737  100000  002534    mov     #bit15,CMDRNG+2
     052414  004737  047376            jsr     pc,POLLWT        ;Go to poll and wait routine.

                            ;*************************************************************************

     052420                         RFD2:                       ;Intr to here.
     052420  062706  000006            add     #6,sp            ;fix stack for interrupt (4), pollwt
                                                                ;subrtn (2)
     052424  012777  065360  127674    mov     #intsrv,@vector  ;Change vector
     052432  004737  060352            jsr     pc,RSPCHK        ;Go to routine that will check on
                                                                ;the response recvd from the mut.
                                                                ;it will check the cmd ref
                                                                ;num, the endcode and status.
2751                                                            ;things off
2752 052436  132737  000010  002371    bitb    #bit3,rsppak+17  ;is this server active already
2753 052444  001467                    beq     dnint            ;branch to Execute Local Program
2754 052446                            ERRSOFT 3,SFTO           ;Soft Error "already active will do
2755                                                            ;an ABORT cmd"
2756 052456                    ABRT                            ;Doing an ABRT do get into idle state
```

M5

Global subroutines

```
052456  032737  100000  002534  ABRT3: b't     #bit15,cmdrng+2         ;test ownersh'p of r'ng make sure we
                                                                       ;own it
052464  001374                          bne     ABRT3                   ;if we don't own 't wait unt'l we do
052466  012737  000016  002446          mov     #14.,cmdlen             ;load length of packet to be sent
052474  112737  000000  002450          movb    #0,cmdlen+2             ;load msg type and cred't
052502  112737  000002  002451          movb    #dup.id,cmdlen+3        ;load DUP connection ID
052510  005237  002452                  'nc     cmdpak                  ;load new CRN
052514  005037  002454                  clr     cmdpak+2
052520  005037  002456                  clr     cmdpak+4
052524  005037  002460                  clr     cmdpak+6
052530  012737  000006  002462          mov     #op.abrt,cmdpak+10      ;load up opcode
052536  005037  002464                  _r      cmdpak+12               ;no mod'f'ers

052542  012777  052604  127556          mov     #RFD3,@vector           ;New vector place
052550  012737  002352  002526          mov     #rsppak,rsprng          ;load response packet area into ring
052556  012737  002452  002532          mov     #cmdpak,cmdrng          ;load command packet area into ring
052564  012737  140000  002530          mov     #140000,RSPRNG+2        ;Port ownership bit.
052572  012737  100000  002534          mov     #bit15,CMDRNG+2
052600  004737  047376                  jsr     pc,POLLWT               ;Go to poll and wait routine

                                ;******;*****************************************************************

052604                          RFD3:                                   ;Intr to here.
052604  062706  000006                  add     #6,sp                   ;fix stack for interrupt (4), pollwt
                                                                       ;subrtn (2)
052610  012777  065360  127510          mov     #intsrv,@vector         ;Change vector
052616  004737  060352                  jsr     pc,RSPCHK               ;Go to routine that will check on
                                                                       ;the response recvd from the mut.
                                                                       ;it will check the cmd ref
                                                                       ;num, the endcode and status.
2757 052622  000623                          br      GDScmd             ;branch back to make sure not busy
2758 052624                          DNINT:
2759 052624  000207                          rts     pc
2760
2761                                 ; Delay routine
2762
2763 052626                          sleep::
2764 052626  010146                          mov     r1,-(sp)
2765 052630  012737  052700  000100          mov     #lks.server,@#LKSvct    ;load the trap handler address
2766 052636  012737  000340  000102          mov     #pri07,@#LKSvct+2       ;priority 7
2767 052644  016601  000004                  mov     4(sp),r1
2768 052650  070137  002604                  mul     herz,r1
2769 052654  010137  002600                  mov     r1,timer
2770 052660  012737  000001  002602          mov     #1,timeout
2771 052666  052737  000100  177546          bis     #bit6,@#lks
2772 052674  012601                          mov     (sp)+,r1
2773 052676  000207                          rts     pc
2774
2775                                 ; Line time clock interrupt service routine
2776
2777 052700                          lks.server::
2778 052700  005737  002600                  tst     timer
2779 052704  003403                          ble     1$
2780 052706  005337  002600                  dec     t'mer
2781 052712  000405                          br      2$
2782 052714  005037  002602          1$:     clr     timeout
2783 052720  042737  000100  177546          bic     #bit6,@#lks
```

Global subroutines

```
2784 052726  000002                2$:     rti
2785
2786                               ; RQDX interrupt server
2787
2788 052730                        saint::
2789 052730  000002                        rti
2790
```

DUP^mt Supplied Program Definitions

```
2792                                    .sbttl  DUP^mt Supplied Program Definitions
2793
2794                                    ;
2795                                    ; Addresses of version 2.9 of ROM based routines.  These are very specific
2796                                    ; addresses which should not be touched.  DW,NH
2797
2798            073450                  dospot  =       073450
2799            074002                  findal  =       074002
2800            071142                  getsec  =       071142
2801            102656                  .pcb    =       102656
2802            067552                  f$rpl   =       067552
2803            076410                  Refcrt  =       076410
2804            076421                  Refcnt  =       076421
2805            074026                  procba  =       074026
2806            034014                  progre  =       034014
2807            074626                  getblo  =       074626
2808            074370                  resdrv  =       074370
2809            034164                  quo.re  =       034164
2810            043172                  desele  =       043172
2811            100206                  reg.7   =       100206
2812            140010                  r$dat   =       140010
2813            100234                  udc.fl  =       100234
2814            140012                  r$cmd   =       140012
2815            037506                  restor  =       037506
2816            076320                  Refsek  =       076320
2817            066040                  prisum  =       066040
2818            074726                  putmsg  =       074726
2819            076232                  Rsffcu  =       076232
2820            110600                  data    =       110600
2821            044640                  $enq.h  =       044640
2822            140020                  w$dat   =       140020
2823            027222                  fill.i  =       027222
2824            100232                  sd.fla  =       100232
2825            100066                  'd.tab  =       100066
2826            140022                  w$cmd   =       140022
2827            044730                  $deq.h  =       044730
2828            066210                  dopass  =       066210
2829            042652                  select  =       042652
2830            074476                  clrbuf  =       074476
2831            013500                  put.ud  =       013500
2832            100052                  tcbs    =       100052
2833            037506                  restor  =       037506
2834            077472                  cret$   =       077472
2835            077472                  c$ret   =       077472
2836            077456                  csv$    =       077456
2837            077456                  c$sav   =       077456
2838            064152                  Rformat =       064152
2839            070540                  Rdofcmd =       070540
2840            072616                  Rgetman =       072616
2841            064520                  fmtunt  =       064520
2842            071222                  getinp  =       071222
2843            105646                  .pkt    =       105646
2844            100050                  pkts    =       100050
2845
2846                                    ; /* Buffers areas offset from DATA */
2847
2848            000000                  u'boff  =       0.
```

C6

DUPfmt Supplied Program Definitons

```
2849    001004                  uibsiz  =       <512.+4.>
2850    001004                  fiboff  =       <uiboff+uibsiz>
2851    000005                  sumsiz  =       5.
2852    000013                  fibsiz  =       <6.+<sumsiz>>
2853    001032                  datoff  =       <fiboff+<2.*fibsiz>>
2854    001000                  datsiz  =       512.
2855    002032                  trkoff  =       <datoff+datsiz>
2856    024260                  Rtrksiz =       10416.
2857
2858                            ; /* Format Information Block */
2859
2860    000000                  f.curcyl=       0       ;       word    curcyl;
2861    000002                  f.badsur=       2       ;       word    badsur;
2862    000004                  f.badblk=       4       ;       word    badblk;
2863    000006                  f.mode  =       6       ;       word    mode;
2864    000010                  f.contin=       10      ;       word    contin;
2865    000012                  f.man   =       12      ;       word    man_usd;
2866
2867    000000                  no      =       0
2868    000001                  yes     =       1
2869    000002                  RECONSTRUCT     =       2
2870    000370                  hexF8   =       370                     ; Hex F8
2871    000376                  hexFE   =       376                     ; Hex FE
2872    000241                  hexA1   =       241                     ; Hex A1
2873    000100                  op.srp  =       100
2874    000133                  op.rt   -       133
2875
2876                            ; UIB.H Macrotized
2877
2878    000034                  i.sur   =       34      ;
2879    000036                  i.cyl   =       36      ;
2880    000106                  i.spots =       106     ;
2881
2882    000006                  p$work  =       6
2883    000002                  t$ucb   =       2
2884    000030                  t$cyl'n =       30
2885    000032                  t$surfa =       32
2886    000044                  t$buffe =       44
2887    000070                  u$mode  =       70
2888    000072                  u$op.sd =       72
2889
```

DUPfmt Supplied Program Def'nitons

```
2891
2892                                    ; The following is the dup supplied program that is used to format
2893                                    ; dr'ves when us'ng version 2 of the m'crocode.  It 's needed
2894                                    ; because version 2 uses the wrong step 'n and step out values
2895                                    ; during the FCT seek.  Touch th's code at your own risk.
2896                                    ; ....DW and NH
2897
2898
2899                                    .sbttl DUPfmt Suppl'ed Program
2900
2901 052732                             DUPfmt:
2902                                    .dsable AMA
2903 052732  001402                             .word   <DUPend-DUPfmt>         ;Byte count low         TEST HEADER
2904 052734  000000                             .word   0                       ;byte count h'gh
2905 052736  000000                             .word   0                       ;overlay low
2906 052740  000000                             .word   0                       ;overlay high
2907 052742     104     125     120             .asc'   /DUPFMT/                ;6 character asc'z name
     052745     106     115     124
2908                                            .even
2909 052750  000001                             .word   1                       ;vers'on number
2910 052752     000                             .byte   0                       ;flags
2911 052753     177                             .byte   177                     ;t'meout
2912 052754  000240                             nop                             ;start down l'ne loaded test
2913
2914 052756                             DUPsta::
2915 052756  000240                             nop                             ;start down l'ne loaded test
2916
2917                                    ;       Relocate ourselves to upper memory
2918
2919 052760  106427  000340                     mtps    #340                    ; D'sable 'nterrupts
2920 052764  005037  100050                     clr     @#pkts
2921 052770  012746  105646                     mov     #.pkt, (sp)
2922 052774  012746  100050                     mov     #pkts,-(sp)
2923 053000  004737  044640                     call    @#$enq.hea
2924 053004  022626                             cmp     (sp)+,(sp)+
2925 053006  012746  105750                     mov     #.pkt+102,-(sp)
2926 053012  012746  100050                     mov     #pkts,-(sp)
2927 053016  004737  044640                     call    @#$enq.hea
2928 053022  022626                             cmp     (sp)+,(sp)+
2929 053024  106427  000000                     mtps    #0                      ; Enable 'nterrupts
2930
2931 053030  012700  001256                     mov     #<DUPend-DUPrest>,r0    ; Number of bytes
2932 053034  010701                             mov     pc,r1                   ;
2933 053036  062701  000020                     add     #DUPrest-.,r1           ; Starting address (From)
2934 053042  012702  106052                     mov     #.pkt+204,r2            ; Top of memory (To)
2935 053046  010203                             mov     r2,r3                   ; Starting address
2936 053050  112122                     1$:     movb    (r1)+,(r2)+             ; Start copying to upper memory
2937 053052  077002                             sob     r0,1$                   ; Done comp'ng
2938
2939 053054  000113                             jmp     (r3)                    ; Start runn'ng there
2940
2941                                    ;       Executable code starts here
2942
2943 053056                             DUPrest:
2944 053056  004537  077456                     jsr     r5,@#cs.$
2945 053062  012746  100052                     mov     #tcbs,-(sp)
2946 053066  004737  044730                     call    @#$deq.hea
```

E6

DUPfmt Supplied Program

```
2947 053072  005726                          tst     (sp)+
2948 053074  010003                          mov     r0,r3
2949 053076  012746  011000                  mov     #11000,-(sp)
2950 053102  012746  110600                  mov     #data,-(sp)
2951 053106  004737  074476                  call    @#clrbuf
2952 053112  022626                          cmp     (sp)+,(sp)+
2953 053114  012704  110600                  mov     #data,r4
2954 053120  012702  111604                  mov     #data+1004,r2
2955 053124  012712  177777                  mov     #177777,(r2)
2956 053130  010446                          mov     r4,-(sp)
2957 053132  010246                          mov     r2,-(sp)
2958 053134  010346                          mov     r3, (sp)
2959 053136  004737  071222                  call    @#getinp
2960 053142  062706  000006                  add     #6,sp
2961 053146  010065  177770                  mov     r0,-10(r5)
2962 053152  005765  177770                  tst      10(r5)
2963 053156  001011                          bne     2$
2964 053160  010446                          mov     r4,-(sp)
2965 053162  010246                          mov     r2,-(sp)
2966 053164  010346                          mov     r3, (sp)
2967 053166  004737  064520                  call    @#fmtunt
2968 053172  062706  000006                  add     #6,sp
2969 053176  010065  177770                  mov     r0, 10(r5)
2970 053202  005765  177770          2$:     tst     -10(r5)
2971 053206  001011                          bne     3$
2972 053210  010446                          mov     r4,-(sp)
2973 053212  010246                          mov     r2,-(sp)
2974 053214  010346                          mov     r3, (sp)
2975 053216  004767  000314                  call    getman
2976 053222  062706  000006                  add     #6,sp
2977 053226  010065  177770                  mov     r0,-10(r5)
2978 053232                          3$:
2979 053232  000137  064326                  jmp     @#<Rformat+154>
2980
```

F6

DUPfmt Supplied Program

```
2982
2983 053236                                    dofcmd::
2984 053236  004537  077456                            jsr     r5,@#csv$
2985 053242  016504  000006                            mov     6(r5),r4
2986 053246  010446                            mov     r4,-(sp)
2987 053250  016546  000012                            mov     12(r5),-(sp)
2988 053254  016546  000010                            mov     10(r5),-(sp)
2989 053260  004767  000004                            call    dofsek
2990 053264  000137  070566                            jmp     @#<Rdofcmd+26>
2991
2992 053270                                    dofsek::
2993 053270  004537  077456                            jsr     r5,@#csv$
2994 053274  016504  000004                            mov     4(r5),r4
2995 053300  016503  000006                            mov     6(r5),r3
2996 053304  021427  177777                            cmp     (r4),#177777
2997 053310  001006                            bne     44$
2998 053312  005014                            clr     (r4)
2999 053314  016546  000010                            mov     10(r5),-(sp)
3000 053320  004737  037506                            call    @#restore
3001 053324  005726                            tst     (sp)+
3002 053326                                    44$:
3003 053326  005065  177770                            clr     -10(r5)
3004 053332                                    46$:
3005 053332  021403                            cmp     (r4),r3
3006 053334  001475                            beq     45$
3007 053336  021403                            cmp     (r4),r3
3008 053340  002016                            bge     47$
3009 053342  010302                            mov     r3,r2
3010 053344  161402                            sub     (r4),r2
3011 053346  000410                            br      52$
3012 053350                                    53$:
3013 053350  005037  100234                            clr     @#udc.flag
3014 053354  012737  000005  140022                    mov     #5,@#w$cmd
3015 053362                                    55$:
3016 053362  005737  100234                            tst     @#udc.flag
3017 053366  001775                            beq     55$
3018
3019 053370                                    54$:
3020 053370                                    51$:
3021 053370                                    52$:
3022 053370  005302                            dec     r2
3023 053372  002366                            bge     53$
3024
3025 053374                                    50$:
3026 053374  010314                            mov     r3,(r4)
3027 053376                                    47$:
3028 053376  021403                            cmp     (r4),r3
3029 053400  003416                            ble     56$
3030 053402  011402                            mov     (r4),r2
3031 053404  160302                            sub     r3,r2
3032 053406  000410                            br      61$
3033 053410                                    62$:
3034 053410  005037  100234                            clr     @#udc.flag
3035 053414  012737  000007  140022                    mov     #7,@#w$cmd
3036 053422                                    64$:
3037 053422  005737  100234                            tst     @#udc.flag
3038 053426  001775                            beq     64$
```

G6

DUPfmt Supplied Program

```
3039 053430                                63$:
3040 053430                                60$:
3041 053430                                61$:
3042 053430  005302                              dec     r2
3043 053432  002366                              bge     62$
3044 053434                                57$:
3045 053434  010314                              mov     r3,(r4)
3046 053436                                56$:
3047 053436                                66$:
3048 053436  012737  000111  140022              mov     #111,@#w$cmd
3049 053444  033727  140010  000040              bit     @#r$dat,#40
3050 053452  001001                              bne     65$
3051 053454                                67$:
3052 053454  000770                              br      66$
3053 053456                                65$:
3054 053456  005703                              tst     r3
3055 053460  003422                              ble     70$
3056 053462  012737  000111  140022              mov     #111,@#w$cmd
3057 053470  033727  140010  000020              bit     @#r$dat,#20
3058 053476  001413                              beq     71$
3059 053500  005014                              clr     (r4)
3060 053502  005265  177770                      inc     -10(r5)
3061 053506  016500  177770                      mov     -10(r5),r0
3062 053512  020027  000003                      cmp     r0,#3
3063 053516  003403                              ble     72$
3064 053520  012700  076320                      mov     #Refsek,r0
3065 053524  000402                              br      43$
3066 053526                                72$:
3067 053526                                71$:
3068 053526                                70$:
3069 053526  000701                              br      46$
3070 053530                                45$:
3071 053530  005000                              clr     r0
3072 053532                                43$:
3073 053532  000137  077472                      jmp     @#cret$
3074
```

H6

DUPfmt Supplied Program

```
3076
3077                                    ;       GETMAN - Get Manuf Bad Block Info
3078
3079 053536                             getman:                                 ; ** Entry Point is GETMAN **
3080 053536  004537  077456                        jsr     r5,@#csv$           ; Save some
3081 053542  112737  000001  102665              movb    #1,@#.pcb+p$work+1      ; Set pass = 1
3082 053550  016502  000004                        mov     4(r5),r2            ; tcb
3083 053554  016503  000006                        mov     6(r5),r3            ; fib
3084 053560  016504  000010                        mov     10(r5),r4           ; u'b
3085 053564  005063  000012                        clr     f.man(r3)           ; Assume FIB.man_usd = no
3086 053570  026327  000006  000002              cmp     f.mode(r3),#RECONSTRUCT ; FIB.mode = RECONSTRUCT?
3087 053576  001002                              bne     1$                      ;
3088 053600  000137  073014                        jmp     @#<Rgetman+176>     ; If so, just ex't
3089 053604  005062  000032             1$:      clr     t$surface(r2)           ; TCB.surface = 0
3090 053610                             getmlp:                                 ; ** Top of Loop **
3091 053610  012762  112632  000044              mov     #<data+trkoff>,t$buffer(r2) ; TCB.buffer = data + trkoff
3092 053616  116237  000032  102664              movb    t$surface(r2),@#.pcb+p$work+0 ; Set up progress counter
3093 053624  005037  102666                        clr     @#.pcb+p$work+2         ;
3094 053630  004767  000052                        call    rdman               ; Read a Track and Check it
3095 053634  004737  074370                        call    @#resdrv            ; Restore drive
3096 053640  005700                              tst     r0                      ; Failed?
3097 053642  001017                              bne     getmck                  ; If so, forget this
3098 053644  012763  000001  000012              mov     #yes,f.man(r3)          ; Else, FIB.man_usd = yes
3099 053652  004737  074026                        call    @#procbad           ; ... process the bad ones
3100 053656  005700                              tst     r0                      ; ... if something mucked up
3101 053660  001402                              beq     getmnx
3102 053662  000137  073016                        jmp     @#<Rgetman+200>     ; ... then out we go
3103 053666                             getmnx:                                 ; ** Get the Next Track **
3104 053666  005262  000032                        inc     t$surface(r2)           ; Increment TCB.surface & loop
3105 053672  026264  000032  000034              cmp     t$surface(r2),i.sur(r4) ; ... if TCB.surface < UIB.sur
3106 053700  002743                              blt     getmlp                  ; ** Bottom of Loop **
3107 053702                             getmck:                                 ; ** Final Check **
3108 053702  000137  072752                        jmp     @#<Rgetman+134>
3109
3110
```

```
3112
3113 053706                                rdman::                         ; ** Entry Point is RDMAN **
3114 053706  010446                                mov     r4,-(sp)        ; Save UIB for now
3115 053710  012746  000004                        mov     #4.,-(sp)       ; Retry Count
3116 053714  016446  000036                        mov     i.cyl(r4),-(sp) ; sp = UIB.cyl - 1
3117 053720  005316                                dec     (sp)            ; ...
3118 053722                                rdmtry:                         ; ** Retry Loop **
3119 053722  005366  000002                        dec     2(sp)           ; decrement retry count
3120 053726  001577                                beq     rdmex           ; if this is it, exit
3121 053730  016204  000002                        mov     t$ucb(r2),r4    ; ucb = TCB.ucb
3122 053734  010446                                mov     r4,-(sp)        ; select( ucb )
3123 053736  004737  042652                        call    @#select        ; ...
3124 053742  005726                                tst     (sp)+           ; ...
3125 053744  012737  000100  140022                mov     #<op.srp>,@#w$cmd ; Set up UDC registers
3126 053752  016200  000044                        mov     t$buffer(r2),r0 ; Set DMA pointer (TCB.buffer)
3127 053756  012701  140020                        mov     #w$dat,r1       ; r1 = pointer to w$dat
3128 053762  010011                                mov     r0,(r1)         ; reg0 = lowest byte of buffer
3129 053764  000300                                swab    r0              ; ...
3130 053766  010011                                mov     r0,(r1)         ; reg1 = middle byte of buffer
3131 053770  005011                                clr     (r1)            ; reg2 = highest byte of buffer
3132 053772  005011                                clr     (r1)            ; reg3 = desired sector number
3133 053774  016211  000032                        mov     t$surface(r2),(r1) ; reg4 = TCB.surface
3134 054000  011611                                mov     (sp),(r1)       ; reg5 = TCB.cylinder
3135 054002  012711  000001                        mov     #1,(r1)         ; reg6 = sector count
3136 054006  013711  100206                        mov     @#reg.7,(r1)    ; reg7 = retry count
3137 054012  016411  000070                        mov     u$mode(r4),(r1) ; reg8 = mode
3138 054016  016446  000072                        mov     u$op.sd(r4),-(sp) ; put_udc( UCB.op_sd)
3139 054022  004737  013500                        call    @#put.udc       ; ...
3140 054026  005726                                tst     (sp)+           ; ...
3141 054030  012713  177777                        mov     #-1.,(r3)       ; FIB.curcyl = -1
3142 054034  010346                                mov     r3,-(sp)        ; dofcmd( op,ucb,
3143                                                                        ; ...      &curcyl,TCB.cylinder )
3144 054036  010446                                mov     r4,-(sp)        ; ... (TCB.cylinder already on stack)
3145 054040  012746  000133                        mov     #op.rt,-(sp)    ; Read a track
3146 054044  004767  177166                        call    dofcmd          ; Do it
3147 054050  062706  000006                        add     #6,sp           ; Pop 'em
3148 054054  005700                                tst     r0              ; Did that work?
3149 054056  001321                                bne     rdmtry          ; If not, retry
3150 054060  012700  000044                        mov     #t$buffer,r0    ; Check Id Field
3151 054064  060200                                add     r2,r0           ; ...
3152 054066                                findid:                         ; ** Find ID Field **
3153 054066  005063  000002                        clr     f.badsur(r3)    ; Init Bad spots per surface
3154 054072  005063  000004                        clr     f.badblk(r3)    ; Init Bad blocks per surface
3155 054076  004737  074002                        call    @#finda1        ; Look for hex A1
3156 054102  020027  000001                        cmp     r0,#1           ; ...
3157 054106  001705                                beq     rdmtry          ; If not found, retry
3158 054110  012701  000376                        mov     #hexFE,r1       ; Next byte =
3159 054114  116604  000001                        movb    1(sp),r4        ; ... TCB.cylinder[8:11] xor FE ?
3160 054120  074104                                xor     r1,r4           ; ...
3161 054122  122004                                cmpb    (r0)+,r4        ; If not,
3162 054124  001360                                bne     findid          ; ... look for id again
3163 054126  122016                                cmpb    (r0)+,(sp)      ; Next byte = TCB.cylinder[0:7] ?
3164 054130  001356                                bne     findid          ; If not, look for id again
3165 054132  122062  000032                        cmpb    (r0)+,t$surface(r2) ; Next byte = TCB.surface?
3166 054136  001353                                bne     findid          ; If not, look for id again
3167
3168 054140  062700  000003                        add     #3,r0           ; Bump past sector and crcs
```

```
3169 054144  004737  074002                call    @#finda1                ; look for another #1
3170 054150  020027  000001                cmp     r0,#1                   ; ...
3171 054154  001662                        beq     rdmtry                  ; If not found, retry
3172 054156  122027  000370                cmpb    (r0)+,#hexF8            ; Next byte = F8 ?
3173 054162  001341                        bne     findid                  ; If not, look for id again
3174 054164                         rdnxt:                                 ; ** Top of Spot Loop **
3175 054164  005746                        tst     -(sp)                   ; Reserved some room
3176 054166  112016                        movb    (r0)+,(sp)              ; Save Spot
3177 054170  112066  000001                movb    (r0)+,1(sp)             ; ...
3178 054174  011601                        mov     (sp),r1                 ; ...
3179 054176  112016                        movb    (r0)+,(sp)              ; Save Position
3180 054200  112066  000001                movb    (r0)+,1(sp)             ; ...
3181 054204  012604                        mov     (sp)+,r4                ; ...
3182 054206  020127  024260                cmp     r1,#Rtrks*z             ; Spot Out of Range?
3183 054212  101325                        bhi     find'd                  ; If so, skip to find next id
3184 054214  005701                        tst     r1                      ; Spot 0 and Position 0 both zero
3185 054216  001002                        bne     rdspot                  ; ...
3186 054220  005704                        tst     r4                      ; ...
3187 054222  001440                        beq     rdmok                   ; If so, skip to exit with success
3188
3189 054224                         rdspot:                                ; ** Process this spot **
3190 054224  010446                        mov     r4, (sp)                ; Check surface
3191 054226  042716  177760                bic     #177760,(sp)            ; ...
3192 054232  026226  000032                cmp     t$surface(r2),(sp)+     ; If surface doesn't match
3193 054236  001313                        bne     findid                  ; ... skip to find next id
3194
3195 054240  006204                        asr     r4                      ; Check cylinder
3196 054242  006204                        asr     r4                      ; ...
3197 054244  006204                        asr     r4                      ; ...
3198 054246  006204                        asr     r4                      ; ...
3199 054250  005704                        tst     r4                      ; If ( (cylinder <= 0)
3200 054252  003705                        ble     find'd                  ; ...  OR
3201 054254  020416                        cmp     r4,(sp)                 ; ...  (cylinder > max cylinder) )
3202 054256  003303                        bgt     findid                  ; ... skip to find next id
3203
3204 054260  010462  0c0030                mov     r4,t$cylinder(r2)       ; Otherwise, set temp TCB.cylinder
3205 054264  016604  .00004                mov     4(sp),r4                ; Reset UIB
3206 054270  026364  000002  000106        cmp     f.badsur(r3),i.spots(r4) ; Already reached limit?
3207 054276  002211                        bge     rdmtry                  ; If so, this is no good (r0 > 0)
3208
3209 054300  010046                        mov     r0,-(sp)                ; Save buffer pointer
3210 054302  004737  073450                call    @#dospot                ; Save block(s) for this spot
3211 054306  012600                        mov     (sp)+,r0                ; Restore buffer pointer
3212 054310  005263  000002                inc     f.badsur(r3)            ; Increment Bad Spot Counter
3213 054314  026327  000002  000100        cmp     f.badsur(r3),#64.       ; ...
3214 054322  002720                        blt     rdnxt                   ; ...
3215 054324  005000                 rdmok: clr     r0                      ; ** Show success **
3216 054326                         rdmex:                                 ; ** Exit **
3217 054326  022626                        cmp     (sp)+,(sp)+             ; Pop two
3218 054330  012604                        mov     (sp)+,r4                ; Restore UIB
3219 054332  000207                        return                          ; ...
3220
```

DUPfmt Supplied Program

```
3222
3223
3224                                    .enable AMA
3225 054334                             DUPend::
3226
```

DUPfmt Suppl'ed Program

```
3228
3229                                    ;*****************************************************************
3230                                    ;                       AUTOsizer
3231                                    ;       This rout'ne runs the Execute Supplied program called AUTOSZ
3232                                    ; This program is downline loaded into the controller to determ'ne
3233                                    ; which drive 's out in the controller. First you must tell which dr've
3234                                    ; you want to format. After listing the drive number the program w'll load
3235                                    ; the program and f'gure which DEC dr've 't is and wh'ch UIT to load into
3236                                    ; the d'sk controller fo rthe format program.
3237                                    ;
3238                                    ;*****************************************************************
3239
3240  4334                              AUTOSizer:
3241 054334                                 excSUPprg   #Autosz,#<Autoend-Autosz>;downline load the program autosz
     054334  032737  100000  002534  ESP4:   b't      #bit15,cmdrng+2          ;test ownership of r'ng make sure we
                                                                               ;own it
     054342  001374                         bne      ESP4                     ;'f we don't own 't wa't unt'l we do
     054344  012737  000050  002446         mov      #50,cmdlen               ;load length of packet to be sent
     054352  112737  000000  002450         movb     #0,cmdlen+2              ;load msg type and cred't value
     054360  112737  000002  002451         movb     #dup.id,cmdlen+3         ;load DUP connect'on ID
     054366  005237  002452                 inc      cmdpak
     054372  005037  002454                 clr      CMDpak+2
     054376  005037  002456                 clr      CMDpak+4
     054402  005037  002460                 clr      CMDpak+6
     054406  012737  000002  002462         mov      #op.esp,CMDpak+10        ;load up opcode
     054414  012737  000000  002464         mov      #0,CMDpak+12             ;no stand alone modif'er
     054422  012737  001162  002466         mov      #<Autoend-Autosz>,cmdpak+14     ;load length of prg into buffer
     054430  005037  002470                 clr      cmdpak+16
     054434  012737  055010  002472         mov      #Autosz,cmdpak+20        ;starting address of downline load prg
     054442  005037  002474                 clr      CMDpak+22
     054446  005037  002476                 clr      CMDpak+24
     054452  005037  002500                 clr      CMDpak+26
     054456  005037  002502                 clr      CMDpak+30
     054462  005037  002504                 clr      CMDpak+32
     054466  005037  002506                 clr      CMDpak+34                ;overlay buffer descr'ptor
     054472  005037  002510                 clr      CMDpak+36
     054476  005037  002512                 clr      CMDpak+40
     054502  005037  002514                 clr      CMDpak+42
     054506  005037  002516                 clr      CMDpak+44
     054512  005037  002520                 clr      CMDpak+46
     054516  012777  054560  125602         mov      #RFD4,@vector            ;New vector place
     054524  012737  002352  002526         mov      #rsppak,rsprng           ;load response packet area into ring
     054532  012737  002452  002532         mov      #cmdpak,cmdrng           ;load command packet area into ring
     054540  012737  140000  002530         mov      #140000,RSPRNG+2         ;Port ownership bit.
     054546  012737  100000  002534         mov      #b't15,CMDRNG+2
     054554  004737  047376                 jsr      pc,POLLWT                ;Go to poll and wait routine.
                                    ;*****************************************************************
     054560                             RFD4:                                 ;Intr to here.
     054560  062706  000006                 add      #6,sp                    ;fix stack for interrupt (4), pollwt
                                                                               ;subrtn (2)
     054564  012777  065360  125534         mov      #intsrv,@vector          ;Change vector
     054572  004737  060352                 jsr      pc,RSPCHK                ;Go to rout'ne that w'll check on
                                                                               ;the response recvd from the mut.
3242 054576                             Recvdata    #msg,#msglen             ;get results of auto s'ze
     054576  032737  100000  002534  RCD5:   b't      #b't15,cmdrng+2          ;test ownership of r'ng make sure we
                                                                               ;own it
     054604  001374                         bne      RCD5                     ;'f we don't own 't wa't unt'l we do
```

M6

DUPfmt Supplied Program

```
054606  012737  000034  002446          mov     #34,cmdlen              ;load length of packet to be sent
054614  112737  000000  002450          movb    #0,cmdlen+2             ;load msg type and cred't
054622  112737  000002  002451          movb    #dup.id,cmdlen+3        ;load DUP connection ID
054630  005237  002452                  inc     cmdpak                  ;load new CRN
054634  005037  002454                  clr     cmdpak+2
054640  005037  002456                  clr     cmdpak+4
054644  005037  002460                  clr     cmdpak+6
054650  012737  000005  002462          mov     #op.rec,cmdpak+10       ;load up opcode
054656  005037  002464                  clr     cmdpak+12               ;no modifiers
054662  012737  000014  002466          mov     #msglen,cmdpak+14
054670  005037  002470                  clr     cmdpak+16
054674  012737  056156  002472          mov     #msg,cmdpak+20  ;load address of buffer descr'ptor
054702  005037  002474                  clr     cmdpak+22
054706  005037  002476                  clr     cmdpak+24
054712  005037  002500                  clr     cmdpak+26
054716  005037  002502                  clr     cmdpak+30
054722  005037  002504                  clr     cmdpak+32

054726  012777  054770  125372          mov     #RFD5,@vector           ;New vector place
054734  012737  002352  002526          mov     #rsppak,rsprng          ;load response packet area into ring
054742  012737  002452  002532          mov     #cmdpak,cmdrng          ;load command packet area into ring
054750  012737  140000  002530          mov     #140000,RSPRNG+2        ;Port ownership b't.
054756  012737  100000  002534          mov     #bit15,CMDRNG+2
054764  004737  047376                  jsr     pc,POLLWT               ;Go to poll and wa't routine.

                        ;*****************************************************************

054770                          RFD5:                                   ;Intr to here.
054770  062706  000006                  add     #6,sp                   ;fix stack for interrupt (4), pollwt
                                                                        ;subrtn (2)
054774  012777  065360  125324          mov     #intsrv,@vector         ;Change vector
055002  004737  060352                  jsr     pc,RSPCHK               ;Go to routine that will check on
                                                                        ;the response recvd from the mut.
                                                                        ;it w'll check the cmd ref
                                                                        ;num, the endcode and status.
3243 055006  000207                     rts     pc                      ;return
```

```
                        .sbttl AUTOSZ

                        ;*****************************************************************
                        ;                       AUTOsz
                        ;       This is the actual down line loaded code which is placed in
                        ; the RAM inside the RQDX3 controller. This code firgures out the
                        ; cylinder size of the drive. From the cylinder size we can determine
                        ; which drive it is. If the drive is a winchester we will step the drive
                        ; 'nto the inner most cylinder. The inner most cylinder for most drives
                        ; is the parking cyl'nder.
                        ;
                        ;+
                        ; AUTOsz - Determine Dr've Type and Size
                        ;
                        ; Input:        None.
                        ;
                        ; Output:
                        ;               A Special Type Message:
                        ;
                        ;               +-------------------+------------+
                        ;               )   Special Msg #10 (decimal)   ) +00
                        ;               +-------------------+------------+
                        ;               )          Status               ) +02
                        ;               +--------------------------------+
                        ;               ) Innermost Cylinder for Unit 0 ) +04
                        ;               +--------------------------------+
                        ;               ) Innermost Cylinder for Unit 1 ) +06
                        ;               +--------------------------------+
                        ;               ) Innermost Cylinder for Unit 2 ) +10
                        ;               +--------------------------------+
                        ;               ) Innermost Cylinder for Unit 3 ) +12
                        ;               +--------------------------------+
                        ;
                        ;               where,  status      =   0 for success,
                        ;                                       1 for UDC never went done,
                        ;                                       2 for UDC never interrupted,
                        ;                                       3 for Seek Failed
                        ;
                        ;                       cylinder    =   3 for RX33 Floppy
                        ;                                       2 for RX50 Floppy
                        ;                                       0 to 2048 for Winnie,
                        ;                                       -1 for Non-existent unit
                        ;
                        ; Note: The Unit Numbers will correspond to the numbers that the Host
                        ;       would use (i.e., not necessarily the DRVSEL numbers).  Thus,
                        ;       Winnies will always precede Flopp'es and "null devices".
                        ;-
                        ;*****************************************************************

055010                  AUTOsz:
                        .dsable AMA
055010  001162                .word   <AUTOend-AUTOsz>        ;Byte count low         TEST HEADER
055012  000000                .word   0                       ;byte count high
055014  000000                .word   0                       ;overlay low
055016  000000                .word   0                       ;overlay high
055020     101  125  124       .ascii  /AUTOSZ/                ;6 character asc'z name
```

[illegible]

```
      055023     117     123     132
 3302                                          .even
 3303 055026  000001                           .word   1                       ;version number
 3304 055030     000                           .byte   0                       ;flags
 3305 055031     177                           .byte   177                     ;timeout
 3306 055032  000240                           nop                             ;start down line loaded test
 3307
 3308 055034                           AUTO::
 3309 055034  000240                           nop                             ;start down line loaded test
 3310
 3311                                  ; Executable Code Starts Here
 3312
 3313 055036  106427  000340                   mtps    #ps7                    ; Set up our own interrupts handlers
 3314 055042  005037  140004                   clr     @#w$fpl                 ; clear the leds
 3315 055046  013746  100002                   mov     @#i$udc,-(sp)           ; Save the MSCP handlers - UDC
 3316 055052  013746  100006                   mov     @#i$clk,-(sp)           ; ... Clock
 3317 055056  013746  100016                   mov     @#i$sec,-(sp)           ; ... Sector
 3318
 3319                                  ; Taken from RQDX3.MAC m$init code:
 3320
 3321 055062  112737  000000  140022           movb    #u.res,@#w$cmd          ; reset the smc9224 chip
 3322 055070  112737  000111  140022           movb    #u.srp+11,@#w$cmd       ; enable interrupts
 3323 055076  112737  000040  140020           movb    #40,@#w$dat             ;
 3324 055104  005067  001042                   clr     s$$bug                  ; assume the bug is not present
 3325 055110  032737  020000  140006           bit     #20000,@#r$fps          ; is the ECO were there?
 3326 055116  001415                           beq     sizset                  ; definitely not
 3327 055120  112737  000001  140022           movb    #u.dd,@#w$cmd           ; deselect all drives
 3328 055126  012700  001000                   mov     #1000,r0                ; wait for a bit
 3329 055132                           sizwt:                                  ; ...
 3330 055132  005300                           dec     r0                      ; ...
 3331 055134  001376                           bne     sizwt                   ; ...
 3332
 3333 055136  032737  020000  140006           bit     #20000,@#r$fps          ; is the ECO were there?
 3334 055144  001002                           bne     sizset                  ; nope
 3335 055146  005267  001000                   inc     s$$bug                  ; say it is
 3336
 3337 055152                           sizset:                                 ; Set up handlers
 3338 055152  010700                           mov     pc,r0                   ; ...
 3339 055154  062700  000646                   add     #<s$$udc-.>,r0          ; Use our own udc handler
 3340 055160  010037  100002                   mov     r0,@#i$udc              ; ...
 3341 055164  010700                           mov     pc,r0                   ; ...
 3342 055166  062700  000674                   add     #<s$$rti-.>,r0          ;
 3343 055172  010037  100006                   mov     r0,@#i$clk              ; Make clock interrupt rti
 3344 055176  010037  100016                   mov     r0,@#i$sec              ; Make sector interrupt rti
 3345 055202  106427  000000                   mtps    #ps0                    ; Make it good
 3346
 3347                                  ; Go Size the Drives
 3348
 3349 055206  010146                           mov     r1,-(sp)                ; Save Registers
 3350 055210  010246                           mov     r2,-(sp)                ; Save Registers
 3351 055212  010346                           mov     r3,-(sp)                ; ...
 3352 055214  010702                           mov     pc,r2                   ; Point to Unit Descriptor Table
 3353 055216  062702  000744                   add     #<msgdat+2>-.,r2        ; ...
 3354 055222  010200                           mov     r2,r0                   ; ...
 3355 055224  012703  000004                   mov     #4.,r3                  ; Initialize all Unit Descriptors
 3356 055230                           siznon:                                 ; ...
 3357 055230  012720  177777                   mov     #-1.,(r0)+              ; ... to "Non-Existant Unit"
```

```
3358 055234  077303                        sob     r3,s'znon               ; ...
3359
3360 055236  012703  000002                mov     #2.,r3                  ; Set Drive Count to logical un't 0
3361
3362 055242                        sizlop::                                ; ** Loop Until We Get All of Them **
3363                                                                       ; **Check if it is a W'nnie**
3364 055242  012737  000010  140002        mov     #bit3,@#rw$pll          ; Set up Pllctl Csr
3365 055250  012737  000104  140022        mov     #u.srp+4,@#w$cmd        ; Set up UDC registers
3366 055256  005037  140020                clr     @#w$dat                 ; ... Head 0
3367 055262  005037  140020                clr     @#w$dat                 ; ... Cyl'nder 0
3368 055266  012737  000110  140022        mov     #u.srp+8.,@#w$cmd       ; ...
3369 055274  012737  000300  140020        mov     #rd.mode,@#w$dat        ; ... Set mode for w'nnie
3370 055302  010300                        mov     r3,r0                   ; Select the Dr've
3371 055304  062700  000044                add     #u.srd,r0               ; ... u.sd.rd=44
3372 055310  004767  000550                jsr     pc,doudc                ; Do JDC command
3373 055314  005700                        tst     r0                      ; Okay?
3374 055316  001402                        beq     s'zfps                  ; Nope, someth'ng 's screwed up
3375 055320  000167  000374                jmp     s'zend
3376 055324                        sizfps:
3377 055324  032737  140000  140006        b't     #b t14+b't15,@#r$fps    ; W'nn'e?
3378 055332  001121                        bne     s'zwin                  ; Yes, go set cyl'nder count
3379
3380 055334                        sizflp:                                 ; ** Check 'f 't 's a Floppy **
3381 055334  012737  000011  140002        mov     #bit0+bit3,@#rw$pll     ; Set Pllctl CSR
3382 055342  112737  000107  140022        movb    #u.srp+7,@#w$cmd        ; Set up UDC reg'sters
3383 055350  112737  000367  140020        movb    #retry,@#w$dat          ; ... retry = 367
3384 055356  010300                        mov     r3,r0                   ; Select the Drive
3385 055360  062700  000054                add     #u.srx,r0               ; ... u.sd.rx=54
3386 055364  004767  000474                jsr     pc,doudc                ; Do UDC command
3387 055370  005700                        tst     r0                      ; Okay?
3388 055372  001152                        bne     sizend                  ; Nope, someth'ng 's screwed up
3389 055374  005004                        clr     r4                      ; Step counter
3390
3391 055376                        steprx:                                 ;** Step In & Out Unt'l Track 0 Found **
3392 055376  020427  000240                cmp     r4,#160.                ; How many t'mes have we step?
3393 055402  002034                        bge     s'zrx                   ; Enough?
3394 055404  112737  000111  140022        movb    #u.srp+9.,@#w$cmd       ; Set up UDC reg'sters
3395 055412  132737  000020  140010        b'tb    #bit4,@#r$dat           ; At track 0?
3396 055420  001025                        bne     s'zrx                   ; Yes, then go check Floppy type
3397 055422  020427  000120                cmp     r4,#80.                 ; Is step counter >= 80 ?
3398 055426  002412                        blt     stepout                 ;
3399 055430  020427  000202                cmp     r4,#130.                ; Is step counter <= 130 ?
3400 055434  003007                        bgt     stepout                 ;
3401 055436  012700  000005                mov     #u.s'1,r0               ; Step 'n one track
3402 055442  004767  000416                jsr     pc,doudc                ; Do UDC command
3403 055446  005700                        tst     r0                      ; Okay?
3404 055450  001123                        bne     s'zend                  ; Nope, someth'ng 's screwed up
3405 055452  000406                        br      stepmore                ;
3406 055454                        stepout:
3407 055454  012700  000007                mov     #u.so1,r0               ; Step out one track
3408 055460  004767  000400                jsr     pc,doudc                ; Do UDC command
3409 055464  005700                        tst     r0                      ; Okay?
3410 055466  001114                        bne     sizend                  ; Nope, something 's screwed up
3411 055470                        stepmore:
3412 055470  005204                        'nc     r4                      ; Increment step counter
3413 055472  000741                        br      steprx                  ; ** Bottom of f'nd track 0 loop **
3414
```

```
3415 055474                            sizrx:                          ; ** Check Floppy type RX50/RX33 **
3416 055474  112737  000111  140022            movb    #u.srp+9.,@#w$cmd       ; Set up UDC registers
3417 055502  132737  000020  140010            bitb    #bit4,@#r$dat           ; At track 0?
3418 055510  001475                            beq     sizdrv                  ;
3419 055512  112737  000104  140022            movb    #u.srp+4,@#w$cmd        ; Set up UDC registers
3420 055520  112737  000001  140020            movb    #1,@#w$dat              ; .. Head =1
3421 055526  010300                            mov     r3,r0                   ; Select the Drive
3422 055530  062700  000054                    add     #u.srx,r0               ; ... u.sd.rx=54
3423 055534  004767  000324                    jsr     pc,doudc                ; Do UDC command
3424 055540  005700                            tst     r0                      ; Okay?
3425 055542  001066                            bne     sizend                  ; Nope, something is screwed up
3426 055544  112737  000111  140022            movb    #u.srp+9.,@#w$cmd       ; Set up UDC registers
3427 055552  132737  000020  140010            bitb    #bit4,@#r$dat           ; At track 0?
3428 055560  001003                            bne     sizrx3                  ; No, it's an RX50
3429 055562  012712  000002                    mov     #2,(r2)                 ; Mark it as an RX50
3430 055566  000444                            br      sizrd                   ;
3431 055570                            sizrx3:
3432 055570  012712  000003                    mov     #3,(r2)                 ; Yes, mark it as an RX33
3433 055574  000441                            br      sizrd                   ; Go do next drive
3434 055576                            sizwin:
3435 055576  005012                            clr     (r2)                    ; It's a Winnie - Set Count to 0
3436
3437 055600  012700  000007                    mov     #u.so1,r0               ; Step out one track
3438 055604  004767  000254                    jsr     pc,doudc                ; Do UDC command
3439 055610  005700                            tst     r0                      ; Okay?
3440 055612  001042                            bne     sizend                  ; Nope, something is screwed up
3441
3442 055614  012700  000003                    mov     #ersek0,r0              ; Assume that seek to 0 failed
3443 055620  112737  000111  140022            movb    #u.srp+9.,@#w$cmd       ; At Cylinder 0?
3444 055626  132737  000020  140010            bitb    #bit4,@#r$dat           ; ...
3445 055634  001431                            beq     sizend                  ; Nope, something's wrong
3446
3447 055636                            sizin:                          ; ** Step In Until Track 0 Found **
3448 055636  005212                            inc     (r2)                    ; Up Cylinder Count
3449 055640  012700  000005                    mov     #u.si1,r0               ; Step In One Cylinder
3450 055644  004767  000214                    jsr     pc,doudc                ; Do UDC Command
3451 055650  005700                            tst     r0                      ; Okay?
3452 055652  001022                            bne     sizend                  ; Nope, something is screwed up
3453
3454 055654  112737  000111  140022            movb    #u.srp+9.,@#w$cmd       ; At Cylinder 0?
3455 055662  132737  000020  140010            bitb    #bit4,@#r$dat           ; If so, skip to bump up
3456 055670  001003                            bne     sizrd                   ; ... descriptors
3457
3458 055672  021227  004000                    cmp     (r2),#2048.             ; SMC Cylinder Limit Reached?
3459 055676  002757                            blt     sizin                   ; ** Bottom of Step In Loop **
3460
3461 055700                            sizrd:                          ; ** This was a Winnie **
3462
3463 055700  062702  000002                    add     #untdsz,r2              ; Bump Pointer to Next Unit Descriptor
3464
3465 055704                            sizdrv:                         ; ** Check Next Drive **
3466 055704  005203                            inc     r3                      ; Up Drive Count
3467 055706  020327  000005                    cmp     r3,#5.                  ; All 4 Drives Checked?
3468 055712  003002                            bgt     sizend                  ; ...
3469 055714  000167  177322                    jmp     sizlop                  ; ** Bottom of Loop **
3470
3471 055720                            sizend:                         ; ** Send Status and Table **
```

AUTOSZ

```
3472 055720  010067  000234                mov     r0,msgdat               ; Save status
3473 055724  012700  000001                mov     #u.dd,r0                ; Deselect Drive
3474 055730  004767  000130                jsr     pc,doudc                ; ...
3475 055734  012603                        mov     (sp)+,r3                ; Pop
3476 055736  012602                        mov     (sp)+,r2                ; ...
3477 055740  012601                        mov     (sp)+,r1                ; ...
3478 055742  106427  000340                mtps    #ps7                    ; Put the MSCP Handlers Back
3479 055746  012637  100016                mov     (sp)+,@#$sec            ; ...
3480 055752  012637  100006                mov     (sp)+,@#$clk            ; ...
3481 055756  012637  100002                mov     (sp)+,@#$udc            ; ...
3482 055762  106427  000000                mtps    #ps0                    ; ...
3483
3484 055766                         sizexi::                               ; ** Okay, talk to the Host **
3485
3486                                ;PutData,msg,msglen - Send Response to Host
3487
3488 055766  010700                        mov     pc,r0                   ;figure the relative address
3489 055770  062700  000166                add     #msg-.,r0               ;... of the buffer
3490 055774  012746  000014                mov     #msglen,-(sp)           ;load length in bytes of the buffer
3491 056000  010046                        mov     r0,-(sp)                ;load relative address of the buffer
3492 056002  013746  000146                mov     @#146,-(sp)             ;load location of routine in microcode
3493 056006  004736                        jsr     pc,@(sp)+               ;call Put Data routine in Ucode
3494 056010  022626                        cmp     (sp)+,(sp)+             ;fix stack
3495
3496                                ; Terminate Supplied Program
3497
3498 056012  013700  000142                mov     @#142,r0                ;load location of routine in microcode
3499 056016  004710                        jsr     pc,(r0)                 ;call Terminate routine in Ucode
3500 056020  000207                        rts     pc                      ; ...
```

```
                                ;+
                                ; UDC Interrupt Handler
                                ;
                                ; Taken from RQDX3.MAC m$udc code:
                                ;-

056022                          s$$udc::                                ; UDC Handler
056022  005767  000124                  tst     s$$bug                  ; is the ECO wire there?
056026  001404                          beq     s$$udi                  ; nope
056030  032737  020000  140006          bit     #20000,@#r$fos          ; is the 9224 interrupt line set?
056036  001011                          bne     s$$rti                  ; if not, must be a bogus interrupt

056040                          s$$udi:                                 ; ...
056040  113746  140012                  movb    @#r$cmd,-(sp)           ; get interrupt status
056044  142716  000035                  bicb    #35,(sp)                ; clear bits of no interest
056050  122726  000240                  cmpb    #240,(sp)+              ; valid status?
056054  001002                          bne     s$$rti                  ; no, it's a bogus interrupt
056056  005267  000072                  inc     s$$flag                 ; set the flag

                                ;+
                                ; Return from Interrupt
                                ;-

056062                          s$$rti::                                ;;; ...
056062  000002                          rti                             ;;; just quit

                                ;+
                                ; DOUDC - Do a UDC Command
                                ;
                                ; This routine sends a commands and waits an interrupt or
                                ; until timer expires.
                                ;
                                ; Input:        r0      = command
                                ; Output:       r0      = 0 for success, non zero for failure
                                ;

        007570                  mseca = 30.*132.                        ; Max Step Rate * some *
                                                                        ;  loop for 7.5 MHz T11 clock

056064                          doudc::                                 ; ** Do a UDC command **
056064  010146                          mov     r1,-(sp)                ; save r1
056066  005067  000062                  clr     s$$flag                 ; Clear udc flag (interrupt pending)
056072  010037  140022                  mov     r0,@#w$cmd              ; Send the command
056076  012700  004000                  mov     #2048.,r0               ; Set the rom timer (max cylinders)

056102                          mswait:                                 ; ** Wait **
056102  012701  007570                  mov     #mseca,r1               ; set one millisecond counter
056106                          msin:                                   ; ** Top of Inner Loop **
056106  005767  000042                  tst     s$$flag                 ; 3.60 udc interrupted
056112  001005                          bne     msend                   ; 1.60 out if udc interrupted
056114  077104                          sob     r1,msin                 ; 2.40  Total: 7.60 @7.5MHz=>
                                                                        ;                8.5457 @6.67MHz
056116  077007                          sob     r0,mswait               ; ** Bottom of Outer Loop **
056120  012700  000002                  mov     #eruint,r0              ; Never Interrupted
056124  000410                          br      douret                  ; ...

056126                          msend:                                  ; ** Interrupt Happened **
```

AUTOSZ

```
3559 056126  012700  000001                mov     #erudon,r0              ; Assume Never Done
3560 056132  013701  140012                mov     @#r$cmd,r1              ; Get the return status
3561 056136  032701  000040                b't     #bit5,r1                ; All done yet?
3562 056142  001401                        beq     douret                  ; If so, pop out of th's
3563
3564 056144  005000                        clr     r0                      ; Assume everything's ok
3565
3566 056146                        douret:                                 ; ** Return **
3567 056146  012601                        mov     (sp)+,r1
3568 056150  000207                        rts     pc                      ; Back to caller
```

SIZER Supplied Program Data

```
                                .sbttl  SIZER Suppl'ed Program Data
                                ;       .psect  c$data
                                ; Special Stuff
056152                          s$$bug: .blkw   1               ; ECO Wire
056154                          s$$flag:.blkw   1               ; UDC flag
                                ; Packet Area
056156      012     140         msg::   .byte   10.,,b.spl      ; F'nal Message
056160                          msgdat: .blkw   5.              ; Status and Un't Descriptor Tab'e
            000014              msglen  = .-msg                 ; Message Length (Byte Count)
            000002              untdsz  = 2.                    ; Un't Descr'ptor Length
                                .enable AMA
056172                          AUTOend:
```

SIZER Supplied Program Data

```
                                ;*****************************************************************
                                ;                       AUTOdisplay
                                ;       This routine will display the results of the autosizers
                                ;  findings. It will say weather the autosizer errored or not and
                                ;  what drives it found.
                                ;
                                ;*****************************************************************

056172                          AUTOdis:
056172  123727  056157  000140            cmpb    msg+1,#.b.spl           ;chec  if Special Message
056200  001401                            beq     1$                      ;if not then no info to print
056202  000207                            rts     pc                      ;so just return

056204  123727  056156  000012  1$:       cmpb    msg,#10.                ;
056212  001401                            beq     2$                      ;check message number
056214  000207                            rts     pc                      ;return if msg number doesn't match
056216                          2$:
056216  005737  056160                    tst     msg+2                   ;test completion status of Autosizer
056222  001457                            beq     24$                     ;if zero no error report the findings
                                                                        ;if not zero then there is an error

                                ; Autosizer Failure Code
056224                              printb  #ASMSG3,msg+2                   ; Print Autosizer Failure Code
056250  023727  056160  000001            cmp     msg+2,#1                ; Is it a UDC never done error ?
056256  001010                            bne     11$                     ; No, check for next code
056260                              printb  #ASMSG4                         ; Yes, Tell error type
056300  023727  056160  000002  11$:       cmp     msg+2,#2                ; Is it a UDC never interrupted error ?
056306  001010                            bne     12$                     ; No, check for next code
056310                              printb  #ASMSG5                         ; Yes, Tell error type
056330  023727  056160  000003  12$:       cmp     msg+2,#3                ; Is it a seek error ?
056336  001010                            bne     13$                     ; No, go reinitialize ctrl
056340                              printb  #ASMSG6                         ; Yes, Tell error type
056360                          13$:
056360  000207                            rts     pc                      ;return

                                ; Autosizer Findings
056362                          24$:
056362                              printb  #ASMSG1                         ; print Autosizer findings
056402  012701  056162                    mov     #msg+4,r1               ; first cylinder entry
056406  005002                            clr     r2                      ; Start with unit number zero
056410  022711  177777          26$:       cmp     #-1.,(r1)               ; Is unit Non-existant ?
056414  001013                            bne     61$                     ; No, check for RX50
056416                              printb  #ASMSG7,R2                      ; Yes, tell it is non-existant
056440  000137  057220                    jmp     20$                     ; ...
056444  022711  000002          61$:       cmp     #2.,(r1)                ; Is unit an RX50 ?
056450  001013                            bne     62$                     ; No, check for RX33
056452                              printb  #ASMSG8,R2                      ; Yes, tell it is an RX50
056474  000137  057220                    jmp     20$                     ; ...
056500  022711  000003          62$:       cmp     #3.,(r1)                ; Is unit an RX33 ?
056504  001013                            bne     63$                     ; No, then it's a Winchester
056506                              printb  #ASMSG9,R2                      ; Yes, tell it is RX33
056530  000137  057220                    jmp     20$                     ; ...
056534                          63$:                                    ; It is a  WINCHESTER
056534                              printb  #ASMSG2,r2,(r1)                 ; Tell it is a Winchester with so many
                                                                        ; cylinders
```

```
3646 056560                            71$:
3647 056560  023711  003102                   cmp     UIT0+UITsiz-2,(r1)      ;if cylinder # equals UIT table # this
3648                                                                           ;is the correct UIT table
3649 056564  001403                           beq     710$
3650 056566  023711  003100                   cmp     UIT0+UITsiz-4,(r1)      ;if cylinder # equals UIT table # this
3651                                                                           ;is the correct UIT table
3652 056572  001012                           bne     72$
3653 056574                            710$:  printb  #DrvTx0                 ;1      rd51
3654 056614  000137  057220                   jmp     20$
3655
3656 056620  023711  003206            72$:   cmp     UIT1+UITsiz-2,(r1)      ;if cylinder # equals UIT table # this
3657                                                                           ;is the correct UIT table
3658 056624  001403                           beq     720$
3659 056626  023711  003204                   cmp     UIT1+UITsiz-4,(r1)      ;if cylinder # equals UIT table # this
3660                                                                           ;is the correct UIT table
3661 056632  001011                           bne     73$
3662 056634                            720$:  printb  #DrvTx1                 ;1      rd52
3663 056654  000561                           br      20$
3664
3665 056656  023711  003312            73$:   cmp     UIT2+UITsiz-2,(r1)      ;if cylinder # equals UIT table # this
3666                                                                           ;is the correct UIT table
3667 056662  001403                           beq     730$
3668 056664  023711  003310                   cmp     UIT2+UITsiz-4,(r1)      ;if cylinder # equals UIT table # this
3669                                                                           ;is the correct UIT table
3670 056670  001011                           bne     74$
3671 056672                            730$:  printb  #DrvTx2                 ;1      rd52
3672 056712  000542                           br      20$
3673
3674 056714  023711  003416            74$:   cmp     UIT3+UITsiz-2,(r1)      ;if cylinder # equals UIT table # this
3675                                                                           ;is the correct UIT table
3676 056720  001403                           beq     740$
3677 056722  023711  003414                   cmp     UIT3+UITsiz-4,(r1)      ;if cylinder # equals UIT table # this
3678                                                                           ;is the correct UIT table
3679 056726  001011                           bne     75$
3680 056730                            740$:  printb  #DrvTx3                 ;1      rd53
3681 056750  000523                           br      20$
3682
3683 056752  023711  003522            75$:   cmp     UIT4+UITsiz-2,(r1)      ;if cylinder # equals UIT table # this
3684                                                                           ;is the correct UIT table
3685 056756  001403                           beq     750$
3686 056760  023711  003520                   cmp     UIT4+UITsiz-4,(r1)      ;if cylinder # equals UIT table # this
3687                                                                           ;is the correct UIT table
3688 056764  001011                           bne     76$
3689 056766                            750$:  printb  #DrvTx4                 ;1      rd54
3690 057006  000504                           br      20$
3691
3692 057010  023711  003626            76$:   cmp     UIT5+UITsiz-2,(r1)      ;if cylinder # equals UIT table # this
3693                                                                           ;is the correct UIT table
3694 057014  001403                           beq     760$
3695 057016  023711  003624                   cmp     UIT5+UITsiz-4,(r1)      ;if cylinder # equals UIT table # this
3696                                                                           ;is the correct UIT table
3697 057022  001011                           bne     77$
3698 057024                            760$:  printb  #DrvTx5                 ;1      rd31
3699 057044  000465                           br      20$
3700
3701 057046  023711  003732            77$:   cmp     UIT6+UITsiz-2,(r1)      ;if cylinder # equals UIT table # this
3702                                                                           ;is the correct UIT table
```

SIZER Supplied Program Data

```
3703 057052  001403                    beq     770$
3704 057054  023711  003730             cmp     UIT6+UITsiz-4,(r1)      ;if cylinder # equals UIT table # this
3705                                                                    ;is the correct UIT table
3706 057060  001011                    bne     78$
3707 057062                    770$:    printb  #DrvTx6                 ;1      rd32
3708 057102  000446                    br      20$
3709
3710 057104  023711  004036     78$:    cmp     UIT7+UITsiz-2,(r1)      ;if cylinder # equals UIT table # this
3711                                                                    ;is the correct UIT table
3712 057110  001403                    beq     780$
3713 057112  023711  004034             cmp     UIT7+UITsiz-4,(r1)      ;if cylinder # equals UIT table # this
3714                                                                    ;is the correct UIT table
3715 057116  001011                    bne     79$
3716 057120                    780$:    printb  #DrvTx7                 ;1      rd
3717 057140  000427                    br      20$
3718
3719 057142  023711  004142     79$:    cmp     UITdf+UITsiz-2,(r1)     ;if cylinder # equals UIT table # this
3720                                                                    ;is the correct UIT table
3721 057146  001403                    beq     790$
3722 057150  023711  004140             cmp     UITdf+UITsiz-4,(r1)     ;if cylinder # equals UIT table # this
3723                                                                    ;is the correct UIT table
3724 057154  001011                    bne     80$
3725 057156                    790$:    printb  #DrvTxc                 ;1      custom rd
3726 057176  000410                    br      20$
3727
3728 057200                    80$:     printb  #ASMSGr                 ;"Unrecognized Drive"
3729
3730 057220  005721           20$:     tst     (r1)+                   ; Point to next unit descriptor
3731 057222  005202                    inc     r2                      ; Set for next unit
3732 057224  020227  000004             cmp     r2,#MaxDrv              ; Last unit?
3733 057230  001402                    beq     27$                     ; Yes, exit routine
3734 057232  000137  056410             jmp     26$                     ; No, do next unit
3735 057236  000207           27$:     rts     pc                      ;
3736
3737                                   ;*************************************************************************
3738                                   ;
3739                                   ;       This routine builds the UIT table or get the UIT table
3740                                   ; depending who the questions are answered to the manual questions.
3741                                   ; If the unit is a listed or regconizable drive we will use a prebuilt
3742                                   ; UIT table. If not we will have to ask all the questions to build
3743                                   ; a table.
3744                                   ;
3745                                   ;*************************************************************************
3746 057240                    BLDUIT:
3747 057240  032737  100000  002336             bit     #bit15,untflgs
3748 057246  001402                    beq     manbld
3749 057250  000137  057556             jmp     autobld
3750
3751 057254                    manbld: printf  #DrvTxa                 ;print out UIT tables and their
3752                                                                    ;related drives
3753 057274                            printf  #DrvTxb                 ;UIN    Drive
3754 057314                            printf  #DrvTx0                 ;0      rd51
3755 057334                            printf  #DrvTx1                 ;1      rd52
3756 057354                            printf  #DrvTx2                 ;2      etc
3757 057374                            printf  #DrvTx3                 ;3      etc
3758 057414                            printf  #DrvTx4                 ;4
3759 057434                            printf  #DrvTx5
```

SIZER Suppl'ed Program Data

```
3760 057454                                printf  @DrvTx6
3761 057474                                printf  @DrvTx7
3762 057514                                pr'ntf  @DrvTxc
3763
3764 057534                        GMANID unt.nbr,UIN,0,17,0,10,no          ;GET Unit identifier number (0-7)
3765                                                                        ;PLACE IN bits 0-3.
3766                                                                        ;no defaults, person must know what
3767                                                                        ;Unit Ident'f'cat'on number.
3768 057554  000515                        br      uitloc                  ;get correct table address 'nto UITadrs
3769
3770 057556                        autobld:
3771 057556  013700  002330                mov     unit,r0                 ;get unit number
3772 057562  006300                        asl     r0                      ;get the byte offset of tbl
3773 057564  012737  000000  002344  1$:    mov     @0,uin                  ;pick UIT number 0
3774 057572  023760  003102  056162        cmp     UIT0+UITsiz-2,msg+4(r0) ;'f cyl'nder # equals UIT table # this
3775                                                                        ;is the correct UIT table
3776 057600  001503                        beq     2$
3777 057602  012737  000001  002344        mov     @1,uin                  ;pick UIT number 1
3778 057610  023760  003206  056162        cmp     UIT1+UITsiz-2,msg+4(r0) ;if cyl'nder # equals UIT table # this
3779                                                                        ;is the correct UIT table
3780 057616  001474                        beq     2$
3781 057620  012737  000002  002344        mov     @2,uin                  ;p'ck UIT number 2
3782 057626  023760  003312  056162        cmp     UIT2+UITsiz-2,msg+4(r0) ;'f cyl'nder # equals UIT table # this
3783                                                                        ;is the correct UIT table
3784 057634  001465                        beq     2$
3785 057636  012737  000003  002344        mov     @3,uin                  ;pick UIT number 3
3786 057644  023760  003416  056162        cmp     UIT3+UITsiz-2,msg+4(r0) ;if cyl'nder # equals UIT table # this
3787                                                                        ;is the correct UIT table
3788 057652  001456                        beq     2$
3789 057654  012737  000004  002344        mov     @4,uin                  ;pick UIT number 4
3790 057662  023760  003522  056162        cmp     UIT4+UITsiz-2,msg+4(r0) ;if cylinder # equals UIT table # this
3791                                                                        ;is the correct UIT table
3792 057670  001447                        beq     2$
3793 057672  023760  003520  056162        cmp     UIT4+UITsiz-4,msg+4(r0) ;if cyl'nder # equals UIT table # this
3794                                                                        ;is the correct UIT table
3795 057700  001443                        beq     2$                      ;automatic recal feature of th's drive
3796 057702  012737  000005  002344        mov     @5,uin                  ;pick UIT number 5
3797 057710  023760  003626  056162        cmp     UIT5+UITsiz-2,msg+4(r0) ;if cyl'nder # equals UIT table # this
3798                                                                        ;is the correct UIT table
3799 057716  001434                        beq     2$
3800 057720  023760  003624  056162        cmp     UIT5+UITsiz-4,msg+4(r0) ;if cyl'nder # equals UIT table # this
3801                                                                        ;is the correct UIT table
3802 057726  001430                        beq     2$                      ;automat'c recal feature of th's drive
3803 057730  012737  000006  002344        mov     @6,uin                  ;p'ck UIT number 6
3804 057736  023760  003732  056162        cmp     UIT6+UITsiz-2,msg+4(r0) ;if cyl'nder # equals UIT table # this
3805                                                                        ;is the correct UIT table
3806 057744  001421                        beq     2$
3807 057746  012737  000007  002344        mov     @7,uin                  ;p'ck UIT number 7
3808 057754  023760  004036  056162        cmp     UIT7+UITsiz-2,msg+4(r0) ;if cyl'nder # equals UIT table # th's
3809                                                                        ;is the correct UIT table
3810 057762  001412                        beq     2$
3811 057764                                printb  #efunrg                 ;'No UIT table su'table for th's dr've'
3812 060004  000137  074424                jmp     dropunt                 ;drop ur't and end pass
3813 060010                        2$:
3814 060010                        uitloc:
3815 060010  012703  003000                mov     #UIT0,r3                ;r3 conta'ns base address of UI' tables
3816 060014  013702  002344                mov     UIN,r2                  ;get the correct UI' table address
```

```
3817                                                            ;into UITadr register
3818 060020  001403                     beq     11$             ;if UIN=0 then set table to UIT0
3819 060022  062703  000104     10$:    add     #UITsiz,r3      ;else multiply UIT size by the UIN
3820                                                            ;number and add to base address
3821 060026  077203                     sob     r2,10$
3822 060030  010337  002320     11$:    mov     r3,UITadr       ;store the proper address of the UIT table
3823 060034  000137  060042             jmp     cont            ;all done
3824
3825 060040                     tblbld:                         ;We must build a UNIT INFORMATION TABLE
3826 060040  000240                     nop                     ;try IRQCBI for custom built tables
3827                                                            ;available thru SDC.
3828 060042  000207             cont:   rts     pc              ;go back
3829                             ;***************************************************************
3830                             ;
3831                             ;       Octal number to ASCII Decimal number
3832                             ;       r1 = address of ascii decimal data
3833                             ;       r0 = octal data word
3834                             ;***************************************************************
3835 060044                     OCTASC:
3836 060044  010246                     mov     r2,-(sp)
3837 060046  010346                     mov     r3,-(sp)
3838 060050  005002                     clr     r2              ;clear the decimal table pointer
3839 060052  005003             1$:     clr     r3              ;clear decimal digit
3840 060054  005203             2$:     inc     r3              ;increment decimal digit
3841 060056  166200  060116             sub     dectbl(r2),r0   ;subtract a power of ten from accumulator
3842 060062  002374                     bge     2$              ;if not negative subtract another
3843 060064  066200  060116             add     dectbl(r2),r0   ;adjust accumulator so positive
3844 060070  005303                     dec     r3              ;adjust dec'mal d'g't
3845 060072  062703  000060             add     #60,r3          ;convert dec'mal  o ascii
3846 060076  110321                     movb    r3,(r1)+        ;mov ascii dig't text 'nto buffer
3847 060100  005722                     t,t     (r2)+           ;increment table pointer
3848 060102  005762  060116             tst     dectbl(r2)      ;check if thats all
3849 060106  001361                     bne     1$
3850 060110  012603                     mov     (sp)+,r3
3851 060112  012602                     mov     (sp)+,r2
3852 060114  000207                     rts     pc
3853 060116                     dectbl:
3854 060116  023420                     .word   10000.
3855 060120  001750                     .word   1000.
3856 060122  000144                     .word   100.
3857 060124  000012                     .word   10.
3858 060126  000001                     .word   1.
3859 060130  000000                     .word   0
3860                             ;***************************************************************
3861                             ;
3862                             ;       ASCII DECIMAL numbers to  Octal numbers
3863                             ;       r1 = address of ascii decimal data
3864                             ;       r0 = address to store octal data low word, high word
3865                             ;***************************************************************
3866 060132                     ASCDEC:
3867 060132  010546                     mov     r5,-(sp)
3868 060134  010446                     mov     r4,-(sp)
3869 060136  010346                     mov     r3,-(sp)
3870 060140  010246                     mov     r2,-(sp)
3871 060142  005004                     clr     r4
3872 060144  005003                     clr     r3
3873 060146  005002                     clr     r2
```

SIZER Supplied Program Data

```
3874 060150  112104                3$:     movb    (r1)+,r4
3875 060152  001423                        beq     1$                      ;if digit equals null than all done
3876 060154  162704  000060                sub     #60,r4
3877 060160  010346                        mov     r3,-(sp)
3878 060162  010246                        mov     r2,-(sp)                ;save accum
3879
3880 060164  012705  000003                mov     #3,r5                   ;accum * 8
3881 060170  006302                4$:     asl     r2
3882 060172  006103                        rol     r3
3883 060174  077503                        sob     r5,4$
3884
3885 060176  006316                        asl     (sp)                    ;accum*2
3886 060200  006166  000002                rol     2(sp)
3887
3888 060204  000241                        clc                             ; accum*8 + accum*2
3889 060206  062602                        add     (sp)+,r2
3890 060210  005503                        adc     r3
3891 060212  062603                        add     (sp)+,r3
3892
3893 060214  060402                        add     r4,r2                   ;add present digit to accum*10
3894 060216  005503                        adc     r3
3895 060220  000753                        br      3$
3896
3897 060222  010220                1$:     mov     r2,(r0)+                ;load lo number
3898 060224  010310                        mov     r3,(r0)                 ;load hi number
3899
3900 060226  012602                        mov     (sp)+,r2                ;restore stack to its orginal
3901 060230  012603                        mov     (sp)+,r3
3902 060232  012604                        mov     (sp)+,r4
3903 060234  012605                        mov     (sp)+,r5
3904 060236  000207                        rts     pc
3905
3906                                ;*****************************************************************
3907                                ;
3908                                ; This routine types out the ASCII information passed
3909                                ; by the d'sk controller. This ASCII information is
3910                                ; contained in the buffer called DATARE and is offset
3911                                ; by 1 word. To fake the DRS macro routine a "%A" is
3912                                ; placed in front of the text.
3913                                ;*****************************************************************
3914
3915 060240                       typDUPbuf:
3916 060240  012701  002610                mov     #datare,r1              ;get data area address of ascii info
3917 060244  063701  002366                add     rsppak+14,r1            ;add the number of byte transfered
3918 060250  105021                1$:     clrb    (r1)+                   ;put null characters into data buffer arter
3919                                                                       ;end of ASCII info
3920 060252  020127  002734                cmp     r1,#prgnam              ;
3921 060256  001374                        bne     1$                      ;we do this to fake out the DRS macro
3922
3923 060260  112737  000045  002610        movb    #45,datare              ;put the "%" delimiter for the DRS macro
3924 060266  112737  000101  002611        movb    #101,datare+1           ;put the "A" for ascii info for the DRS macro
3925 060274                                printx  #PB13                   ;New Line <cr><lf>
3926 060314                                printx  #datare                 ;print the message returned from the controller
3927
3928 060334                       clrDUPbuf:
3929 060334  012701  002610                mov     #datare,r1              ;clear out entire data area
3930 060340  105021                2$:     clrb    (r1)+                   ;
```

```
3931 060342  020127  002734              cmp     r..,#prgsz
3932 060346  001374                      bne     .$
3933 060350  000207                      rts     pc
3934                            ;*****************************************************************
3935                            ;
3936                            ;       THIS ROUTINE IS TO CHECK ON THE RESPONSE PACKET
3937                            ;       GOODNESS. THE COMMAND REFERENCE NUMBER, THE END CODE
3938                            ;       AND THE STATUS ARE TESTED.
3939                            ;*****************************************************************
3940
3941 060352                     RSPCHK:
3942
3943 060352  013701  002452          mov         cmdpak,r1
3944 060356  013700  002352          mov         rsppak,r0
3945 060362  020001                  cmp         r0,r1                   ;compare CRN numbers
3946 060364  001014                  bne         1$
3947 060366  013701  002462          mov         cmdpak+10,r1
3948 060372  062701  000200          add         #200,r1
3949 060376  013700  002362          mov         rsppak+10,r0
3950 060402  020001                  cmp         r0,r1                   ;compare Opcodes
3951 060404  001004                  bne         1$
3952 060406  013701  002364          mov         rsppak+12,r1            ;check the status
3953 060412  001001                  bne         1$
3954 060414  000207                  rts         pc                      ;if all checks then return
3955
3956                                                                      ;if all doesn't check then
3957                                                                      ;report a bad packet
3958 060416  022701  000004          1$:     cmp     #4,r1
3959 060422  001005                  bne     100$                        ;if status is not 4 for GUS, then
3960                                                                      ;report fatal error 10
3961 060424  022737  000003  002462      cmp     #op.gus,cmdpak+10
3962 060432  001001                  bne     100$
3963 060434  000207                  rts     pc                          ;if status is 4 for GUS, return to
3964                                                                      ;calling program
3965
3966 060436                     100$:           ERRDF   10,df11         ;Bad response packet
3967 060446                     PRNTpkt:
3968 060446                             Printb  #PB11crn,cmdpak,rsppak  ;Expected CRN XXXX ,Received CRN YYYY
3969 060476  005001                    clr     r1                      ;Make sure ID will be properly
3970                                                                     ;represented in r1
3971 060500  113701  002451          movb    cmdlen+3,r1             ;load r1 with DRS or MSCP ID
3972 060504  022701  000002          cmp     #DUP.id,r1              ;Was this a DUP command ?
3973 060510  001402                  beq     99$                     ;if so, check DUP response opcode reply
3974 060512  000137  061344          jmp     191$                    ;Jump to MSCP status code check
3975 060516  013701  002362      99$:mov     rsppak+10,r1            ;check response opcode reply
3976 060522  032701  000200          bit     #200,r1                 ;see if a end command response was send
3977 060526  001010                  bne     2$
3978 060530                          printx  #PB11end                ;No end bit in response packet endcode
3979 060550  022701  000201      2$: cmp     #201,r1
3980 060554  001010                  bne     3$                      ;check if Get Dust Status command
3981 060556                          printx  #PB11GDS
3982 060576  022701  000202      3$: cmp     #202,r1
3983 060602  001010                  bne     4$                      ;check if Execute Supplied Program
3984 060604                          printx  #PB11ESP
3985 060624  022701  000203      4$: cmp     #203,r1
3986 060630  001010                  bne     5$                      ;check if Execute Local Program
3987 060632                          printx  #PB11ELP
```

SIZER Supplied Program Data

```
3988 060652  022701  000204          5$:     cmp     #204,r1
3989 060656  001010                          bne     6$                      ;check if Send Data
3990 060660                                  printx  #PB11SD
3991 060700  022701  000205          6$:     cmp     #205,r1
3992 060704  001022                          bne     7$                      ;check if Receive Data
3993 060706                                  printx  #PB11RD
3994 060726                                  Printb  #PBSF0,r3,r5            ;"type xxx, message number xxxxx is
3995                                                                         ;unknown to this program"
3996 060752  022701  000206          7$:     cmp     #206,r1
3997 060756  001010                          bne     8$                      ;check if Abort Program
3998 060760                                  printx  #PB11AP
3999 061000                          8$:     Printb  #PB11op,cmdpak+10,rsppak+10
4000                                                                         ;CMDpkt opcode XXXX,RSPpkt opcode YYYYY
4001
4002 061030  013701  002364                  mov     rsppak+12,r1            ;find out what kind of status we have
4003 061034  022701  000000                  cmp     #0.,r1
4004 061040  001010                          bne     10$
4005 061042                                  printx  #pb11s0                 ;status:  successful
4006 061062  022701  000001          10$:    cmp     #1.,r1
4007 061066  001010                          bne     11$
4008 061070                                  printx  #pb11s1                 ;status:  Invalid Command
4009 061110  022701  000002          11$:    cmp     #2.,r1
4010 061114  001010                          bne     12$
4011 061116                                  printx  #pb11s2                 ;status:  No Region Available
4012 061136  022701  000003          12$:    cmp     #3.,r1
4013 061142  001010                          bne     13$
4014 061144                                  printx  #pb11s3                 ;status:  No Region Suitable
4015 061164  022701  000004          13$:    cmp     #4.,r1
4016 061170  001010                          bne     14$
4017 061172                                  printx  #pb11s4                 ;status:  Program Not Known
4018 061212  022701  000005          14$:    cmp     #5.,r1
4019 061216  001010                          bne     15$
4020 061220                                  printx  #pb11s5                 ;status:  Load Failure
4021 061240  022701  000006          15$:    cmp     #6.,r1
4022 061244  001010                          bne     16$
4023 061246                                  printx  #pb11s6                 ;status:  Standalone
4024 061266  022701  000011          16$:    cmp     #9.,r1
4025 061272  001010                          bne     19$
4026 061274                                  printx  #pb11s9                 ;status:  Host Buffer Access error
4027 061314                          19$:
4028 061314                                  Printb  #PB11sts,rsppak+12      ;Response packet status XXXX
4029 061340  000137  074424                  jmp     dropunt                 ;drop unit and go on
4030                                 ;*****************************************************************
4031                                 ;
4032                                 ;       The following code was necessary to add a RSPCHK for
4033                                 ;               the MSCP macros.                - GJK
4034                                 ;
4035                                 ;*****************************************************************
4036
4037 061344  013701  002362          191$:   mov     rsppak+10,r1            ;check response packet reply
4038 061350  032701  000200                  bit     #200,r1                 ;see if end command response sent
4039 061354  001010                          bne     192$
4040 061356                                  printx  #MSCPend                ;no end bit in response packet endcode
4041 061376  022701  000203          192$:   cmp     #203,r1
4042 061402  001010                          bne     193$                    ;check if GUS command
4043 061404                                  printx  #MSCPGUS
4044 061424  022701  000204          193$:   cmp     #204,r1
```

SIZER Suppl'ed Program Data

```
4045 061430  001010                    bne     194$                    ;check if SCC command
4046 061432                            pr'ntx  #MSCPSCC
4047 061452  022701  000211    194$:   cmp     #211,r1
4048 061456  001010                    bne     195$                    ;check 'f Online command
4049 061460                            pr'ntx  #MSCPONL
4050 061500  022701  000241    195$:   cmp     #241,r1
4051 061504  001010                    bne     196$                    ;check 'f Read command
4052 061506                            pr'ntx  #MSCPRD
4053 061526  022701  000242    196$:   cmp     #242,r1
4054 061532  001010                    bne     197$                    ;check if Wr'te command
4055 061534                            pr'ntx  #MSCPWRT
4056 061554              197$:         pr'ntx  #MSCPOP,cmdpak+10,rsppak+10
4057                                                                   ;print CMDpak opcode XXXX,
4058                                                                   ;RSPpak opcode YYYY
4059 061604  013701  002364    20$:    mov     rsppak+12,r1            ;find out what k'nd of status we have
4060 061610  122701  000001            cmpb    #1.,r1
4061 061614  001010                    bne     21$
4062 061616                            pr'ntx  #ME10                   ;status:  Invalid Command
4063 061636  022701  000002    21$:    cmp     #2.,r1
4064 061642  001010                    bne     22$
4065 061644                            printx  #ME20                   ;status:  Command Aborted
4066 061664  022701  000C03    22$:    cmp     #3.,r1
4067 061670  001012                    bne     23$
4068 061672                            printb  #ME30,UNIT              ;status:  Unit Offline - Unit Unknown
4069 061716  022701  000043    23$:    cmp     #35.,r1
4070 061722  001012                    bne     24$
4071 061724                            pr'ntb  #ME31,UNIT              ;status:  Unit Offl'ne - Unit Disabled
4072 061750  022701  000103    24$:    cmp     #67.,r1
4073 061754  001012                    bne     25$
4074 061756                            pr'ntb  #ME32,UNIT              ;status:  Unit Offl'ne - Unit Inoperative
4075 062002  022701  000203    25$:    cmp     #131.,r1
4076 062006  001012                    bne     26$
4077 062010                            pr'ntb  #ME34,UNIT              ;slatus:  Un't Of'l'ne - Duplicate Unit
4078                                                                   ;         Number
4079 062034  022701  000403    26$:    cmp     #259.,r1
4080 062040  001012                    bne     27$
4081 062042                            pr'ntb  #ME38,UNIT              ;status:  Unit Offline - Unit Disabled
4082                                                                   ;         by Field Service or D'agnost'c
4083 062066  022701  000004    27$:    cmp     #4.,r1
4084 062072  001012                    bne     30$
4085 062074                            pr'ntb  #ME40,UNIT              ;status:  Un't Available
4086 062120  022701  000245    30$:    cmp     #165.,r1
4087 062124  001012                    bne     31$
4088 062126                            pr'ntb  #ME55,UNIT              ;status:  Med'a Format Error - Not
4089                                                                   ;         Formatted w/512 Byte Sectors
4090 062152  022701  000305    31$:    cmp     #197.,r1
4091 062156  001012                    bne     32$
4092 062160                            pr'ntb  #ME56,UNIT              ;status:  Med'a Format Error - Not
4093                                                                   ;         Formatted or FCT Corrupted
4094 062204  022701  000345    32$:    cmp     #229.,r1
4095 062210  001012                    bne     33$
4096 062212                            pr'ntb  #ME57,UNIT              ;status:  Med'a Format Error -
4097                                                                   ;         Uncorrectable ECC Error
4098 062236  022701  000405    33$:    cmp     #261.,r1
4099 062242  001010                    bne     34$
4100 062244                            pr'ntx  #ME58                   ;status:  Media Format Error - RCT
4101                                                                   ;         Corrupted
```

SIZER Supplied Program Data

```
4102 062264  022701  010006        34$:    cmp     #4102.,r1
4103 062270  001012                        bne     35$
4104 062272                                printb  #ME6128,UNIT          ;status:  Software Write Protected
4105 062316  022701  020006        35$:    cmp     #8198.,r1
4106 062322  001012                        bne     36$
4107 062324                                printb  #ME6256,UNIT          ;status:  Hardware Write Protected
4108 062350  022701  000007        36$:    cmp     #7.,r1
4109 062354  001010                        bne     37$
4110 062356                                printx  #ME70                 ;status:  Compare Error
4111 062376  022701  000010        37$:    cmp     #8.,r1
4112 062402  001010                        bne     40$
4113 062404                                printx  #ME80                 ;status:  Data Error - Force Error
4114                                                                      ;         Modifier Used
4115 062424  022701  000110        40$:    cmp     #72.,r1
4116 062430  001010                        bne     41$
4117 062432                                printx  #ME82                 ;status:  Data Error - Invalid Header
4118 062452  022701  000150        41$:    cmp     #104.,r1
4119 062456  001010                        bne     42$
4120 062460                                printx  #ME83                 ;status:  Data Error - Data Sync Timeout
4121 062500  022701  000210        42$:    cmp     #136.,r1
4122 062504  001010                        bne     43$
4123 062506                                printx  #ME84                 ;status:  Data Error - Correctable
4124                                                                      ;         Error in ECC Field
4125 062526  022701  000350        43$:    cmp     #232.,r1
4126 062532  001010                        bne     44$
4127 062534                                printx  #ME87                 ;status:  Data Error - Uncorrectable
4128                                                                      ;         ECC Error
4129 062554  022701  000410        44$:    cmp     #264.,r1
4130 062560  001010                        bne     45$
4131 062562                                printx  #ME88                 ;status:  Data Error - One Symbol ECC
4132                                                                      ;         Error
4133 062602  022701  000450        45$:    cmp     #296.,r1
4134 062606  001010                        bne     46$
4135 062610                                printx  #ME89                 ;status:  Data Error - Two Symbol ECC
4136                                                                      ;         Error
4137 062630  022701  000510        46$:    cmp     #328.,r1
4138 062634  001010                        bne     47$
4139 062636                                printx  #ME810                ;status:  Data Error - Three Symbol ECC
4140                                                                      ;         Error
4141 062656  022701  000550        47$:    cmp     #360.,r1
4142 062662  001010                        bne     50$
4143 062664                                printx  #ME811                ;status:  Data Error - Four Symbol ECC
4144                                                                      ;         Error
4145 062704  022701  000610        50$:    cmp     #392.,r1
4146 062710  001010                        bne     51$
4147 062712                                printx  #ME812                ;status:  Data Error - Five Symbol ECC
4148                                                                      ;         Error
4149 062732  022701  000650        51$:    cmp     #424.,r1
4150 062736  001010                        bne     52$
4151 062740                                printx  #ME813                ;status:  Data Error - Six Symbol ECC
4152                                                                      ;         Error
4153 062760  022701  000710        52$:    cmp     #456.,r1
4154 062764  001010                        bne     53$
4155 062766                                printx  #ME814                ;status:  Data Error - Seven Symbol ECC
4156                                                                      ;         Error
4157 063006  022701  000750        53$:    cmp     #488.,r1
4158 063012  001010                        bne     54$
```

F8

SIZER Supplied Program Data

```
4159 063014                                    printx  #ME815          ;status:  Data Error - Eight Symbol ECC
4160                                                                   ;         Error
4161 063034  022701  000011          54$:      cmp     #9.,r1
4162 063040  001010                            bne     55$
4163 063042                                    printx  #ME90           ;status:  Host Buf Acc Err - Cause Not
4164                                                                   ;         Available
4165 063062  022701  000051          55$:      cmp     #41.,r1
4166 063066  001010                            bne     56$
4167 063070                                    printx  #ME91           ;status:  Host Buf Acc Err - Odd
4168                                                                   ;         Transfer Address
4169 063110  022701  000111          56$:      cmp     #73.,r1
4170 063114  001010                            bne     57$
4171 063116                                    printx  #ME92           ;status:  Host Buf Acc Err - Odd Byte
4172                                                                   ;         Count
4173 063136  022701  000151          57$:      cmp     #105.,r1
4174 063142  001010                            bne     60$
4175 063144                                    printx  #ME93           ;status:  Host Buf Acc Err -
4176                                                                   ;         Non-Existent Memory Error
4177 063164  022701  000211          60$:      cmp     #137.,r1
4178 063170  001010                            bne     61$
4179 063172                                    printx  #ME94           ;status:  Host Buf Acc Err - Host
4180                                                                   ;         Memory Parity Error
4181 063212  022701  000251          61$:      cmp     #169.,r1
4182 063216  001010                            bne     62$
4183 063220                                    printx  #ME95           ;status:  Host Buf Acc Err - Invalid
4184                                                                   ;         Page Table Entry
4185 063240  022701  000052          62$:      cmp     #42.,r1
4186 063244  001010                            bne     63$
4187 063246                                    printx  #MEA1           ;status:  Controller Err - SERDES
4188                                                                   ;         Overrun or Underrun
4189 063266  022701  000112          63$:      cmp     #74.,r1
4190 063272  001010                            bne     64$
4191 063274                                    printx  #MEA2           ;status:  Controller Err - EDC Error
4192 063314  022701  000152          64$:      cmp     #106.,r1
4193 063320  001010                            bne     65$
4194 063322                                    printx  #MEA3           ;status:  Controller Err - Inconsistent
4195                                                                   ;         Internal Control Structure
4196 063342  022701  000212          65$:      cmp     #138.,r1
4197 063346  001010                            bne     66$
4198 063350                                    printx  #MEA4           ;status:  Controller Err - Internal EDC
4199                                                                   ;         Error
4200 063370  022701  000252          66$:      cmp     #170.,r1
4201 063374  001010                            bne     67$
4202 063376                                    printx  #MEA5           ;status:  Controller Err - LESI Adapter
4203                                                                   ;         Card Parity Err on Input
4204 063416  022701  000312          67$:      cmp     #202.,r1
4205 063422  001010                            bne     70$
4206 063424                                    printx  #MEA6           ;status:  Controller Err - LESI Adapter
4207                                                                   ;         Card Parity Err on Output
4208 063444  022701  000352          70$:      cmp     #234.,r1
4209 063450  001010                            bne     71$
4210 063452                                    printx  #MEA7           ;status:  Controller Err - LESI Adapter
4211                                                                   ;         Card "cable 'n place" Not
4212                                                                   ;         Asserted
4213 063472  022701  000412          71$:      cmp     #266.,r1
4214 063476  001010                            bne     72$
4215 063500                                    printx  #MEA8           ;status:  Controller Err - Controller
```

```
4216                                                    ;           Overrun or Underrun
4217 063520  022701  000452          72$:    cmp     #298.,r1
4218 063524  001010                        bne     73$
4219 063526                                 printx  #MEA9           ;status:  Controller Err - Controller
4220                                                    ;           Memory Error
4221 063546  022701  000053          73$:    cmp     #43.,r1
4222 063552  001012                        bne     74$
4223 063554                                 pr'ntb  #MEB1,UNIT      ;status:  Drive Error - Drive Command
4224                                                    ;           Time Out
4225 063600  022701  000113          74$:    cmp     #75.,r1
4226 063604  001010                        bne     75$
4227 063606                                 pr'ntx  #MEB2           ;status:  Drive Error - Controller
4228                                                    ;           Detected Transmission Error
4229 063626  022701  000153          75$:    cmp     #107.,r1
4230 063632  001010                        bne     76$
4231 063634                                 pr'ntx  #MEB3           ;status:  Drive Error - Position Error
4232 063654  022701  000213          76$:    cmp     #139.,r1
4233 063660  001010                        bne     77$
4234 063662                                 printx  #MEB4           ;status:  Drive Error - Lost Read/Write
4235                                                    ;           Ready During Transfer
4236 063702  022701  000253          77$:    cmp     #171.,r1
4237 063706  001012                        bne     80$
4238 063710                                 pr'ntb  #MEB5,UNIT      ;status:  Drive Error - Drive Clock
4239                                                    ;           Dropout
4240 063734  022701  000313          80$:    cmp     #203.,r1
4241 063740  001010                        bne     81$
4242 063742                                 pr'ntx  #MEB6           ;status:  Dr've Error - Lost Receiver
4243                                                    ;           Ready For Transfer
4244 063762  022701  000353          81$:    cmp     #235.,r1
4245 063766  001012                        bne     82$
4246 063770                                 pr'ntb  #MEB7,UNIT      ;status:  Dr've Error - Dr've Detected
4247                                                    ;           Error
4248 064014  022701  000413          82$:    cmp     #267.,r1
4249 064020  001010                        bne     83$
4250 064022                                 pr'ntx  #MEB8           ;status:  Drive Error - Controller
4251                                                    ;           Detected Pulse or State
4252                                                    ;           Parity Error
4253 064042  022701  000513          83$:    cmp     #331.,r1
4254 064046  001010                        bne     84$
4255 064050                                 printx  #MEB10          ;status:  Dr ve Error - Cortroller
4256                                                    ;           Detected Protocol Error
4257 064070  022701  000553          84$:    cmp     #363.,r1
4258 064074  001012                        bne     85$
4259 064076                                 pr'ntb  #MEB11,UNIT     ;status:  Dr've Error - Drive Failed
4260                                                    ;           In'tializat'on
4261 064122  022701  000613          85$:    cmp     #395.,r1
4262 064126  001012                        bne     86$
4263 064130                                 pr'ntb  #MEB12,UNIT     ;status:  Dr've Error - Drive Igonored
4264                                                    ;           In'tialization
4265 064154  022701  000653          86$:    cmp     #427.,r1
4266 064160  001010                        bne     87$
4267 064162                                 printx  #MEB13          ;status:  Drive Error   Receiver Ready
4268                                                    ;           Collision
4269                                    ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
4270                                    ;
4271                                    ;       The following is appl'cat'on dependent.  During LBN testing, if someone
4272                                    ;       phys'cally opens the drive door, a status of 'On't Ava'lable' 's
```

```
                                        ;       reported for the MSCP WRITE command; therefore, the status will be
                                        ;       reported and execution terminated.  Otherwise, the status will be
                                        ;       reported, and execution will continue.
                                        ;
                                        ;                                                       - GJK
                                        ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
064202  022737  000041  002462  87$:    cmp     #41,cmdpak+10
064210  001001                          bne     88$
064212  000410                          br      999$
064214  022737  000042  002462  88$:    cmp     #42,cmdpak+10
064222  001021                          bne     89$
064224  022737  000004  002364          cmp     #4,rsppak+12            ;Was the status Unit Available due to
                                                                        ;someone physically opening the drive
                                                                        ;door during LBN testing?
064232  001415                          beq     89$                     ;report status and terminate execution
064234                          999$:   Printb  #MSCPsts,rsppak+12      ;Otherwise, print response packet
                                                                        ;status XXXX
064260  005237  002566                  inc     ERRCNT                  ;Update bad block counter
064264  000207                          rts     pc                      ;If MSCP WRITE command, continue until
                                                                        ;all LBNs are tested
064266                          89$:
064266                                  Printb  #MSCPsts,rsppak+12      ;Otherwise, print response packet
                                                                        ;status XXXX
064312  000137  074424                  jmp     dropunt                 ;drop unit and go on

                                        ;**********************************************************************
                                        ;
                                        ;                       BIT FIFTEEN TEST
                                        ;**********************************************************************
064316                          BIT15T:
064316  032714  100000                  bit     #bit15,(r4)
064322  001001                          bne     100$
064324  000207                          rts     pc
064326                          100$:   ERRDF   9,df12                  ;Fatal SA error
064336  011401                          mov     (r4),r1
064340  022701  001000                  cmp     #1000,r1
064344  001010                          bne     1$
064346                                  printx  #pb1201                 ;
064366  022701  100001          1$:     cmp     #100001,r1
064372  001010                          bne     2$
064374                                  printx  #pb1202                 ;
064414  022701  100002          2$:     cmp     #100002,r1
064420  001010                          bne     3$
064422                                  printx  #pb1203                 ;
064442  022701  100003          3$:     cmp     #100003,r1
064446  001010                          bne     4$
064450                                  printx  #pb1204                 ;
064470  022701  100004          4$:     cmp     #100004,r1
064474  001010                          bne     5$
064476                                  printx  #pb1205                 ;
064516  022701  100005          5$:     cmp     #100005,r1
064522  001010                          bne     6$
064524                                  printx  #pb1206                 ;
```

SIZER Supplied Program Data

```
4330 064544  022701  100006        6$:   cmp     #100006,r1
4331 064550  001010                      bne     7$
4332 064552                              printx  #pb1207          ;
4333 064572  022701  100007        7$:   cmp     #100007,r1
4334 064576  001010                      bne     8$
4335 064600                              printx  #pb1208          ;
4336 064620  022701  100010        8$:   cmp     #100010,r1
4337 064624  001010                      bne     9$
4338 064626                              printx  #pb1209          ;
4339 064646  022701  100011        9$:   cmp     #100011,r1
4340 064652  001010                      bne     10$
4341 064654                              printx  #pb1210          ;
4342 064674  022701  100012        10$:  cmp     #100012,r1
4343 064700  001010                      bne     11$
4344 064702                              printx  #pb1211          ;
4345 064722  022701  100013        11$:  cmp     #100013,r1
4346 064726  001010                      bne     12$
4347 064730                              printx  #pb1212          ;
4348 064750  022701  100014        12$:  cmp     #100014,r1
4349 064754  001010                      bne     13$
4350 064756                              printx  #pb1213          ;
4351 064776  022701  100015        13$:  cmp     #100015,r1
4352 065002  001010                      bne     14$
4353 065004                              printx  #pb1214          ;
4354 065024  022701  100016        14$:  cmp     #100016,r1
4355 065030  001010                      bne     15$
4356 065032                              printx  #pb1215          ;
4357 065052  022701  100017        15$:  cmp     #100017,r1
4358 065056  001010                      bne     16$
4359 065060                              printx  #pb1216          ;
4360 065100  022701  100020        16$:  cmp     #100020,r1
4361 065104  001010                      bne     17$
4362 065106                              printx  #pb1217          ;
4363 065126  022701  100021        17$:  cmp     #100021,r1
4364 065132  001010                      bne     18$
4365 065134                              printx  #pb1218          ;
4366 065154  022701  100022        18$:  cmp     #100022,r1
4367 065160  001010                      bne     19$
4368 065162                              printx  #pb1219          ;
4369 065202  022701  100023        19$:  cmp     #100023,r1
4370 065206  001010                      bne     20$
4371 065210                              printx  #pb1220          ;
4372 065230  022701  100024        20$:  cmp     #100024,r1
4373 065234  001010                      bne     21$
4374 065236                              printx  #pb1221          ;
4375 065256  022701  100025        21$:  cmp     #100025,r1
4376 065262  001010                      bne     22$
4377 065264                              printx  #pb1222          ;
4378 065304  022701  100026        22$:  cmp     #100026,r1
4379 065310  001010                      bne     23$
4380 065312                              printx  #pb1223          ;
4381 065332                       23$:
4382 065332                              printb  #pb12,r1         ;SA value: xxxxx
4383 065354  000137  074424              jmp     dropunt          ;drop unit and go on
```

J8

SIZER Supplied Program Data

```
                                ;*****************************************************************************
                                ;       Unexpected Interrupt Server
                                ;
                                ;*****************************************************************************
065360                          intsrv:

                                        ERRSF   8,sf100 ;Fatal SA error
065360                                  docln                   ;do clean up and quit
065370  000137  074424                  jmp     dropunt         ;drop test unit and end pass
065372
```

SIZER Supplied Program Data

```
4397 065376                                BGNPROT
4398 065376  177777                        .WORD -1
4399 065400  177777                        .WORD -1
4400 065402  177777                        .WORD -1
4401 065404                                ENDPROT
4402
4403 065404                               BGNINIT                          ;Sequential example
4404
4405 065404  042737  000100  177546              bic     #bit6,@#LKS       ;make sure clock is off
4406
4407 065412                                 READEF        #EF.CONTINUE     ;Continue command?
4408 065420                                 BCOMPLETE     conton           ;Yes, get no P-table but still initialize
4409 065422                                 READEF        #EF.NEW          ;New pass
4410 065430                                 BNCOMPLETE    next             ;if not new then go to next unit number
4411 065432                               SETUP:
4412 065432  012737  177777  002310              mov     #-1,LOGUNIT       ;Initialize logical unit nbr
4413 065440                               NEXT:
4414 065440  005237  002310                        inc     LOGUNIT          ;Point to next logical unit
4415 065444  023737  002310  002012              cmp     LOGUNIT,L$UNIT    ;Have we passed maximum?
4416 065452  001002                              bne     1$               ;No
4417 065454  000137  065632                      jmp     ABORT            ;Yes, abort the pass
4418 065460                               1$:
4419 065460                                       GPHARD LOGUNIT,PLOC      ;Get the P-table
4420 065472                                       BNCOMPLETE NEXT          ;if not available get next unit
4421
4422 065474  013700  002314                      mov     ploc,r0
4423 065500  010037  002316                      mov     r0,ptbl          ;store the Ptable address for unit
4424 065504  012037  002324                      mov     (r0)+,ipreg      ;store IPreg address into register
4425 065510  012037  002326                      mov     (r0)+,vector     ;store vector
4426 065514  012037  002330                      mov     (r0)+,unit       ;store logical drive number
4427 065520  012037  002334                      mov     (r0)+,sernbr     ;store the serial number
4428 065524  012037  002336                      mov     (r0)+,untflgs
4429
4430 065530  005037  002540              conton: clr     LSTCRN           ;basic initialization stuff
4431 065534  005037  002544                      clr     LSTVCT
4432 065540  005037  002546                      clr     LOPRGI
4433 065544  005037  002550                      clr     HIPRGI
4434
4435 065550  013746  000004              1$:     mov     @#4,-(sp)        ;test to see if controller is there
4436 065554  012737  065570  000004              mov     #$2,@#4
4437 065562  005077  114536                      clr     @IPreg           ;put controller into known state
4438 065566  000410                              br      $3
4439
4440 065570                               $2:     ERRDF   7,DF4            ;NXM trap at controller IP address
4441 065600                                       dodu    LOGUNIT          ;drop unit
4442 065606  000714                              br      next             ;get new unit
4443
4444 065610  012637  000004              $3:     mov     (sp)+,@#4        ;move value back into location 4
4445
4446 065614  012700  000076                      mov     #76,r0           ;clean out all packets and interrupt flags
4447 065620  012701  002346                      mov     #rsp1,r1         ;and the command area
4448 065624  005021                       $4:     clr     (r1)+
4449 065626  077002                              sob     r0,$4
4450
4451 065630  000401                              br      end
4452
4453 065632                               ABORT:
```

SIZER Supplied Program Data

```
4454 065632                                 DOCLN                   ;Do clean-up and abort the pass
4455 065634                           END:
4456 065634                                 ENDINIT                 ;Finished
4457
4458
4459 065636                           BGNAUTO
4460 065636                           DODU LOGUNIT
4461 065644                           ENDAUTO
4462
4463 065646                           BGNCLN
4464 065646  005077  114452                  clr     @IPreg          ;get controller into known state
4465 065652  042737  000100  177546         bic     #bit6,@#LKS     ;make sure clock is off
4466 065660                                 Break                   ;waste some time
4467 065662                           ENDCLN
4468
4469 065664                           BGNDU
4470 065664  042737  000100  177546         bic     #bit6,@#LKS     ;make sure clock is off
4471 065672                                 printf #DRPunt,unit
4472 065716                           ENDDU
4473
```

```
 4475 065720                                BGNTST 1
 4476 065720  042737  000100  177546              bic     #bit6,@#LKS             ;make sure clock is off
 4477 065726  012737  000012  002574              mov     #10.,DELAY              ;load data for 10 second delay
 4478 065734  004737  051522                      jsr     pc,hrdint               ;init the controller
 4479 065740  012737  000074  002574              mov     #60.,DELAY              ;load data for 60 second delay
 4480 065746  032737  010000  002336              bit     #bit12,untflgs          ;check if just want to test a floppy
 4481 065754  001002                              bne     tstdrv
 4482 065756  000137  067672                      jmp     ROVER
 4483
 4484                                    ;*************************************************************************
 4485                                    ;
 4486                                    ;       test the diskette using MSCP commands
 4487                                    ;
 4488                                    ;*************************************************************************
 4489 065762                             tstdrv:
 4490 065762                                     printx  #ASMSGT         ;Output a <cr><lf> for looks
 4491
 4492 066002                                     GMANID  drv.nbr,UNIT,D,0,0,255.,NO
 4493                                                                     ;Ask user to enter the test drive number
 4494
 4495 066022                                     printx  #DSKUT          ;Print message indicating disk under test
 4496 066042  012737  000012  002574              mov     #10.,DELAY      ;load data for 10 second delay
 4497 066050  004737  051522                      jsr     pc,HRDINT       ;In't controller to allow drive to be
 4498                                                                     ;brought ON LINE
 4499 066054  012737  000074  002574              mov     #60.,DELAY      ;load data for 60 second delay
 4500 066062                                     SCC                     ;Set the Controller Characterist'cs
      066062  032737  100000  002534  SCC6:       bit     #bit15,cmdrng+2         ;test ownership of r'ng to make sure
                                                                                  ;we own it
      066070  001374                              bne     SCC6                    ;'f we don't, wait until we do
      066072  012737  000040  002446              mov     #40,cmdlen              ;load length of packet to be sent
      066100  112737  000000  002450              movb    #0,cmdlen+2             ;load message type and credit value
      066106  112737  000000  002451              movb    #MSCP.id,cmdlen+3       ;load MSCP connection ID
      066114  005237  002452                      inc     cmdpak                  ;load new CRN
      066120  005037  002454                      clr     cmdpak+2
      066124  005037  002456                      clr     cmdpak+4
      066130  005037  002460                      clr     cmdpak+6
      066134  012737  000004  002462              mov     #op.scc,cmdpak+10       ;load opcode
      066142  005037  002464                      clr     cmdpak+12               ;load modifiers
      066146  005037  002466                      clr     cmdpak+14               ;NO MODIFIERS
      066152  005037  002470                      clr     cmdpak+16               ;load controller flags
      066156  005037  002472                      clr     cmdpak+20               ;load default MSCP timeout value
      066162  005037  002474                      clr     cmdpak+22
      066166  005037  002476                      clr     cmdpak+24
      066172  005037  002500                      clr     cmdpak+26
      066176  005037  002502                      clr     cmdpak+30
      066202  005037  002504                      clr     cmdpak+32
      066206  005037  002506                      clr     cmdpak+34
      066212  005037  002510                      clr     cmdpak+36

      066216  012777  066260  114102              mov     #RFD6,@vector           ;NEW VECTOR PLACE
      066224  012737  002352  002526              mov     #rsppak,rsprng          ;load response packet area into ring
      066232  012737  002452  002532              mov     #cmdpak,cmdrng          ;load command packet area into ring
      066240  012737  140000  002530              mov     #140000,rsprng+2        ;PORT OWNERSHIP BIT.
      066246  012737  100000  002534              mov     #bit15,cmdrng+2
      066254  004737  047376                      jsr     pc,POLLWT               ;GO TO POLL AND WAIT ROUTINE.
                                        ;*************************************************************************
      066260                                RFD6:                               ;INTR TO HERE.
```

SIZER Supplied Program Data

```
        066260  062706  000006                  add     #6,sp                   ;fix stack for interrupt (4),
                                                                                ;pollwt subrtn (2)
        066264  012777  065360  114034          mov     #intsrv,@vector         ;CHANGE VECTOR
        066272  004737  060352                  jsr     pc,RSPCHK               ;Go to routine that will check on
                                                                                ;the response recvd from the mut.
                                                                                ;it will check the cmd ref
                                                                                ;num, the endcode, and status.
  4501  066276                                  ONLINE                  ;Bring the Unit On Line
        066276  032737  100000  002534  ONL7:   bit     #bit15,cmdrng+2         ;test ownership of r'ng to make sure
                                                                                ;we own it
        066304  001374                          bne     ONL7                    ;'f we don't, wa t until we do
        066306  012737  000044  002446          mov     #44,cmdlen              ;load length of packet to be sent
        066314  112737  000000  002450          movb    #0,cmdlen+2             ;load message type and credit value
        066322  112737  000000  002451          movb    #MSCP.id,cmdlen+3       ;load MSCP connection ID
        066330  005237  002452                  inc     cmdpak                  ;load new CRN
        066334  005037  002454                  clr     cmdpak+2
        066340  013737  002330  002456          mov     UNIT,cmdpak+4           ;unit number
        066346  005037  002460                  clr     cmdpak+6
        066352  012737  000011  002462          mov     #op.onl,cmdpak+10       ;load opcode
        066360  005037  002464                  clr     cmdpak+12               ;load modifiers
        066364  005037  002466                  clr     cmdpak+14               ;reserved
        066370  005037  002470                  clr     cmdpak+16               ;flags
        066374  005037  002472                  clr     cmdpak+20
        066400  005037  002474                  clr     cmdpak+22
        066404  005037  002476                  clr     cmdpak+24
        066410  005037  002500                  clr     cmdpak+26
        066414  005037  002502                  clr     cmdpak+30
        066420  005037  002504                  clr     cmdpak+32
        066424  005037  002506                  clr     cmdpak+34               ;use default tuning parameters
        066430  005037  002510                  clr     cmdpak+36

        066434  012777  066476  113664          mov     #RFD7,@vector           ;NEW VECTOR PLACE
        066442  012737  002352  002526          mov     #rsppak,rsprng          ;load response packet area into ring
        066450  012737  002452  002532          mov     #cmdpak,cmdrng          ;load command packet area into ring
        066456  012737  140000  002530          mov     #140000,rsprng+2        ;PORT OWNERSHIP BIT.
        066464  012737  100000  002534          mov     #bit15,cmdrng+2
        066472  004737  047376                  jsr     pc,POLLWT               ;GO TO POLL AND WAIT ROUTINE.
                        ;*************************************************************************
        066476                          RFD7:                                   ;INTR TO HERE.
        066476  062706  000006                  add     #6,sp                   ;fix stack for interrupt (4),
                                                                                ;pollwt subrtn (2)
        066502  013737  002416  002556          mov     rsppak+44,MAXLLBN       ;save low word of Max Available LBNs
        066510  013737  002420  002560          mov     rsppak+46,MAXHLBN       ;save high word of Max Available LBNs
        066516  162737  000001  002556          sub     #1,maxllbn              ;get max lbn versus size
        066524  005637  002560                  sbc     maxhlbn                 ;
        066530  012777  065360  113570          mov     #intsrv,@vector         ;CHANGE VECTOR
        066536  004737  060352                  jsr     pc,RSPCHK               ;Go to routine that will check on
                                                                                ;the response recvd from the mut.
                                                                                ;it will check the cmd ref
                                                                                ;num, the endcode, and status.
  4502  066542                                  GUS                     ;Get the Unit Status
        066542  032737  100000  002534  GUS10:  bit     #bit15,cmdrng+2         ;test ownership of r'ng to make sure
                                                                                ;we own it
        066550  001374                          bne     GUS10                   ;if we don't, wait until we do
        066552  012737  000014  002446          mov     #14,cmdlen              ;load length of packet to be sent
        066560  112737  000000  002450          movb    #0,cmdlen+2             ;load message type and cred't value
        066566  112737  000000  002451          movb    #MSCP.id,cmdlen+3       ;load MSCP connect'on ID
```

```
066674  005237  002452                  inc     cmdpak                  ;load new CRN
066700  005037  002454                  clr     cmdpak+2
066704  013737  002330  002456          mov     UNIT,cmdpak+4           ;unit number
066712  005037  002460                  clr     cmdpak+6
066716  012737  000003  002462          mov     #op.gus,cmdpak+10       ;load opcode
066724  005037  002464                  clr     cmdpak+12               ;load modifiers
066730  005037  002466                  clr     cmdpak+14               ;NO MODIFIERS

066734  012777  066676  113464          mov     #RFD10,@vector          ;NEW VECTOR PLACE
066742  012737  002352  002526          mov     #rsppak,rsprng          ;load response packet area into ring
066650  012737  002452  002532          mov     #cmdpak,cmdrng          ;load command packet area into ring
066656  012737  140000  002530          mov     #140000,rsprng+2        ;PORT OWNERSHIP BIT.
066664  012737  100000  002534          mov     #bit15,cmdrng+2
066672  004737  047376                  jsr     pc,POLLWT               ;GO TO POLL AND WAIT ROUTINE.
                        ;*****************************************************************
066676                          RFD10:                                  ;INTR TO HERE.
066676  062706  000006                  add     #6,sp                   ;fix stack for interrupt (4),
                                                                        ;pollwt subrtn (2)
066702  012777  065360  113416          mov     #intsrv,@vector         ;CHANGE VECTOR
066710  013737  002416  002564          mov     rsppak+44,trksiz
066716  013737  002564  002562          mov     trksiz,bytsiz           ;Calculate bytes per track
066724  000337  002562                  swab    bytsiz
066730  006337  002562                  asl     bytsiz                  ;BYTSIZ = TRKSIZ * 1000 Octal
066734  004737  060352                  jsr     pc,RSPCHK               ;Go to routine that will check on
                                                                        ;the response recvd from the mut.
                                                                        ;it will check the cmd ref
                                                                        ;num, the endcode, and status.
4503 066740                             CMPR                    ;Compare the data written to the disk
     066740  005001                     clr     r1              ;make sure bits 8-15 are zero in r1 and r2
     066742  005002                     clr     r2
     066744  005037  002552             clr     LOLBN           ;Clear low and high words of LBN counter
     066750  005037  002554             clr     HILBN
     066754  005037  002566             clr     ERRCNT          ;Clear cumulative error counter
     066760  005037  002570             clr     TRKCNT          ;Clear track counter
     066764                     NUTRK11:
     066764  005000                     clr     r0              ;Set offset = 0
     066766  005003                     clr     r3              ;Clear bad byte counter
     066770                             WRITE                   ;Send data from SNDBUF to disk
             000012                     B=B+1                   ;increment the CRN number
     066770                             wrttmp  \B              ;Call variables B, C, and D as if they are numbers (\)
     066770  032737  100000  002534     WRT12:  bit     #bit15,cmdrng+2 ;test ownership of ring to make sure
                                                                        ;we own it
     066776  001374                     bne     WRT12           ;if we don't, wait until we do
     067000  012737  000040  002446     mov     #40,cmdlen      ;load length of packet to be sent
     067006  112737  000000  002450     movb    #0,cmdlen+2     ;load message type and credit value
     067014  112737  000000  002451     movb    #MSCP.id,cmdlen+3       ;load MSCP connection ID
     067022  005237  002452             inc     cmdpak          ;load new CRN
     067026  005037  002454             clr     cmdpak+2
     067032  013737  002330  002456     mov     UNIT,cmdpak+4   ;unit number
     067040  005037  002460             clr     cmdpak+6
     067044  012737  000042  002462     mov     #op.wr,cmdpak+10        ;load opcode
     067052  005037  002464             clr     cmdpak+12       ;load modifiers
     067056  013737  002562  002466     mov     BYTSIZ,cmdpak+14        ;byte count
     067064  005037  002470             clr     cmdpak+16
     067070  012737  003000  002472     mov     #SNDBUF,cmdpak+20       ;address of buffer
     067076  005037  002474             clr     cmdpak+22
     067102  005037  002476             clr     cmdpak+24
```

C9

SIZER Supplied Program Data

```
067106  005037  002500                  clr     cmdpak+26
067112  005037  002502                  clr     cmdpak+30
067116  005037  002504                  clr     cmdpak+32
067122  013737  002552  002506          mov     LOLBN,cmdpak+34         ;low word of lbn
067130  013737  002554  002510          mov     HILBN,cmdpak+36         ;high word of lbn

067136  012777  067200  113162          mov     #RFD12,@vector          ;NEW VECTOR PLACE
067144  012737  002352  002526          mov     #rsppak,rsprng          ;load response packet area into ring
067152  012737  002452  002532          mov     #cmdpak,cmdrng          ;load command packet area into ring
067160  012737  140000  002530          mov     #140000,rsprng+2        ;PORT OWNERSHIP BIT.
067166  012737  100000  002534          mov     #bit15,cmdrng+2
067174  004737  047376                  jsr     pc,POLLWT               ;GO TO POLL AND WAIT ROUTINE.
                        ;*****************************************************************
067200                          RFD12:                                  ;INTR TO HERE.
067200  062706  000006                  add     #6,sp                   ;fix stack for interrupt (4),
                                                                        ;pollwt subrtn (2)
067204  012777  065360  113114          mov     #intsrv,@vector         ;CHANGE VECTOR
067212  004737  060352                  jsr     pc,RSPCHK               ;Go to routire that will check on
                                                                        ;the response recvd from the mut.
                                                                        ;it will check the cmd ref
                                                                        ;num, the endcode, and status.
067216                                  READ                    ;Get data from disk and place it in RCVBUF
        000013                  B=B+1                   ;increment the CRN number
067216                          readtmp \B              ;Call variable B as if it were a number (\)
                                                        ;the low word of lbn, and HILBN carries the high
                                                        ;word of lbn
067216  032737  100000  002534  READ13: bit     #bit15,cmdrng+2         ;test ownership of ring to make sure
                                                                        ;we own it
067224  001374                          bne     READ13                  ;if we don't  wait until we do
067226  012737  000040  002446          mov     #40,cmdlen              ;load lengt  of packet to be sent
067234  112737  000000  002450          movb    #0,cmdlen+2             ;load message type and credit value
067242  112737  000000  002451          movb    #MSCP.id,cmdlen+3       ;load MSCP connection ID
067250  005237  002452                  inc     cmdpak                  ;load new CRN
067254  005037  002454                  clr     cmdpak+2
067260  013737  002330  002456          mov     UNIT,cmdpak+4           ;unit number
067266  005037  002460                  clr     cmdpak+6
067272  012737  000041  002462          mov     #op.RD,cmdpak+10        ;load opcode
067300  005037  002464                  clr     cmdpak+12               ;load modifiers
067304  013737  002562  002466          mov     BYTSIZ,cmdpak+14        ;byte count
067312  005037  002470                  clr     cmdpak+16
067316  012737  030376  002472          mov     #RCVBUF,cmdpak+20       ;address of buffer
067324  005037  002474                  clr     cmdpak+22
067330  005037  002476                  clr     cmdpak+24
067334  005037  002500                  clr     cmdpak+26
067340  005037  002502                  clr     cmdpak+30
067344  005037  002504                  clr     cmdpak+32
067350  013737  002552  002506          mov     LOLBN,cmdpak+34         ;lo word of lbn
067356  013737  002554  002510          mov     HILBN,cmdpak+36         ;hi word of lbn

067364  012777  067426  112734          mov     #RFD13,@vector          ;NEW VECTOR PLACE
067372  012737  002352  002526          mov     #rsppak,rsprng          ;load response packet area into ring
067400  012737  002452  002532          mov     #cmdpak,cmdrng          ;load command packet area into ring
067406  012737  140000  002530          mov     #140000,rsprng+2        ;PORT OWNERSHIP BIT.
067414  012737  100000  002534          mov     #bit15,cmdrng+2
067422  004737  047376                  jsr     pc,POLLWT               ;GO TO POLL AND WAIT ROUTINE.
                        ;*****************************************************************
067426                          RFD13:                                  ;INTR TO HERE.
```

SIZER Supplied Program Data

```
067426  062706  000006                  add     #6,sp                   ;fix stack for interrupt (4),
                                                                        ;pollwt subrtn (2)
067432  012777  065360  112666          mov     #intsrv,@vector         ;CHANGE VECTOR
067440  004737  060352                  jsr     pc,RSPCHK               ;Go to routine that will check on
                                                                        ;the response recvd from the mut.
                                                                        ;it will check the cmd ref
                                                                        ;num, the endcode, and status.

067444  126060  030376  003000  CMP11:  cmpb    RCVBUF(r0),SNDBUF(r0)
                                                                ;Is the data in SNDBUF equal to data in RCVBUF?
067452  001401                          beq     UPDT11          ;If so, skip bad byte counter update
067454  005203                          inc     r3              ;Update bad byte counter
067456  005200                  UPDT11: inc     r0              ;Increment offset
067460  023700  002562                  cmp     BYTSIZ,r0
067464  001367                          bne     CMP11           ;If not at the end of buffers, compare next byte
067466  005703                          tst     r3
067470  001413                          beq     CNTR11          ;Branch over Bad Byte Report if none found

067520  063737  002564  002552  CNTR11: add     TRKSIZ,LOLBN    ;Update track counters
067526  005537  002554                  adc     HILBN           ;Add carry from LOLBN to HILBN

067532  023737  002554  002560  OVER11: cmp     HILBN,MAXHLBN   ;If high word of LBN <> Maximum high word
                                                                ;of LBN, update counters
067540  001011                          bne     JMP11
067542  023737  002552  002556          cmp     LOLBN,MAXLLBN   ;If high word of LBN = maximum high word
                                                                ;of LBN and low word
                                                                ;of LBN <= Maximum low word of LBN,
                                                                ;go to next block
067550  002011                          bge     END11
067552  022703  000000                  cmp     #0,r3           ;Check to see if any bad bytes found
067556  001402                          beq     JMP11           ;If none, go to next track
067560  005237  002566                  inc     ERRCNT          ;Otherwise, update error count
067564  005237  002570          JMP11:  inc     TRKCNT          ;Update track counter
067570  000137  066764                  jmp     NUTRK11         ;Go to next track
4504                                                            ;with the data in memory
4505 067574                             printx  #DONE           ;Print message indicating that all LBNs
4506                                                            ;on the disk
4507                                                            ;have been tested and bad status reported
4508 067614                             printb  #BTFND,ERRCNT   ;Print message indicating number of bad
4509                                                            ;blocks found
4510
4511 067640                             GMANIL  do.agn,ENDIT,0,NO
4512                                                            ;Ask user if wants to test another floppy
4513
4514 067654  005737  002572                  tst     ENDIT           ;Was response no??
4515 067660  001002                          bne     1$
4516 067662  000137  074424                  jmp     dropunt         ;If so, drop unit and end pass
4517 067666  000137  065762          1$:     jmp     tstdrv          ;If response was yes, jump to tstdrv
4518
4519 067672  122737  000023  002340  ROVER:  cmpb    #Mrqdx3,mdlnbr          ;Doesn't want to test floppy
4520                                                                    ;check if RQDX3 controller
4521                                                                    ;and continue formatting
4522 067700  001403                          beq     2$
4523 067702  042737  100000  002336          bic     #bit15,untflgs          ;if other ther RQDX3 than impossible
4524                                                                    ;to run auto sizer or in auto mode
4525 067710  032737  100000  002336  2$:     bit     #bit15,untflgs          ;test if auto mode is enabled
4526 067716  001412                          beq     1$                      ;if not skip the auto sizer routine
```

SIZER Supplied Program Data

```
4527
4528 067720  012700  177777            mov     #-1,r0
4529 067724  000240            11$:    nop                     ;waste just a little time
4530 067726  077002                    sob     r0,11$
4531
4532 067730  005037  002606            clr     recv.done       ;say is the first time for check on pollwt
4533 067734  004737  054334            jsr     pc,AUTOsizer    ;if it's then run AUTO SIZER on the
4534                                                            ;controller
4535 067740  004737  056172            jsr     pc,AUTOd's      ;display information from autosizer
4536                                                            ;routine
4537 067744                    1$:                             ; ...
4538 067744  012737  000001  002606    mov     #1,recv.done    ;
4539 067752  005077  112346            clr     @IPreg          ;can any spurious interrupts
4540 067756                            printx  #ASMSGT         ; ...
4541 067776                    ELPcmd:
4542 067776  000401                    br      4$              ; set this to a NOP for APT compatability
4543 070000  000415                    br      3$              ; skip manual question
4544 070002  005037  002322    4$:     clr     boot            ; WARNING - remove boot diskette first
4545 070006                    GMANIL  bot.dev,BOOT,-1,YES     ; Insert new diskette
4546                                                            ; DO you want to continue
4547 070022  005737  002322            tst     BOOT            ;
4548 070026  001002                    bne     3$              ; Yes, run format
4549 070030  000137  074424            jmp     dropunt         ; No, drop unit
4550 070034                    3$:
4551
4552 070034  012737  000012  002574    mov     #10.,delay      ;load data for 10 second delay
4553 070042  004737  051522            jsr     pc,hrdint       ; Reinit ctrl in case of unknown state
4554 070046  012737  000074  002574    mov     #60.,delay      ;load data for 60 second delay
4555 070054                        printb      #pb9,mdlnbr     ; Print the disk controller model number
4556 070100                        printb      #pb10,mcdnbr    ; Print microcode version number in dec.
4557
4558 070124  032737  100000  002336        bit     #bit15,untflgs  ;test if auto mode is enabled
4559 070132  001011                        bne     1$              ;branch if in auto mode else
4560 070134                    GMANID ASK.prg,PRGnam,A,-1,6.,6.,yes    ;ask for the User what local program
4561                                                                   ;he wants to run
4562 070154  000411                        br      2$
4563 070156                    1$:
4564 070156  012737  047506  002734        mov     #"FO,PRGnam             ;place "FORMAT" into asci buffer if
4565                                                                       ;in auto mode
4566 070164  012737  046522  002736        mov     #"RM,PRGnam+2
4567 070172  012737  052101  002740        mov     #"AT,PRGnam+4
4568 070200                    2$:
4569 070200  023727  002342  000002        cmp     mcdnbr,#2       ;check microcode rev number
4570 070206  001402                        beq     NHDW2           ;If rev = 2 continue execution
4571 070210  000137  070462                jmp     NHDW1
4572 070214                    NHDW2:
4573
4574 070214                    excSUPprg       #DUPfmt,#<DUPend-DUPfmt>;downline load the program DUPFMT
     070214  032737  100000  002534    ESP14:  bit     #bit15,cmdrng+2 ;test ownership of ring make sure we
                                                                       ;own it
     070222  001374                        bne     ESP14           ;if we don't own it wait until we do
     070224  012737  000050  002446        mov     #50,cmdlen      ;load length of packet to be sent
     070232  112737  000000  002450        movb    #0,cmdlen+2     ;load msg type and credit value
     070240  112737  000002  002451        movb    #dup.id,cmdlen+3    ;load DUP connection ID
     070246  005237  002452                inc     cmdpak
     070252  005037  002454                clr     CMDpak+2
     070256  005037  002456                clr     CMDpak+4
```

F9

SIZER Supplied Program Data

```
070262  005037  002460                 clr     CMDpak+6
070266  012737  000002  002462         mov     #op.esp,CMDpak+10       ;load up opcode
070274  012737  000000  002464         mov     #0,CMDpak+12            ;no stand alone modifier
070302  012737  001402  002466         mov     #<DUPend-DUPfmt>,cmdpak+14      ;load length of prg into buffer
070310  005037  002470                 clr     cmdpak+16
070314  012737  052732  002472         mov     #DUPfmt,cmdpak+20       ;starting address of downline load prg
070322  005037  002474                 clr     CMDpak+22
070326  005037  002476                 clr     CMDpak+24
070332  005037  002500                 clr     CMDpak+26
070336  005037  002502                 clr     CMDpak+30
070342  005037  002504                 clr     CMDpak+32
070346  005037  002506                 clr     CMDpak+34               ;overlay buffer descriptor
070352  005037  002510                 clr     CMDpak+36
070356  005037  002512                 clr     CMDpak+40
070362  005037  002514                 clr     CMDpak+42
070366  005037  002516                 clr     CMDpak+44
070372  005037  002520                 clr     CMDpak+46
070376  012777  070440  111722         mov     #RFD14,@vector          ;New vector place
070404  012737  002352  002526         mov     #rsppak,rsprng          ;load response packet area into ring
070412  012737  002452  002532         mov     #cmdpak,cmdrng          ;load command packet area into ring
070420  012737  140000  002530         mov     #140000,RSPRNG+2        ;Port ownership bit.
070426  012737  100000  002534         mov     #bit15,CMDRNG+2
070434  004737  047376                 jsr     pc,POLLWT               ;Go to poll and wait routine.
                        ;*****************************************************************************
070440                          RFD14:                                  ;Intr to here.
070440  062706  000006                 add     #6,sp                   ;fix stack for interrupt (4), pollwt
                                                                        ;subrtn (2)
070444  012777  065360  111654         mov     #intsrv,@vector         ;Change vector
070452  004737  060352                 jsr     pc,RSPChK               ;Go to routine that will check on
                                                                        ;the response recvd from the mut.
4575
4576 070456  000137  070674                 jmp     RCDcmd
4577
4578 070462                      NHDW1:
4579
4580 070462                      EXLCPRG PRGnam                          ;Execute Local program "FORMAT" or
070462  032737  100000  002534  ELP15:  bit     #bit15,cmdrng+2         ;test ownership of ring make sure we
                                                                        ;own it
070470  001374                         bne     ELP15                   ;if we don't own it wait until we do
070472  012737  000022  002446         mov     #22,cmdlen              ;load length of packet to be sent
070500  112737  000000  002450         movb    #0,cmdlen+2             ;load msg type and credit
070506  112737  000002  002451         movb    #dup.id,cmdlen+3        ;load DUP connection ID
070514  005237  002452                 inc     cmdpak                  ;load new CRN
070520  005037  002454                 clr     cmdpak+2
070524  005037  002456                 clr     cmdpak+4
070530  005037  002460                 clr     cmdpak+6
070534  012737  000003  00246c         mov     #op.elp,cmdpak+10       ;load up opcode
070542  012737  000001  002464         mov     #stdaln,cmdpak+12       ;stand alone modifier
070550  012700  000006                 mov     #6,r0                   ;6 letters transfer
070554  012701  002466                 mov     #cmdpak+14,r1           ;starting address to place program name
070560  012702  002734                 mov     #PRGnam,r2              ;start of Program Name
070564  112221                 rfdj15: movb    (r2)+,(r1)+             ;add 2 to bycnt then store
070566  077002                         sob     r0,rfdj15

070570  012777  070632  111530         mov     #RFD15,@vector          ;New vector place
070576  012737  002352  002526         mov     #rsppak,rsprng          ;load response packet area into ring
070604  012737  002452  002532         mov     #cmdpak,cmdrng          ;load command packet area into ring
```

SIZER Supplied Program Data

```
070612  012737  140000  002530          mov     #140000,RSPRNG+2        ;Port ownership bit.
070620  012737  100000  002534          mov     #bit15,CMDRNG+2
070626  004737  047376                  jsr     pc,POLLWT               ;Go to poll and wait routine.

                                ;*************************************************************************

070632                              RFD15:                              ;Intr to here.
070632  062706  000006                  add     #6,sp                   ;fix stack for interrupt (4), pollwt
                                                                        ;subrtn (2)
070636  012777  065360  111462          mov     #intsrv,@vector         ;Change vector
070644  004737  060352                  jsr     pc,RSPCHK               ;Go to routine that will check on
                                                                        ;the response recvd from the mut.
                                                                        ;it will check the cmd ref
                                                                        ;num, the endcode and status.
                                                                        ;whatever they wrote
4581
4582
4583 070650  122737  000011  002371          cmpb    #bit3+bit0,rsppak+17    ;is this program a standalone, DUP
4584                                                                    ;dialog type
4585 070656  001406                          beq     1$                      ;
4586 070660                                  ERRDF   2,DF3                   ;"Dev'ce Fatal can't do remote programs'
4587 070670  000137  074424                  jmp     dropunt                 ;drop unit and go on
4588
4589 070674                          1$:
4590 070674                          RCDcmd:
4591 070674                          RECVDAT #datare,#80.
     070674  032737  100000  002534 RCD16:  b't     #bit15,cmdrrg+2         ;test ownership of r'ng make sure we
                                                                        ;own 't
     070702  001374                          bne     RCD16                   ;if we don't own it wa't until we do
     070704  012737  000034  002446          mov     #34,cmdlen              ;load length of packet to be sent
     070712  112737  000000  002450          movb    #0,cmdlen+2             ;load msg type and cred't
     070720  112737  000002  002451          movb    #dup.id,cmdlen+3        ;load DUP connection ID
     070726  005237  002452                  'nc     cmdpak                  ;load new CRN
     070732  005037  002454                  clr     cmdpak+2
     070736  005037  002456                  clr     cmdpak+4
     070742  005037  002460                  clr     cmdpak+6
     070746  012737  000005  002462          mov     #op.rec,cmdpak+10       ;load up opcode
     070754  005037  002464                  clr     cmdpak+12               ;no mod'f'ers
     070760  012737  000120  002466          mov     #80.,cmdpak+14
     070766  005037  002470                  clr     cmdpak+16
     070772  012737  002610  002472          mov     #datare,cmdpak+20       ;load address of buffer descriptor
     071000  005037  002474                  clr     cmdpak+22
     071004  005037  002476                  clr     cmdpak+24
     071010  005037  002500                  clr     cmdpak+26
     071014  005037  002502                  clr     cmdpak+30
     071020  005037  002504                  clr     cmdpak+32

     071024  012777  071066  111274          mov     #RFD16,@vector          ;New vector place
     071032  012737  002352  002526          mov     #rsppak,rsprng          ;load response packet area into ring
     071040  012737  002452  002532          mov     #cmdpak,cmdrng          ;load command packet area into ring
     071046  012737  140000  002530          mov     #140000,RSPRNG+2        ;Port ownership bit.
     071054  012737  100000  002534          mov     #bit15,CMDRNG+2
     071062  004737  047376                  jsr     pc,POLLWT               ;Go to poll and wait rout'ne.

                                ;*************************************************************************

     071066                              RFD16:                              ;Intr to here.
     071066  062706  000006                  add     #6,sp                   ;fix stack for interrupt (4), pollwt
```

```
                                                                                ;subrtn (2)
071072  012777  065360  111226          mov     #intsrv,@vector               ;Change vector
071100  004737  060352                  jsr     pc,RSPCHK                     ;Go to routine that will check on
                                                                              ;the response recvd from the mut.
                                                                              ;it will check the cmd ref
                                                                              ;num, the endcode and status.

                                ;+
                                ;       get
                                ;       r3 = type
                                ;       r4 = SA adrs
                                ;       r5 = sub number
                                ;-
071104  113703  002611          DUPDLG: movb    datare+1,r3                   ;get dup type info
071110  006203                          asr     r3
071112  006203                          asr     r3
071114  006203                          asr     r3
071116  006203                          asr     r3
071120  042703  177760                  bic     #type,r3                      ;mask off all but DUP type
071124  013705  002610                  mov     datare,r5                     ;get dup message number info
071130  042705  170000                  bic     #msgnbr,r5                    ;clear out top 4 bits


                                ;+
                                ; Check for the type.
                                ; if QUESTION type, it will be answered by sending
                                ; an answer through a Send command which will be followed
                                ; by a Receive command to await further instructions.
                                ;
                                ; If a DEFAULT QUESTION type is given an answer will
                                ; either be given or a blank send command returned.
                                ; Either way we will do a Send command followed by a
                                ; Receive command.
                                ;
                                ; if INFORMATIONAL type, check message number and type
                                ; informmation according to message number given.
                                ;
                                ; if FATAL ERROR type, check message number and print
                                ; error message accordingly. No other commands will
                                ; be given following this type of command.
                                ;
                                ; If TERMINATION type check the message number and print the
                                ; correct message. usually this implies a succesful
                                ; end to the formatter. After this command we exit the program
                                ;
                                ; If SPECIAL type we are asking for the FCT table to be passed
                                ; to the RQDX3 controller. We will send the table with a Send
                                ; command and then to a Receive command to proceed.
                                ;
071134  022703  000001          qstn:   cmp     #Question,r3                  ;test for "question" subtype
071140  001117                          bne     dfqstr                        ;if not branch
071142  032737  020000  002336          bit     #bit13,untflgs                ;see if we are working on a known
                                                                              ;controller
071150  001077                          bne     qnbra                         ;if not type out ascii
071152  122737  000106  002734          cmpb    #'F,prgnam                    ;if running the format program then
                                                                              ;print info
071160  001073                          bne     qnbra                         ;else just go for an answer
```

SIZER Supplied Program Data

```
4643 071162  004737  060334          qnbr0:  jsr     pc,clrDUPbuf            ;clear out data buffer so DRS macros
4644                                                                        ;don t show default
4645 071166  022705  000000                  cmp     #0,r5                   ;check for message number
4646 071172  001036                          bne     qnbr7                   ;check for next message number
4647 071174  032737  100000  002336          bit     #bit15,untflgs
4648 071202  001011                          bne     1$
4649 071204                          GMANID  qfdat,DATARE,A,177777,10.,10.,no                ;DATE MM-DD-YYYY ?
4650 071224  000417                          br      2$
4651 071226  012737  033060  002610  1$:     mov     #"06,data e             ;The date is  ot used anyway so any
4652                                                                        ;date will do
4653 071234  012737  030455  002612          mov     #"-1,datare+2           ;T ll be celebra ing th s day
4654 071242  012737  026467  002614          mov     #"7-,datare+4
4655 071250  012737  034461  002616          mov     #"19,datare+6
4656 071256  012737  030460  002620          mov     #"86,datare+10
4657 071264  000137  072022          2$:     jmp     SDTcmd                  ;branch to Send Data command
4658
4659 071270  022705  000007          qnbr7:  cmp     #7,r5                   ;check for message number
4660 071274  001025                          bne     qnbra                   ;check for next message number
4661 071276  032737  100000  002336          bit     #bit15,untflgs
4662 071304  001011                          bne     1$
4663 071306                          GMANID  qfser,DATARE,A,177777,8.,10.,NO         ;SERIAL NUMBER 9 dig ts ?
4664 071326  000406                          br      2$
4665 071330  013700  002334          1$:     mov     sernbr,r0
4666 071334  012701  002610                  mov     #datare,r1              ;place to stick asc i
4667 071340  004737  060044                  jsr     pc,OCTASC               ;convert octal to dec mal asci
4668 071344  000137  072022          2$:     jmp     SDTcmd
4669
4670 071350  004737  060240          qnbra:  jsr     pc,typDUPbuf            ;type out ASCII sent by d sk controller
4671 071354                          GMANID  ASK.ANSWER,DATARE,A,177777,0.,10.,YES   ;g ve t an answer
4672 071374  000137  072022                  jmp     SDTcmd                  ;branch to Send Data commard
4673
4674
4675 071400  022703  000002          dfqstn: cmp     #DefQuest,r3            ;test for "Default Quest on" subtype
4676 071404  001402                          beq     1$
4677 071406  000137  072236                  jmp     nfrm                    ;f not branch
4678 071412  032737  020000  002336  1$:     bit     #bit13,untflgs          ;see f we are work ng on a known
4679                                                                        ;controller
4680 071420  001402                          beq     2$
4681 071422  000137  071776                  jmp     dqnbra                  ;f not type out asci
4682 071426  122737  000106  002734  2$:     cmpb    #'F,prgnam              ;f runn ng the format program then
4683                                                                        ;pr nt nfo
4684 071434  001160                          bne     dqnbra                  ;else just go for an answer
4685
4686 071436  004737  060334          dqnbr1: jsr     pc,clrDUPbuf            ;clear out data buffer so DRS macros
4687                                                                        ;don t show default
4688 071442  022705  000001                  cmp     #1,r5                   ;check for message numbe
4689 071446  001043                          bne     dqnbr4                  ;check for next message r mber
4690                                                                        ;put n message number
4691 071450  032737  100000  002336          bit     #bit15,untflgs
4692 071456  001011                          bne     3$
4693 071460                          GMANID  dfunt,DATARE,A,177777,0,3,YES   ;Ask for UNIT NUMBER 0-255 ?
4694 071500  000406                          br      4$
4695 071502  013700  002330          3$:     mov     unit,r0                 ;get unit number if in auto mode from
4696                                                                        ;Hardware P table
4697 071506  012701  002610                  mov     #datare,r1              ;store dec mal asc  conversion n
4698                                                                        ;data area
4699 071512  004     060044                  jsr     pc,OCTASC               ;convert octal to asc  dec mal n
```

SIZER Supplied Program Data

```
                                                                        ;data area
                                                                        ;address of ascii decimal data
                                                                        ;address to store octal conversion
                                                                        ;convert ascii decimal to octal
                                                                        ;make sure unit number is less than 4
                                                                        ;or between 0-3
                                                                        ;subtract 4 until unit is less than four
                                                                        ;branch to Send Data command
                                                                        ;check for message number
                                                                        ;check for next message number
                                                                        ;set the default for NO
                                                                        ;Use existing bad block information
                                                                        ;(Y or N)?
                                                                        ;branch to Send Data command
                                                                        ;check for message number
                                                                        ;check for next message number
                                                                        ;Set the default for YES
                                                                        ;Use Down Line Load (Y or N)?
                                                                        ;branch to Send Data command
                                                                        ;check for message number
                                                                        ;check for next message number
                                                                        ;set the default for NO
                                                                        ;is this auto mode
                                                                        ;NO, ask question
                                                                        ;Yes see if RD51
                                                                        ; first cylinder entry
                                                                        ;
                                                                        ; point to current unit entry
                                                                        ; Is it an RD51?
                                                                        ; NO, all done
                                                                        ; YES, make question answer yes because
                                                                        ; NO FCT tables on RD51
                                                                        ; set the default for NO
                                                                        ; and skip question
                                                                        ;Continue if bad block information is
                                                                        ;naccessable (Y or N)?
                                                                        ;if unknown use default and continue
                                                                        ;who knows, maybe it will be useful
                                                                        ;some day
                                                                        ;type out ASCII sent by disk controller
```

SIZEP Supplied Program Data

```
 4757 072002                                GMANID  ASK.ANSWER,DATARE,A,177777,0..10..YES          ;give it an answer
 4758
 4759 072022                                SDTcmd:
 4760 072022                                        SENDDAT #datare,#10.                    ;sent the answer
      072022  032737  100000  002534  SDT17:  bit     #bit15,cmdrng+2                 ;test ownership of ring make sure we
                                                                                        ;own it
      072030  001374                          bne     SDT17                           ;if we don t own it wait until we do
      072032  012737  000034  002446          mov     #34,cmdlen                      ;load length of packet to be sent
      072040  112737  000000  002450          movb    #0,cmdlen+2                     ;load msg type and credit
      072046  112737  000002  002451          movb    #dup.id,cmdlen+3                ;load DUP connection ID
      072054  005237  002452                  inc     cmdpak                          ;load new CRN
      072060  005037  002454                  clr     cmdpak+2
      072064  005037  002456                  clr     cmdpak+4
      072070  005037  002460                  clr     cmdpak+6
      072074  012737  000004  002462          mov     #op.sen,cmdpak+10               ;load up opcode
      072102  005037  002464                  clr     cmdpak+12                       ;no modifiers
      072106  012737  000012  002466          mov     #10.,cmdpak+14
      072114  005037  002470                  clr     cmdpak+16
      072120  012737  002610  002472          mov     #datare,cmdpak+20               ;load address of buffer descriptor
      072126  005037  002474                  clr     cmdpak+22
      072132  005037  002476                  clr     cmdpak+24
      072136  005037  002500                  clr     cmdpak+26
      072142  005037  002502                  clr     cmdpak+30
      072146  005037  002504                  clr     cmdpak+32

      072152  012777  072214  110146          mov     #RFD17,@vector                  ;New vector place
      072160  012737  002352  002526          mov     #rsppak,rsprng                  ;load response packet area into ring
      072166  012737  002452  002532          mov     #cmdpak,cmdrng                  ;load command packet area into ring
      072174  012737  140000  002530          mov     #140000,RSPRNG+2                ;Port ownership bit.
      072202  012737  100000  002534          mov     #bit15,CMDRNG+2
      072210  004737  047376                  jsr     pc,POLLWT                       ;Go to poll and wait routine.

                              ;*****************************************************************************

      072214                          RFD17:                                          ;Intr to here.
      072214  062706  000006                  add     #6,sp                           ;fix stack for interrupt (4), pollwt
                                                                                      ;subrtn (2)
      072220  012777  065360  110100          mov     #intsrv,@vector                 ;Change vector
      072226  004737  060352                  jsr     pc,RSPCHK                       ;Go to routine that will check on
                                                                                      ;the response recvd from the mut.
                                                                                      ; t will check the cmd ref
                                                                                      ;num, the endcode and status.
 4761 072232  000137  070674                  jmp     RCDcmd                          ;do another receive cmd
 4762
 4763
 4764
 4765 072236  022703  000003          infrm:  cmp     #Inform,r3                      ;test for "Informational" subtype
 4766 072242  001046                          bne     term                            ;if not branch
 4767 072244  032737  020000  002336          bit     #bit13,untflgs                  ;see if we are working on a known
 4768                                                                                  ;controller
 4769 072252  001036                          bne     'nbra                           ;if not type out asc
 4770 072254  122737  000106  002734          cmpb    #'F,prgnam                      ;if running the format program then
 4771                                                                                  ;print info
 4772 072262  001032                          bne     'nbra
 4773
 4774 072264  022705  000000          'nbr0:  cmp     #0,r5                           ;check for message number
 4775 072270  001012                          bne     'nbr1                           ;check for next message number
```

SIZER Supplied Program Data

```
4776 072272  004737  060334            jsr     pc,clrDUPbuf        ;clear out DUP buffer so there is no
4777                                                                 ;echo on last ASCII
4778 072276                            printf  #sfbegt             ;format begin
4779 072316  022705  000001    1$nbr1:  cmp     #1,r5               ;check for message number
4780 072322  001012                    bne     1$nbra              ;check for next message number
4781 072324  004737  060334            jsr     pc,clrDUPbuf        ;clear out DUP buffer so there is no
4782                                                                 ;echo on last ASCII
4783 072330                            printf  #sfdont             ;format complete
4784
4785 072350  004737  060240    1$nbra:  jsr     pc,typDUPbuf        ;type out ASCII sent by disk controller
4786 072354  000137  070674            jmp     RCDcmd              ;do another receive command
4787
4788
4789
4790 072360  022703  000004    term:    cmp     #terminat,r3        ;test for termination type
4791 072364  001116                    bne     ftler               ;if not branch
4792 072366  032737  020000  002336    bit     #bit13,untflgs      ;see if we are working on a known
4793                                                                 ;controller
4794 072374  001076                    bne     tnbra               ;if not type out ascii
4795 072376  122737  000106  002734    cmpb    #'F,prgnam          ;if running the format program then
4796                                                                 ;branch to error routine
4797 072404  001072                    bne     tnbra
4798
4799 072406  022705  000014    tnbr12:  cmp     #12.,r5             ;test for sub number #1
4800 072412  001012                    bne     tnbr13              ;branch if not sub number #1
4801 072414                            printf  #sffcut
4802 072434  000137  074424            jmp     dropunt             ;drop test unit and end pass
4803
4804 072440  022705  000015    tnbr13:  cmp     #13.,r5             ;test for msg number
4805 072444  001052                    bne     tnbra               ;branch if not right number
4806 072446                            printf  #sffcnt             ;
4807 072466  032737  100000  002336    bit     #bit15,untflgs      ;are we in auto mode
4808 072474  001434                    beq     2$                  ;  NO , then we are all done
4809                                                                 ;  YES, is this an RX33
4810 072476  013701  002330            mov     unit,r1             ; first cylinder entry
4811 072502  006301                    asl     r1                  ;
4812 072504  062701  056162            add     #msg+4,r1           ; point to current unit entry
4813 072510  022711  000003            cmp     #3,(r1)             ; Is it an RX33?
4814 072514  001024                    bne     2$                  ; NO, all done
4815                                                                 ; YES, as if it wants to continue or not
4816
4817 072516  005077  107602            clr     @IPreg              ; reinit the controller stop spurious
4818                                                                 ;interrupts
4819 072522                    GMANIL  bot.con,BOOT, 1,YES         ; Do you want to format another?
4820
4821 072536  005737  002322            tst     BOOT                ; Yes, execute local program
4822 072542  001007                    bne     1$                  ; No, tell him to insert bootable media
4823
4824 072544                    GMANIL  bot.rep,BOOT,-1,YES         ; Please insert boot media and hit return
4825 072560  000402                    br      2$                  ;
4826 072562  000137  067776    1$:     jmp     ELPcmd              ;
4827 072566  000137  074424    2$:     jmp     dropunt             ;
4828
4829 072572  004737  060240    tnbra:   jsr     pc,typDUPbuf        ;type out ASCII sent by disk controller
4830 072576                            printf  #PF2                ;print finished local program without
4831                                                                 ;procedure error
4832 072616  000137  074432            jmp     etst                ;end DUP diaglog but stay in test loop
```

M9


```
4833
4834
4835 072622  022703  000005          ftler:  cmp     #Ftlerr,r3          ;test for "Fatal Error' subtype
4836 072626  001402                          beq     1$
4837 072630  000137  074104                  jmp     spcl                ;if not branch
4838 072634  032737  020000  002336  1$:     bit     #bit13,untflgs      ;see if we are working on a known
4839                                                                      ;controller
4840 072642  001004                          bne     3$                  ;if not type out ascii
4841 072644  122737  000106  002734          cmpb    #'F,prgnam          ;if running the format program then
4842                                                                      ;branch to error routine
4843 072652  001414                          beq     2$
4844 072654  004737  060240          3$:     jsr     pc,typDUPbuf        ;type out ASCII sent by disk controller
4845 072660                                  printf  #DF15               ;Fatal error reported when running
4846                                                                      ;local program
4847 072700  000137  074424                  jmp     dropunt             ;drop unit and end pass
4848
4849 072704                          2$:     ERRHRD  1,HRDO              ;Hard device  error
4850
4851 072714  022705  000001          fnbr1:  cmp     #1,r5               ;test for sub number #1
4852 072720  001012                          bne     fnbr2               ;branch if not sub number #1
4853 072722                          gstsf:
4854 072722                                  printb  #efstat             ;"GET STATUS failure"
4855 072742  000137  074424                  jmp     dropunt             ;drop unit and end pass
4856
4857 072746  022705  000002          frbr2:  cmp     #2.,r5              ;test for msg number
4858 072752  001012                          bne     fnbr3               ;branch if not right number
4859 072754                                  printf  #efsndt             ;
4860 072774  000137  074424                  jmp     dropunt             ;drop unit and end pass
4861
4862 073000  022705  000003          fnbr3:  cmp     #3.,r5              ;test for msg number
4863 073004  001012                          bne     fnbr4               ;branch if not right number
4864 073006                                  printf  #efcmdt             ;
4865 073026  000137  074424                  jmp     dropunt             ;drop unit and end pass
4866
4867 073032  022705  000004          fnbr4:  cmp     #4.,r5              ;test for msg number
4868 073036  001012                          bne     fnbr5               ;branch if not right number
4869 073040                                  printf  #efrcvt             ;
4870 073060  000137  074424                  jmp     dropunt             ;drop unit and end pass
4871
4872 073064  022705  000005          fnbr5:  cmp     #5.,r5              ;test for msg number
4873 073070  001012                          bne     fnbr6               ;branch if not right number
4874 073072                                  printf  #efbust             ;
4875 073112  000137  074424                  jmp     dropunt             ;drop unit and end pass
4876
4877 073116  022705  000006          fnbr6:  cmp     #6.,r5              ;test for msg number
4878 073122  001012                          bne     fnbr7               ;branch if not right number
4879 073124                                  printf  #efinit             ;
4880 073144  000137  074424                  jmp     dropunt             ;drop unit and end pass
4881
4882 073150  022705  000007          fnbr7:  cmp     #7.,r5              ;test for msg number
4883 073154  001012                          bne     fnbr8               ;branch if not right number
4884 073156                                  printf  #efnut              ;
4885 073176  000137  074424                  jmp     dropunt             ;drop unit and end pass
4886
4887 073202  022705  000010          fnbr8:  cmp     #8.,r5              ;test for msg number
4888 073206  001012                          bne     fnbr9               ;branch if not right number
4889 073210                                  printf  #efdxft
```

N9

SIZER Supplied Program Data

```
 4890 073230  000137  074424                jmp     dropunt          ;drop unit and end pass
 4891
 4892 073234  022705  000011        fnbr9:  cmp     #9.,r5           ;test for msg number
 4893 073240  001012                bne     fnbr10           ;branch if not right number
 4894 073242                        printf  #effcct          ;
 4895 073262  000137  074424        jmp     dropunt          ;drop unit and end pass
 4896
 4897 073266  022705  000012        fnbr10: cmp     #10.,r5          ;test for msg number
 4898 073272  001012                bne     fnbr11           ;branch if not right number
 4899 073274                        printf  #efsekt          ;
 4900 073314  000137  074424        jmp     dropunt          ;drop unit and end pass
 4901
 4902 073320  022705  000013        fnbr11: cmp     #11.,r5          ;test for msg number
 4903 073324  001012                bne     fnbr12           ;branch if not right number
 4904 073326                        printf  #efrcct          ;
 4905 073346  000137  074424        jmp     dropunt          ;drop unit and end pass
 4906
 4907 073352  022705  000014        fnbr12: cmp     #12.,r5          ;test for msg number
 4908 073356  001012                bne     fnbr13           ;branch if not right number
 4909 073360                        printf  #eflbft          ;
 4910 073400  000137  074424        jmp     dropunt          ;drop unit and end pass
 4911
 4912 073404  022705  000015        fnbr13: cmp     #13.,r5          ;test for msg number
 4913 073410  001012                bne     fnbr14           ;branch if not right number
 4914 073412                        printf  #effcwt          ;
 4915 073432  000137  074424        jmp     dropunt          ;drop unit and end pass
 4916
 4917 073436  022705  000016        fnbr14: cmp     #14.,r5          ;test for msg number
 4918 073442  001012                bne     fnbr15           ;branch if not right number
 4919 073444                        printf  #efrcrt          ;
 4920 073464  000137  074424        jmp     dropunt          ;drop unit and end pass
 4921
 4922 073470  022705  000017        fnbr15: cmp     #15.,r5          ;test for msg number
 4923 073474  001012                bne     fnbr16           ;branch if not right number
 4924 073476                        printf  #efrcwt          ;
 4925 073516  000137  074424        jmp     dropunt          ;drop unit and end pass
 4926
 4927 073522  022705  000020        fnbr16: cmp     #16.,r5          ;test for msg number
 4928 073526  001012                bne     fnbr17           ;branch if not right number
 4929 073530                        printf  #efrcft          ;
 4930 073550  000137  074424        jmp     dropunt          ;drop unit and end pass
 4931
 4932 073554  022705  000021        fnbr17: cmp     #17.,r5          ;test for msg number
 4933 073560  001012                bne     fnbr18           ;branch if not right number
 4934 073562                        printf  #effcrt          ;
 4935 073602  000137  074424        jmp     dropunt          ;drop unit and end pass
 4936
 4937 073606  022705  000022        fnbr18: cmp     #18.,r5          ;test for msg number
 4938 073612  001012                bne     fnbr19           ;branch if not right number
 4939 073614                        printf  #effcnt          ;
 4940 073634  000137  074424        jmp     dropunt          ;drop unit and end pass
 4941
 4942 073640  022705  000023        fnbr19: cmp     #19.,r5          ;test for msg number
 4943 073644  001012                bne     fnbr20           ;branch if not right number
 4944 073646                        printf  #effcdt          ;
 4945 073666  000137  074424        jmp     dropunt          ;drop unit and end pass
 4946
```

```
4947 073672  022705  000024          fnbr20: cmp     #20.,r5         ;test for msg number
4948 073676  001012                          bne     fnbr21          ;branch if not right number
4949 073700                                  printf  #eftmot         ;
4950 073720  000137  074424                  jmp     dropunt         ;drop unit and end pass
4951
4952 073724  022705  000025          fnbr21: cmp     #21.,r5         ;test for msg number
4953 073730  001012                          bne     fnbr22          ;branch if not right number
4954 073732                                  printf  #efillt         ;
4955 073752  000137  074424                  jmp     dropunt         ;drop unit and end pass
4956
4957 073756  022705  000026          fnbr22: cmp     #22.,r5         ;test for msg number
4958 073762  001012                          bne     fnbr23          ;branch if not right number
4959 073764                                  printf  #efwart         ;
4960 074004  000137  074424                  jmp     dropunt         ;drop unit and end pass
4961
4962 074010  022705  000027          fnbr23: cmp     #23.,r5         ;test for msg number
4963 074014  000412                          br      fnbr24          ;branch if not right number
4964 074016                                  printf  #efinpt         ;
4965 074036  000137  074424                  jmp     dropunt         ;drop unit and end pass
4966
4967
4968 074042  022705  000030          fnbr24: cmp     #24.,r5         ;test for msg number
4969 074046  001012                          bne     1$
4970 074050                                  printf  #efmedt
4971 074070  000137  074424                  jmp     dropunt         ;drop unit and end pass
4972
4973 074074  004737  060240          1$:     jsr     pc,typDUPbuf    ;type out ASCII sent by disk controller
4974 074100  000137  074424                  jmp     dropunt         ;drop unit and end pass
4975
4976
4977
4978
4979 074104  022703  000006          spcl:   cmp     #specl,r3       ;test for special type
4980 074110  001137                          bne     unkwn           ;branch if not known
4981 074112  032737  020000  002336          bit     #bit13,untflgs  ;see if we are working on a known
4982                                                                 ;controller
4983 074120  001004                          bne     2$              ;if not type out ascii
4984 074122  122737  000106  002734          cmpb    #'F,prgnam      ;if running the format program then
4985                                                                 ;print info
4986 074130  001414                          beq     1$
4987 074132  004737  060240          2$:     jsr     pc,typDUPbuf    ;type out ASCII sent by disk controller
4988 074136                                  printf  #DF16           ;special command issued by local
4989                                                                 ;program did not know how to handle
4990 074156  000137  074410                  jmp     unkwn           ;report error
4991
4992 074162  022705  000002          1$:     cmp     #2,r5           ;test for message number 1
4993 074166  001110                          bne     unkwn           ;branch if not known
4994 074170  004737  057240                  jsr     pc,bldu it      ;go get or build UIT table
4995 074174                                  SENDDAT UITadr,#UITsiz  ;sent Unit Information table
     074174  032737  100000  002534  SDT20:  bit     #bit15,cmdrng+2 ;test ownership of ring make sure we
                                                                     ;own it
     074202  001374                          bne     SDT20           ;if we don't own it wait until we do
     074204  012737  000034  002446          mov     #34,cmdlen      ;load length of packet to be sent
     074212  112737  000000  002450          movb    #0,cmdlen+2     ;load msg type and credit
     074220  112737  000002  002451          movb    #dup.id,cmdlen+3 ;load DUP connection ID
     074226  005237  002452                  inc     cmdpak          ;load new CRN
     074232  005037  002454                  clr     cmdpak+2
```

SIZER Supplied Program Data

```
074236  005037  002456                 clr     cmdpak+4
074242  005037  002460                 clr     cmdpak+6
074246  012737  000004  002462         mov     #op.sen,cmdpak+10       ;load up opcode
074254  005037  002464                 clr     cmdpak+12               ;no modifers
074260  012737  000104  002466         mov     #UITsiz,cmdpak+14
074266  005037  002470                 clr     cmdpak+16
074272  013737  002320  002472         mov     UITadr,cmdpak+20        ;load address of buffer descriptor
074300  005037  002474                 clr     cmdpak+22
074304  005037  002476                 clr     cmdpak+24
074310  005037  002500                 clr     cmdpak+26
074314  005037  002502                 clr     cmdpak+30
074320  005037  002504                 clr     cmdpak+32

074324  012777  074366  105774         mov     #RFD20,@vector          ;New vector place
074332  012737  002352  002526         mov     #rsppak,rsprng          ;load response packet area into ring
074340  012737  002452  002532         mov     #cmdpak,cmdrng          ;load command packet area into ring
074346  012737  140000  002530         mov     #140000,RSPRNG+2        ;Port ownership bit.
074354  012737  100000  002534         mov     #bit15,CMDRNG+2
074362  004737  047376                 jsr     pc,POLLWT               ;Go to poll and wait routine.

                        ;***************************************************************

074366                          RFD20:                                  ;Intr to here.
074366  062706  000006                 add     #6,sp                   ;fix stack for interrupt (4), pollwt
                                                                        ;subrtn (2)
074372  012777  065360  105726         mov     #intsrv,@vector         ;Change vector
074400  004737  060352                 jsr     pc,RSPCHK               ;Go to routine that will check on
                                                                        ;the response recvd from the mut.
                                                                        ;it will check the cmd ref
                                                                        ;num, the endcode and status.
4996 074404  000137  070674                 jmp     RCDcmd             ;do another receive cmd
4997
4998
4999 074410                     unkwn:
5000 074410                             ERRSF   0,SFO                   ; system error unknown response
5001 074420  004737  060446             jsr     pc,PRNTpkt              ;type out packet information
5002
5003 074424                     dropurt:
5004 074424                             DODU    LOGUNIT                 ;drop the unit
5005
5006 074432                     etst:
5007 074432                             doclr                           ;take controller offline
5008 074434                     ENDTST
```

SIZER Suppl'ed Program Data

```
5010 074436                             BGNHRD
5011
5012 074440                                 GPRMA 'p.adr,0,0,160000,177776,YES   ;Get IP reg addr (170000-177776)
5013                                                                             ;place in word 2 of the table
5014                                                                             ;default value 's from default
5015                                                                             ;table.
5016
5017 074450                                 GPRMA vec.adr,2,0,0,776,YES          ;Get the vector addr (octal 0 776)
5018                                                                             ;place in word
5019                                                                             ;default value is from default
5020                                                                             ;table.
5021
5022 074460                                 GPRML tst.dsk,10,bit12,YES           :ask if they want to test floppy
5023
5024 074466                                 XFERT label0                         ;If last gpprml input is true (y)
5025                                                                             ;transfer control to label.
5026
5027 074470                                 GPRML auto.md,10,bit15,YES           ;ask if they want to go into auto mode
5028                                                                             ;This will format the drive us'ng
5029                                                                             ;tne autos'zer
5030
5031 074476                                 XFERF label0                         ;If last gprml input is false (n)
5032                                                                             ;transfer control to label.
5033
5034 074500                                 GPRMD drv.nbr,4,D,-1,0,255.,YES      ;Get the logical drive (DECIMAL 0-255)
5035                                                                             ;place in word. Default value is from
5036                                                                             ;default table.
5037
5038 074512                                 GPRMD ser.nbr,6,D,-1,1,012345.,YES   ;Get the drive serial number
5039                                                                             ;place in word. Default value 's from
5040                                                                             ;default table.
5041
5042
5043 074524                             label0:
5044
5045 074524                             ex't hrd
5046 074526                              ENDHRD
5047
5048
5049 074526                              LASTAD
     074532                             L$LAST::
5050 074532                             ENDMOD
5051            000001                    .END
```

E10

Symbol table

```
A      = 000000          BIT9   = 001000 G        C$INLP= 000020          DOFSEK  053270 G        EFSTAT  022107
ABORT    065632          BLOUIT   057240          C$MANI= 000050          DONE    030211           EFTMOT  023211
ABRT3    052456          BOE    = 000400 G        C$MAP = 000102          DOPASS= 066210           EFUNRG  023457
ADR    = 000020 G        BOOT     002322          C$MEM = 000031          DOSPOT= 073450           EFWART  023312
ASCDEC   060132          BOT.CO   010421          C$MMU = 000103          DOUDC   056064 G         EF.CON= 000036 G
ASK.AN   021272          BOT.DE   007726          C$MSG = 000023          DOURET  056146           EF.NEW= 000035 G
ASK.DB   006717          BOT.RE   010331          C$ONR= 000034           DO.AGN  004302           EF.PWR= 000034 G
ASK.LB   006772          BTFND    030054          C$OPNW= 000104          DO.CON  004447           EF.RES= 000037 G
ASK.PR   006576          BTRPT    030121          C$PNTB= 000014          DQNBRA  071776           EF.STA= 000040 G
ASK.RB   007045          BYTSIZ   002562          C$PNTF= 000017          DQNBR1  071436           ELPCMD  067776
ASK.XB   006644          CINTR    002522          C$PNTS= 000016          DQNBR4  071556           ELP15   070462
ASMSGR   005661          CLRBUF= 074476           C$PNTX= 000015          DQNBR5  071626           END     065634
ASMSGT   006531          CLRDUP   060334          C$PUTB= 000072          DQNBR6  071676           ENDIT   002572
ASMSG1   005755          CMDLEN   002446          C$PUTW= 000073          DROPUN  074424           END11   067574
ASMSG2   006256          CMDPAK   002452          C$QIO  = 000377         DRPUNT  020520           ERRCNT  002566
ASMSG3   006301          CMDRNG   002532          C$RDBU= 000007          DRVTXA  004503           ERSEKO= 000003
ASMSG4   006363          CMP11    067444          C$REFG= 000047          DRVTXB  004532           ERUDON= 000001
ASMSG5   006433          CNTR11   067520          C$REL  = 000077         DRVTXC  005565           ERUINT= 000002
ASMSG6   006505          CONT     060042          C$RESE= 000033          DRVTXO  004626           ESP14   070214
ASMSG7   006043          CONTON   065530          C$RET  = 077472         DRVTX1  004722           ESP4    054334
ASMSG8   006110          CRET$  = 077472          C$REVI= 000003          DRVTX2  005016           ETST    074432
ASMSG9   006174          CSV$   = 077456          C$RFLA= 000021          DRVTX3  005112           EVL    = 000004 G
ASSEMB= 000010           C$AU   = 000052          C$RPT  = 000025         DRVTX4  005206           E$END  = 002100
AUTO     055034 G        C$AUTO= 000061           C$SAV  = 077456         DRVTX5  005302           E$LOAD= 000035
AUTOBL   057556          C$BRK  = 000022          C$SEFG= 000046          DRVTX6  005376           FCPR    007426
AUTODI   056172          C$BSEG= 000004           C$SPRI= 000041          DRVTX7  005472           FCPW    007360
AUTOEN   056172          C$BSUB= 000002           C$SVEC= 000037          DRV.NB  004176           FIBOFF= 001004
AUTOSI   054334          C$CLCK= 000062           C$TOME= 000076          DSKUT   030335           FIBSIZ= 000013
AUTOSZ   055010          C$CLEA= 000012           DATA   = 110600         DTBL    051450           FILL.I= 027222
AUTO.M   004326          C$CLOS= 000035           DATARE  002610          DTMP    051444           FINDA1= 074002
B      = 000020          C$CLP1= 000006           DATOFF= 001032          DUPDLG  071104           FINDID  054066
BIT0   = 000001 G        C$CPBF= 000074           DATSIZ= 001000          DUPEND  054334 G         FMTTRK  007214
BIT00  = 000001 G        C$CPME= 000075           DBN     002756          DUPFMT  052732           FMTUNT= 064520
BIT01  = 000002 G        C$CVEC= 000036           DECASC  051220          DUPRES  053056           FNBR1   072714
BIT02  = 000004 G        C$DCLN= 000044           DECTBL  060116          DUPSTA  052756 G         FNBR10  073266
BIT03  = 000010 G        C$DODU= 000051           DEFQUE= 000002          DUP.ID= 000002           FNBR11  073320
BIT04  = 000020 G        C$DRPT= 000024           DELAY   002574          EFBUST  022246           FNBR12  073352
BIT05  = 000040 G        C$DU   = 000053          DESELE= 043172          EFCMDT  022164           FNBR13  073404
BIT06  = 000100 G        C$EDIT= 000003           DFBAD   021027          EFDXFT  022371           FNBR14  073436
BIT07  = 000200 G        C$ERDF= 000055           DFCON   021127          EFFCCT  022460           FNBR15  073470
BIT08  = 000400 G        C$ERHR= 000056           DFDWN   021077          EFFCDT  023154           FNBR16  073522
BIT09  = 001000 G        C$ERRO= 000060           DFPTBL  002276 G        EFFCNT  023130           FNBR17  073554
BIT1   = 000002 G        C$ERSF= 000054           DFQSTN  071400          EFFCRT  023105           FNBR18  073606
BIT10  = 002000 G        C$ERSO= 000057           DFUNT   020766          EFFCWT  022740           FNBR19  073640
BIT11  = 004000 G        C$ESCA= 000010           DF1     011670          EFILLT  023240           FNBR2   072746
BIT12  = 010000 G        C$ESEG= 000005           DF11    012154          EFINIT  022272           FNBR20  073672
BIT13  = 020000 G        C$ESUB= 000003           DF12    012211          EFINPT  023413           FNBR21  073724
BIT14  = 040000 G        C$ETST= 000001           DF13    012245          EFLBFT  022655           FNBR22  073756
BIT15  = 100000 G        C$EXIT= 000032           DF14    012321          EFMEDT  023434           FNBR23  074010
BIT15T   064316          C$FREQ= 000101           DF15    012402          EFNUT   022335           FNBR24  074042
BIT2   = 000004 G        C$FRME= 000100           DF16    012472          EFRCCT  022566           FNBR3   073000
BIT3   = 000010 G        C$GETB= 000026           DF2     011732          EFRCFT  023070           FNBR4   073032
BIT4   = 000020 G        C$GETW= 000027           DF3     012001          EFRCRT  023021           FNBR5   073064
BIT5   = 000040 G        C$GMAN= 000043           DF4     012111          EFRCVT  022215           FNBR6   073116
BIT6   = 000100 G        C$GPHR= 000042           DIAGMC= 000000          EFRCWT  023044           FNBR7   073150
BIT7   = 000200 G        C$GPRI= 000040           DNINT   052624          EFSEKT  022547           FNBR8   073202
BIT8   = 000400 G        C$INIT= 000011           DOFCMD  053236 G        EFSNDT  022136           FNBR9   073234
```

F10

Symbol table

```
FPRPT   050300        G$RADA= 000140        LKS    = 177546 G     L$SPTP  002024 G       ME811   025417
FRPTB   007120        G$RADB= 000000        LKSVCT= 000100 G      L$STA   002030 G       ME812   025451
FTLER   072622        G$RADD= 000040        LKS.SE  052700 G      L$TEST  002114 G       ME813   025503
FTLERR= 000005        G$RADL= 000120        LKVEC  = 000100        L$TIML  002014 G       ME814   025534
F$AU   = 000015       G$RADO= 000020        LOCAL   002312        L$UNIT  002012 G       ME815   025567
F$AUTO= 000020        G$XFER= 000004        LOE    = 040000 G     L10000  002310         ME82    025132
F$BGN  = 000040       G$YES  = 000010       LOGUNI  002310        L10002  065634         ME83    025155
F$CLEA= 000007        HERZ    002604        LOLBN   002552        L10003  065644         ME84    025203
F$DU   = 000016       HEXA1  = 000241       LOPRGI  002546        L10004  065662         ME87    025246
F$END  = 000041       HEXFE  = 000376       LOT    = 000010 G     L10005  065716         ME88    025302
F$HARD= 000004        HEXF8  = 000370       LSTCMD  002542        L10006  074434         ME89    025333
F$HW   = 000013       HILBN   002554        LSTCRN  002540        L10007  074526         ME90    025622
F$INIT= 000006        HIPRGI  002550        LSTVCT  002544        MANBLD  057254         ME91    025657
F$JMP  = 000050       HOE    = 100000 G     L$ACP   002110 G      MAXDRV= 000004         ME92    025710
F$MOD  = 000000       HRDINT  051522        L$APT   002036 G      MAXHLB  002560         ME93    025733
F$MSG  = 000011       HRDO    012777        L$AUT   002070 G      MAXLLB  002556         ME94    025771
F$PROT= 000021        IBE    = 010000 G     L$AUTO  065636 G      MCDNBR  002342         ME95    026026
F$PWR  = 000017       IDU    = 000040 G     L$CCP   002106 G      MDLNBR  002340         MOD1    002000 G
F$RPL  = 067552       ID.TAB= 100066        L$CLEA  065646 G      MEA1    026063         MRQDX1= 000007
F$RPT  = 000012       IER    = 020000 G     L$CO    002032 G      MEA2    026130         MRQDX3= 000023
F$SEG  = 000003       INBRA   072350        L$DEPO  002011 G      MEA3    026146         MSCPEN  027524
F$SOFT= 000005        INBRO   072264        L$DESC  002126 G      MEA4    026222         MSCPGU  027603
F$SRV  = 000010       INBR1   072316        L$DESP  002076 G      MEA5    026251         MSCPON  027712
F$SUB  - 000002       INFORM= 000003        L$DEVP  002060 G      MEA6    026325         MSCPOP  030001
F$SW   = 000014       INFRM   072236        L$DISP  002124 G      MEA7    026402         MSCPRD  027736
F$TEST= 000001        INTSRV  065360        L$DLY   002116 G      MEA8    026466         MSCPSC  027637
F.BADB= 000004        IPREG   002324        L$DTP   002040 G      MEA9    026531         MSCPST  027463
F.BADS= 000002        IP.ADR  004144        L$DTYP  002034 G      MEB1    026565         MSCPWR  027757
F.CONT= 000010        ISR    = 000100 G     L$DU    065664 G      MEB10   027242         MSCP.I= 000000
F.CURC= 000000        IXE    = 004000 G     L$DUT   002072 G      MEB11   027311         MSECA  = 007570
F.MAN  = 000012       I$AU   = 000041       L$DVTY  002160 G      MEB12   027357         MSEND   056126
F.MODE= 000006        I$AUTO= 000041        L$EF    002052 G      MEB13   027426         MSG     056156 G
GDSCMD  052272        I$CLK  = 100006       L$ENVI  002044 G      MEB2    026626         MSGDAT  056160
GDS0    047600        I$CLN  = 000041       L$ETP   002102 G      MEB3    026701         MSGLEN= 000014
GDS2    052272        I$DU   = 000041       L$EXP1  002046 G      MEB4    026726         MSGNBR= 170000
GETBLO= 074626        I$HRD  = 000041       L$EXP4  002064 G      MEB5    027014         MSIN    056106
GETINP= 071222        I$INIT= 000041        L$EXP5  002066 G      MEB6    027052         MSWAIT  056102
GETMAN  053536        I$MOD  = 000041       L$HARD  074440 G      MEB7    027117         NEXT    065440
GETMCK  053702        I$MSG  = 000041       L$HIME  002120 G      MEB8    027156         NHDW1   070462
GETMLP  053610        I$PROT= 000040        L$HPCP  002016 G      ME10    023507         NHDW2   070214
GETMNX  053666        I$PTAB= 000041        L$HPTP  002022 G      ME20    023533         NO     = 000000
GETSEC= 071142        I$PWR  = 000041       L$HW    002276 G      ME30    023557         NUTRK1  066764
GOBIT   052256        I$RPT  = 000041       L$ICP   002104 G      ME31    023651         NXTTIM  002576
GSTSF   072722        I$SEC  = 100016       L$INIT  065404 G      ME32    023731         OCTASC  060044
GUS10   066542        I$SEG  = 000041       L$LADP  002026 G      ME34    023767         ONL7    066276
G$CNTO= 000200        I$SETU= 000041        L$LAST  074532 G      ME38    024041         OP.ABR= 000006
G$DELM= 000372        I$SRV  = 000041       L$LOAD  002100 G      ME40    024141         OP.ELP= 000003
G$DISP= 000003        I$SUB  = 000041       L$LUN   002074 G      ME55    024175         OP.END= 000200
G$EXCP= 000400        I$TST  = 000041       L$MREV  002050 G      ME56    024263         OP.ESP= 000002
G$HILI= 000002        I$UDC  = 100002       L$NAME  002000 G      ME57    024347         OP.GDS= 000001
G$LOLI= 000001        I.CYL  = 000036       L$PRIO  002042 G      ME58    024461         OP.GUS= 000003
G$NO   = 000000       I.SPOT= 000106        L$PROT  065376 G      ME59    024557         OP.ONL= 000011
G$OFFS= 000400        I.SUR  = 000034       L$PRT   002112 G      ME6128  024701         OP.RD  = 000041
G$OFSI= 000376        JMP11   067564        L$REPP  002062 G      ME6256  024754         OP.REC= 000005
G$PRMA= 000001        J$JMP  = 000167       L$REV   002010 G      ME70    025027         OP.RT  = 000133
G$PRMD= 000002        LABEL0  074524        L$SPC   002056 G      ME80    025051         OP.SCC= 000004
G$PRML= 000000        LBN     002771        L$SPCP  002020 G      ME810   025364         OP.SEN= 000004
```

G10

Symbol table

```
OP.SRP= 000100          PB1209  017120          QUESTI= 000001          SAINT   052730 G        SVCTST= 177777
OP.WR = 000042          PB1210  017221          QUO.RE= 034164          SCC6    066062          S$LSYM= 010000
OVER11  067532          PB1211  017263          RBN     003004          SCPR    007543          S$$BUG  056152
O$APTS= 000000          PB1212  017317          RCDCMD  070674          SCPW    007474          S$$FLA  056154
O$AU  = 000000          PB1213  017374          RCD16   070674          SDTCMD  072022          S$$RTI  056062 G
O$BGNR= 000000          PB1214  017440          RCD5    054576          SDT17   072022          S$$UDC  056022 G
O$BGNS= 000000          PB1215  017511          RCVBUF  030376          SDT20   074174          S$$UDI  056040
O$DU  = 000001          PB1216  017552          RDDFCT  007330          SD.FLA= 100232          TBLBLD  060040
O$ERRT= 000000          PB1217  017646          RDMAN   053706 G        SELECT= 042652          TBQ0    010473
O$GNSW= 000000          PB1218  017743          RDMEX   054326          SERNBR  002334          TBQ1    010560
O$PCIN= 000001          PB1219  020020          RDMOK   054324          SER.NB  004224          TBQ10   011022
O$SETU= 000001          PB1220  020057          RDMTRY  053722          SETUP   065432          TBQ11   011045
PBF0    014151          PB1221  020144          RDNXT   054164          SFBEGT  021277          TBQ12   011074
PBF1    014251          PB1222  020213          RDOFCM= 070540          SFCYLT  021763          TBQ13   011133
PBF10   015204          PB1223  020306          RDSPOT  054224          SFDBBT  021545          TBQ14   011146
PBF2    014400          PB13    014061          RD.MOD= 000300          SFDONT  021320          TBQ15   011165
PBF3    014454          PB3     013245          READ13  067216          SFFCNT  022036          TBQ16   011176
PBF4    014550          PB4     013313          RECONS= 000002          SFFCUT  022004          TBQ17   011223
PBF5    014613          PB5     013365          RECV.D  002606          SFRBBT  021750          TBQ18   011242
PBF6    014660          PB6     013456          REFCNT= 076421          SFRCBT  021465          TBQ19   011261
PBF7    014755          PB7     013560          REFCRT= 076410          SFREVT  021344          TBQ2    010602
PBF8    015054          PB8     013612          REFSEK= 076320          SFR1T   021366          TBQ20   011314
PBF9    015144          PB9     013646          REG.7 = 100206          SFR2T   021420          TBQ21   011344
PBSF0   020452          PF2     014064          RESDRV= 074370          SFTRYT  021705          TBQ22   011376
PB0     013134          PKTS  = 100050          RESTOR= 037506          SFT0    013022          TBQ23   011411
PB1     013163          PLOC    002314          RETRY = 000367          SFT1    013073          TBQ24   011424
PB10    013710          PNT   = 001000 G        RFDJ15  070564          SFXBBT  021625          TBQ25   011437
PB11    013752          POLLW   047376          RFD0    047746          SF0     012614          TBQ26   011452
PB11AP  015667          POLLWT  047376          RFD10   066676          SF1     012663          TBQ28   011464
PB11CR  015244          PRGNAM  002734          RFD12   067200          SF100   012724          TBQ29   011514
PB11EL  015566          PRI   = 002000 G        RFD13   067426          SIZDRV  055704          TBQ3    010624
PB11EN  015422          PRISUM= 066040          RFD14   070440          SIZEND  055720          TBQ30   011545
PB11ES  015531          PRI00 = 000000 G        RFD15   070632          SIZEXI  055766 G        TBQ31   011573
PB11GD  015501          PRI01 = 000040 G        RFD16   071066          SIZFLP  055334          TBQ32   011635
PB11OP  015314          PRI02 = 000100 G        RFD17   072214          SIZFPS  055324          TBQ4    010646
PB11RD  015642          PRI03 = 000140 G        RFD2    052420          SIZIN   055636          TBQ5    010670
PB11SD  015620          PRI04 = 000200 G        RFD20   074366          SIZLOP  055242 G        TBQ6    010712
PB11ST  015366          PRI05 = 000240 G        RFD3    052604          SIZNON  055230          TBQ7    010734
PB11S0  015711          PRI06 = 000300 G        RFD4    054560          SIZRD   055700          TBQ8    010756
PB11S1  015736          PRI07 = 000340 G        RFD5    054770          SIZRX   055474          TBQ9    011000
PB11S2  015770          PRNTPK  060446          RFD6    066260          SIZRX3  055570          TCBS  = 100052
PB11S3  016026          PROCBA= 074026          RFD7    066476          SIZSET  055152          TCPR    007660
PB11S4  016063          PROGRE= 034014          RFORMA= 064152          SIZWIN  055576          TCPW    007612
PB11S5  016117          PS0   = 000000          RGETMA= 072616          SIZWT   055132          TERM    072360
PB11S6  016146          PS7   = 000340          RINTR   002524          SLEEP   052626 G        TERMIN= 000004
PB11S9  016173          PTBL    002316          ROVER   067672          SNDBUF  003000          TIME    051162
PB11W0  016236          PURL0   051344          RPDFCT  007253          SPCL    074104          TIMEOU  002602
PB11W1  016322          PUTMSG= 074726          RSFFCU= 076232          SPECL = 000006          TIMER   002600
PB12    020423          PUT.UD= 013500          RSPCHK  060352          STDALN= 000001          TIMOUT  052156
PB1201  016413          P$WORK= 000006          RSPPAK  002352          STEPMO  055470          TMPBUF  051426
PB1202  016477          QFDAT   020735          RSPRNG  002526          STEPOU  055454          TNBRA   072572
PB1203  016564          QFSER   021216          RSP1    002346          STEPRX  055376          TNBR12  072406
PB1204  016635          QFUIT   020660          RTRKSI= 024260          SUMSIZ= 000005          TNBR13  072440
PB1205  016676          QNBRA   071350          RW$PLL= 140002          SVCGBL= 000000          TRKCNT  002570
PB1206  016737          QNBR0   071162          R$CMD = 140012          SVCINS= 177777          TRKOFF= 002032
PB1207  017011          QNBR7   071270          R$DAT = 140010          SVCSUB= 177777          TRKSIZ  002564
PB1208  017064          QSTN    071134          R$FPS = 140006          SVCTAG= 177777          TRP100  051172
```

H10

Symbol table

```
TSTDRV  065762          T$PTHV= ****** GX       T1 .    065720 G        UNTDSZ= 000002          W$FPL  = 140004
TST.DS  004261          T$PTNU= 000000          UAM    = 000200 G       UNTFLG  002336          XBN     002743
TYPASC  020561          T$SAVL= 177777          UDC.FL= 100234          UNT.NB  006534          X$ALWA= 000000
TYPDUP  060240          T$SEGL= 177777          UIBOFF= 000000          UPDT11  067456          X$FALS= 000040
TYPE   = 177760         T$SIZE= ****** GX       UIBSIZ= 001004          U$MODE= 000070          X$OFFS= 000400
T$ARGC= 000001          T$SUBN= 000000          UIN     002344          U$OP.S= 000072          X$TRUE= 000020
T$BUFF= 000044          T$SURF= 000032          UITADR  002320          U.DD   = 000001         YES    = 000001
T$CODE= 001004          T$TAGL= 177777          UITDF   004040          U.RD   = 000003         $DEQ.H= 044730
T$CYLI= 000030          T$TAGN= 010010          UITLOC  060010          U.RES  = 000000         $ENQ.H= 044640
T$ERRN= 000000          T$TEMP= 000000          UITOTH= 000010          U.SI1  = 000005         $2      065570
T$EXCP= 000000          T$TEST= 000001          UITSIZ= 000104          U.SO1  = 000007         $3      065610
T$FLAG= 000041          T$TSTM= 177777          UIT0    003000          U.SRD  = 000044         $4      065624
T$FREE= ****** GX       T$TSTS= 000C 1          UIT1    003104          U.SRP  = 000100         .A.DEF= 000040
T$GMAN= 000000          T$UCB  = 000002         UIT2    003210          U.SRX  = 000054         .A.FAT= 000120
T$HILI= 030071          T$$AUT= 010003          UIT3    003314          VECTOR  002326          .A.INF= 000060
T$LAST= 000001          T$$CLE= 010004          UIT4    003420          VEC.AD  004157          .A.QUE= 000020
T$LOLI= 000001          T$$DU  = 010005         UIT5    003524          WARNIN  004347          .A.TER= 000100
T$LSYM= 010000          T$$HAR= 010007          UIT6    003630          WRNGST  052216          .A.TYP= 000020
T$LTNO= 000001          T$$HW  = 010000         UIT7    003734          WRT12   066770          .B.SPL= 000140
T$NEST= 177777          T$$INI= 010002          UNIT    002330          W$CMD  = 140022         .PCB   = 102656
T$NSO  = 000000         T$$PRO= 010001          UNKWN   074410          W$DAT  = 140020         .PKT   = 105646
T$NS1  = 000004         T$$TES= 010006

. ABS.  074532    000   (RW,I,GBL,ABS,OVR)
        000000    001   (RW,I,LCL,REL,CON)
Errors detected:  0

*** Assembler stat'st'cs


Work  f'le  reads: 527
Work  f'le wr'tes: 503
Size of work file: 45192 Words  ( 177 Pages)
S'ze of core pool: 19684 Words  ( 75 Pages)
Operat'ng  system: RSX 11M/PLUS (Under VAX/VMS)

Elapsed t'me: 00:14:14.07
ZRQCE0,ZRQCE0.LST/-SP=SVC35R/ML,ZRQCE0.MAC
```