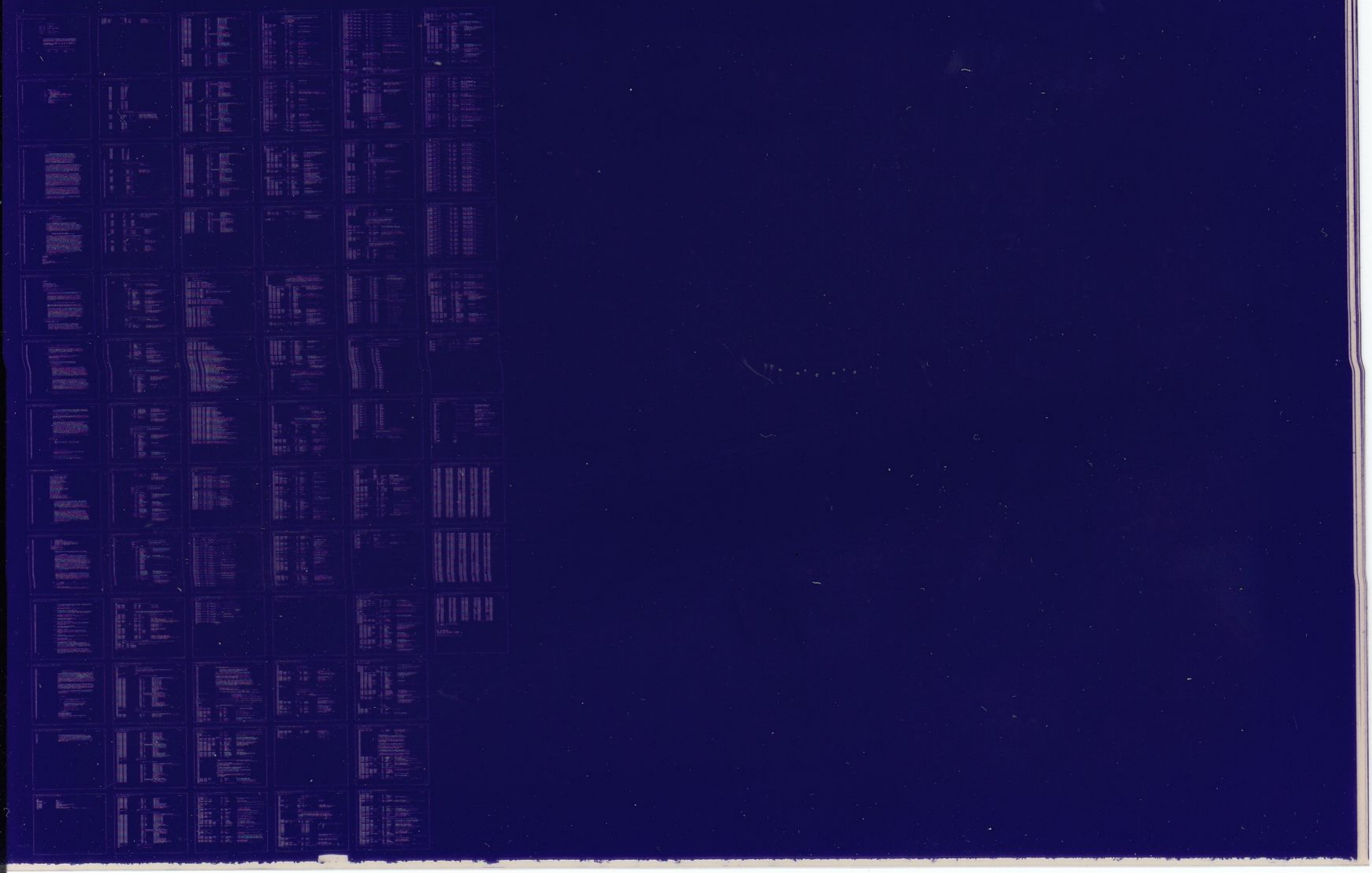
RQDX3 RD51/52
RD53

RQDX3 FORMATTER
CZRQCB0

AH-U110B-MC
1 OF 1 OCT 1985

digital
MADE IN USA



```
.REM %(

                        IDENTIFICATION
                        - --   - ---

PRODUCT CODE:   AC-U109B MC

PRODUCT NAME:   CZRQCB0  RQDX3 FORMATTER

PRODUCT DATE:   JUL 15,  1985

MAINTAINER:     DIAGNOSTIC ENGINEERING

AUTHOR:         Richard Dietz
```

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 differnt 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 similiar 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. If the AUTOSZ program did not recoginize the drive, twenty or so questions will be asked so that a UIT table can be built. This table will enable quick support of future drives such as the RD31 and RD54. 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.

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. If a new or unlisted DEC drive is to be formatted, the UIT table can be built by answering about twenty questions. These twenty questions require a very good understanding of the drive parameters and is made as a engineering tool for formatting new DEC drives. Caution do not use non DEC drives you are liable to destroy 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 drives configured into a Q bus PDP-11 system.

2.2 SOFTWARE REQUIREMENTS

This diagnostic was written using DRS the Diagnostic Supervisor. The diagnostic is expected to be run under XXDP diagnostic operating system. When the auto sizer routine is used it is possible to run the formatter under APT. If manual intervention 's neccessary or the auto sizer is not used the program will not be APT compatable although it will be APT loadable. When in manual mode the diagnostic uses a lot of manual questions answering DUP format questions send by the RQDX3 firmware. For this reason the diagnostic is APT loadable but not APT controllable unless the autosizer is used, in which case no manual questions are asked.

2.3 QUESTIONS ASKED AND THEIR ANSWERS

2.3.1 HARDWARE QUESTIONS FROM DIAGNOSTIC SOFTWARE

The diagnostic is a standard DRS program with the standard DRS commands. Below I have a script of the questions asked an the answers to the initial DRS questions. The Default value for the IP address is 172150. This is 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 Auto Mode has a default of yes. This mode will run an auto sizer to determine the proper drive characteristic table to give to the controller. This auto sizer will figure out how many cylinders on the drive and through a small look up table we decide which table to down-line load to the RQDX3 controller. If Auto mode is used no manual questions will have to be answered. All the questions will be asked in the Hardware Questions. If Auto mode is not used all the questions are asked manually and the characteristics table must be choosen manual. Assuming We picked auto mode the user would have to enter a drive number and a serial number. After this a warning message will appear asking if the user wants to proceed. The default is no so the/ user must type "Y" in order to format his drives.

Typical Diagnostic Script:

```
boot up XXDP
.RUN ZRQC??
ZRQCB0.BIN

DRSXM-A0
ZRQC-B-0
RQDX3 Disk Format Utility
Unit is RD51,RD52,RD53,or RQDX3 Proto-type Winchester drive
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>
Auto Format Mode Y ? <rtn>

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

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

Assuming the user asnwered yes to the auto mode question this is the all the questions he will have to answer unless the drive is unidentifiable in which case the diagnostic will go into manual mode.

If the user wants to be really lazy he can answer no to answering hardware questions in which case all winchesters will be formatted and if there is any floppies on the system it will error. Since the winchesters are always before the floppies you are gauranteed to format all the winnies before getting an error because of trying to format a floppy.

2.3.2 MANUAL QUESTIONS FROM CONTROLLER FIRMWARE

If the user answered no to auto mode then he must answer all the questions by hand. The defaults are suggested but the user must know which Unit Information Table the want to use or the DEC drive name.

Manual Questions are asked from inside the diagnostic and are not part of the P table as described in the DRS programmers guide. The first question and the UIT table questions are asked by the host program all other questions are asked by the RQDX3's firmware. For purposes of international support these questions given by the controller are not used but a message number return along with the question is used to look up the translated question contained in this diagnostic. If the message number is unknown the ASCII data is printed out as is in English. To turn off controller reported messages just set the IXE flag in the diagnostic monitor. Below is a script of the manual questions asked. Depending on how certain questions are answered will depend on what questions will be asked.

```
Text printed, Questions asked ,and  replies:

MSCP Controllre model # : 019
    Microcode version # : 001
```

Every MSCP controller has a model number. The RQDX3s model number is 19. The RQDX1 model number is 7. This also reports the microcode revision number. This model number is used to determine weather or not to run the AUTOSZ program. If the controller is an RQDX3 the AUTOSZ program will be run to determine the drive. If the drive is not recognized a question will ask if you wish to preceed. If you are not famaliar with the disk geometry

```
                        of the drive I suggest you default out and call Field Serv'ce.

What local program do you want to run (A) FORMAT ?

        This question asks what controller local program you want to run.
        Usually if not always we will want to run FORMAT. If you get curious
        you can write DIRECT which is a controller local program which list
        all the controller local programs. The default is to run the local
        program FORMAT. At the prompt just hit "return".

Enter date <MM-DD-YYYY>: (A) ? current date

        There is no default to the date question. You must use the
        appropriate form to answer the date. If not the question will
        be asked again until it is in the correct form.
        EXAMPLE  12-12-1985

Enter unit number to format <0>: (A) ?

        The default unit number is unit or physical drive
        zero. If the drive you want to format is other then
        drive 0 then make sure you type the number followed
        by a carriage return.

Use existing Bad Block Information <N> ? N

        The default is "no" which is probable the best choice for RQDX3
        controllers. For an RQDX1/2 the best choice would be "yes". If this
        question is answered "yes" the down line load question is skipped.
        For new RQDX3 drives you must answer the down line load question and
        therfore should answer "no" to this question. Existing bad block
        information is written by the drive manufacturer on a special part of
        the disk. It is perferable to revector the bad blocks listed by the
        manufacturer. Even if we answer "no" on a RQDX3 to this question as
        long as we answer "yes" to the Down Line Load question the Manufactures
        bad block information will still be used. If the UIT table already
        exists on the drive it would be OK to answer "yes".

Use Down Line Load <Y> ? Y

        If this is a drive straight from the manufactures or taken from
        an old RQDX1/2 system then you want to answer "yes" to this question.
        If this is a reformat of a drive that was already formatted on a
        RQDX3 system before then a "no" maybe answered to this question all
        thow this is not suggested. The only way to get to this question is to
        answer "no" to the use bad block information. If this question is also
        answered "no" the bad block information will not be used. The disk will
        do 3 write read passes to try and find the bad blocks. Doing 3 passes
        of reads and writes will only find about 1/4 of the bad blocks listed by
        the manufactures bad block table and take several minutes longer.
        Therfore I suggest always answering 'yes" to this question if format'ng
        on a RQDX3 controller. If this is an RQDX1/2 always answer "no".


Continue if Bad Block Information is inaccessible <N>? Y

        I always answer "yes". If the bad block informat'on can not be found
```

you still want to format your drive. For this reason I always pick "yes". In most cases the manufacturing tables should be there unless you have a Proto-type drive. If you are interested in knowing weather the bad block information is on the drive answer "no".

```
Enter serial number <6 digits> ? 012345
```

This question has no default. A serial number should be picked for the drive that is different then another drive on the system. This number should be non-zero. Perferable the serial number should be use but this is not necessary.

2.3.3 UIT TABLES

The UIT tables are stored in this program. There are 7 large data tables formed in this diagnostic that contain the drive parameters for certain DEC drives. There are only 4 RQDX3 Winchester drive manufactures. So only 4 of the tables contain any information. The others are there for future drives. If Yes is answered to the Down Line Load question then a table will be DMAed to the disk controller. 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 disk controller. Once in the disk controller the table will be written to the disk drive. This table should never be erased unless the drive is broken or format is run again. If the drive is not recoginized the program will go into manual mode. When in manual mode a list of the drives and assiacated UIT numbers will be displayed. Here you can pick the UIT you want to down line load to the drive.

NOTE this 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:
 5:
 6:
 7:
10:        other

Enter Unit Identifier Table (UIT) (0) ?
```

If you know the name of the drive then just enter the number representing the drive name. If you have a proto type drive then enter "10" representing OTHER.

```
Unit Information/parameter questions, used to build a UIT:
```

If the drive was unidentified by the AUTOSZ program or if you answered other to the manually picked UIT table then

these questions will be asked of you.

DBN size (decimal) (ASCII) value ?
LBN size (decimal) (ASCII) value ?
RBN size (decimal) (ASCII) value ?
Sectors per track (D) value ?
Surfaces per unit (D) value ?
Cylinders per unit (D) value ?
Write precomp cylinder (D) value ?
Reduce write current cylinder (D) value ?
Seek Rate (D) value ?
Use CRC or ECC (D) value ?
Number of RCT copies (D) value ?
Media (lo wrd) (O) value ?
Media (hi wrd) (O) value ?
Sector Interleave (n-to-1) (D) value ?
Surface to Surface Skew (D) value ?
Cylinder to Cylinder Skew (D) value ?
Gap size 0 (D) value ?
Gap size 1 (D) value ?
Gap size 2 (D) value ?
Gap size 3 (D) value ?
Sync size (D) value ?
MSCP cylinders per Unit (D) value ?
MSCP Groups per Cylinder (D) value ?
MSCP Tracks per Group (D) value ?
Max allowed bad spots per surface (D) value ?
Bad spot tolerance (bytes) (D) value ?

There are many questions to build a UIT table. These questions were added mainly to help the engineers use new drives and come up with proper parameters that would optimize the drive to the controller. I would not suggest using this option unless you know MSCP and disk geometry very well. It is possible to patch in the default parameters into the table. The tables address is a UITDF: Once the defaults are patched in, parameters can be changed very easily. UIT0: is located at address 3000 followed by UIT1-7 followed by UITdf.

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 will appear. There may be 60+ minutes between the messages. If the extented messages are allowed 3 "Verification 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 cylinders on the drive. At the end of the format if extended messages are on a table will be printed out reporting the results of the format. Ussually there are several bad spots on a disk. This is very common and is NOT a mistake. These bad blocks are revectored to new areas on the disk. If the manufactures bad block information is used which 's ussually the case. There will only be 1 verification pass.

Completion Report:

```
                        xxx     Revectored LBNs
                        xxx     Primary 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 retired on check pass
                    FCT was not used
                    TEST UNIT 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 purposesly done so one doesn't reformat it twice.

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 between 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 will appear on tre terminal.

3. ERRORS

There are many types of errors possible while formatting a drive. First the system has to be configured right. The drives have to be jumpered right along with the disk controller. If you get an error read the entire error message carefully. See if there is something simple wrong such as loss and misconfigured drives before calling FS. This is ussually the case very seldom do the drive or controller break. So check the cables, check the jumpers, try several times and if you still can't format then call Field Service.

error #	Comment	Problem
0,SF0	;unkown response Not a DUP standard local program or Data Error in local program execution.	
1,HRD0	;Fatal DUP type returned Error with Format program check detailed error message more then likely th's 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,write
                        protected, RX50 instead of an RDxx.

                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 idle state but was found in an active state.
                        Try again and if still there check for ECOs and new Microcode.

                4,DF2   ;wrong step bit set after interrupt
                        Controller initialazation error. Controller is broken or at
                        wrong address and something is in its place.

                5,DF1   ;controller timeout during hard init
                        Controller error, controller is slow or it can't interrupt the
                        Q bus. Controller is dead.

                6,SFT1  ;wrong model #,wrong controller
                        This is not really an error. You are using the wrong formatter
                        program to for the wrong disk controller. It still might work
                        but no guarantees.

                7,DF4   ;NXM trap at controller IP address
                        Wrong configuration address of the controller check for
                        wrong jumper settings.

                8,SF100 ;Unexpected interrupt
                        Something in system interrupting or late interrupt. This
                        could be the system clock or an interrupt from an IO port.
                        If the interrupt is at address 4,10 probable 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
                        Inapropriate 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 probable a drive error
                        or a drive with many revector blocks.

                12,DF14 ;no iterrupt after get dust status command controller dead
                        The controller got lost. The program running in the controller
                        got out of synch with the host program. This could mean several
                        th'ngs. Check for a loose controller 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 r ng 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 controllers 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 and back with out 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 protocol state 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. If an error arrises the program loops until either the formatter works successfully or a the disk controller is considered broken.

5.0 GLOSSARY

ZRQCb0 follows the module name format described in the XXDP Programmer's Guide.

RQ--- Identifies the hardware and thus the module.

--C-- Distiguishes between two or more different diagnostics for the same generic device. The sequence A, B, C, ETC. must be used for each additional diagnostic.

---b- Specifies the module revision.

----0 Specifies the 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 guide (JON::disk$user1:[diagl'b.xxdp])

8.0 REVISION HISTORY

```
574
575                                     Revision B contains an autosizing routine which will
576                                     size the drive instead of having the user pick the drive table.
577                                     This will keep people out of the systems and lower the changes
578                                     of loose cables etc. Also added a AUTO mode which allows no manual
579                                     interventions. Set up the default p-table to format drive 0-3.
580                                     Since floppies are always the last drive in the system this is
581                                     gauranteed to format all the drive in the system and error when it
582                                     gets to the floppy.
583
584                                 )%
```

```
                                .MCALL SVC
000000                          SVC
000000                          .ENABLE ABS,AMA
        002000                  .=2000
002000                          BGNMOD MOD1
002000                          POINTER BGNDU,BGNCLN,BGNPROT,BGNSETUP
002000                          HEADER ZRQC,B,0,600,0
002122                          DISPATCH 1
002126                          DESCRIPT <RQDX3 Disk Format Utility>
002160                          DEVTYPE <RD51,RD52,RD53          *** Answer "Y" to "Change HW (L) ?" ***>

```

```
599 002260                      BGNHW DFPTBL
600 002262  172150                  .WORD   172150          ;IP address
601 002264  000154                  .WORD   154             ;Vector address
602 002266  000000                  .WORD   000000          ;unit zero as defualt drive
603 002270  030071                  .WORD   012345.         ;serial number
604 002272  100000                  .word   100000          ;auto sizer="yes", warning='no  or don't continue
605 002274                      ENDHW
606
```

```
608 002274                          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
    609                                 .sbttl Literals
    610
    611                                 ;+
    612                                 ; Mask values to mask out specified flags
    613                                 ;-
    614         000010                          UITothr = 10            ;UIT other
    615                                                                 ;if UIT doesn't exist
    616
    617                                 ;+
    618                                 ; Misc.
    619                                 ;-
    620         000004                          MaxDrv  = 4             ;Maximum Number of drives
    621         000002                          DUP.id  = bit1          ;DUP connection ID
    622         000007                          Mrqdx1  = 7.            ;model number for RQDX1
    623         000023                          Mrqdx3  = 19.           ;model number for RQDX3
    624         000001                          stdaln  = bit0
    625                                 ;+
    626                                 ; Opcodes for DUP commands
    627                                 ;-
    628         000001                          op.gds  = 1
    629         000006                          op.abrt = 6
    630         000004                          op.sen  = 4
    631         000005                          op.rec  = 5
    632         000003                          op.elp  = 3
    633         000002                          op.esp  = 2
    634         000200                          op.end  = 200
    635                                 ;+
    636                                 ; Message type masks
    637                                 ;-
    638         000001                          Question = 1
    639         000002                          DefQuest = 2
    640         000003                          inform   = 3
    641         000004                          terminat = 4
    642         000005                          ftlerr   = 5
    643         000006                          specl    = 6
    644
    645         177760                          type    = 177760
    646         170000                          msgnbr  = 170000
    647                                 ;+
    648                                 ;Auto sizer literals
    649                                 ;-
    650
    651                                 ; Interrupt Service Routines and Prior'ty Levels
    652
```

Literals

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

                        ; CSRs

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

                        ; RECEIVE DATA ASCII reply message types:

                        .a.typ  =       20                      ; ASCII Message Type Multiplier
                        .a.que  =       1*.a.typ                ; Question
                        .a.def  =       2*.a.typ                ; Default question
                        .a.inf  =       3*.a.typ                ; Information
                        .a.ter  =       4*.a.typ                ; Termination
                        .a.fat  =       5*.a.typ                ; Fatal error

                        ; RECEIVE DATA binary message types.

                        .b.spl  =       6*.a.typ                ; Special

                        ; Status Codes returned by SIZER (Success is zero)

                        erudon          =       1               ; UDC Never Done
                        eruint          =       2               ; UDC Never Interrupted
                        ersek0          =       3               ; Couldn't Restore to Cyl 0

                        ; UDC Commands

                        op.res          =       0               ; Reset 9224
                        op.dd           =       1               ; Deselect Drive
                        op.rd           =       3               ; Restore Drive
                        op.si1          =       5               ; Step In One Cylinder
                        op.sd.rd        =       44              ; Select Drive
                        op.srp          =       100             ; Set Register Pointer
                        rd.mode         =       300             ; RD Mode
```

Macro

```
                                .sbttl  Macro
                                ;+
                                ;       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 where a number (\)
                                .ENDM

                                .MACRO  GDSTMP  B
                                .list
                                GDS'B:  bit     #bit15,cmdrng+2         ;test ownership of ring make sure we own it
                                        bne     GDS'B                   ;if we don't own it wait until we do
                                        mov     #14.,cmdlen             ;load lenght of packet to be send
                                        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.gds,cmdpak+10       ;load up opcode
                                        clr     cmdpak+12               ;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     #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

                                ;+
                                ;       Execute an ABORT command and then checks the response.
                                ;-
                                .MACRO  ABRT            ;Execute an ABORT command
                                B=B+1                           ;increment the CRN number
                                abrttmp \B              ;call variable B as if it where a number (\)
                                .ENDM

                                .MACRO  ABRTTMP B
                                .list
                                ABRT'B: bit     #bit15,cmdrng+2         ;test ownership of ring make sure we own it
                                        bne     ABRT'B                  ;if we don't own it wait until we do
                                        mov     #14.,cmdlen             ;load lenght of packet to be send
                                        movb    #0,cmdlen+2             ;load msg type and credit
```

Macro

```
                        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.abrt,cmdpak+10      ;load up opcode
                        clr     cmdpak+12               ;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     #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 FNDCODE AND STATUS.
                        .nlist
                .ENDM


                ;+
                ;       Execute a Send data cmd in dup and then check the response for the proper info
                ;

                .MACRO  SENDDAT SPLACE,SBYTCN   ;Execute a Send Data command
                B=B+1                           ;increment the CRN number
                sendtmp \B,SPlace,Sbytcn        ;call variable A,B as if it where a number (\)
                .ENDM

                .MACRO  SENDTMP B,Splace,Sbytcnt
                        .list
                SDT'B:  bit     #bit15,cmdrng+2         ;test ownership of ring make sure we own it
                        bne     SDT'B                   ;if we don't own it wait until we do
                        mov     #34,cmdlen              ;load lenght of packet to be send
                        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.sen,cmdpak+10       ;load up opcode
                        clr     cmdpak+12               ;no modifiers
                        mov     Sbytcnt,cmdpak+14
                        clr     cmdpak+16
                        mov     Splace,cmdpak+20        ;load address of buffer describtor
                        clr     cmdpak+22
                        clr     cmdpak+24
                        clr     cmdpak+26
                        clr     cmdpak+30
                        clr     cmdpak+32
```

Macro

```
                        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


                ;+
                ;       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,Rplace,Rbytcnt       ;call variable A,B as if it where 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 do
                        mov     #34,cmdlen              ;load lenght of packet to be send
                        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.
                ;*************************************************************************
```

Macro

```
                        RFD'B:                                  ;INTR TO HERE.
                            add     #6,sp                       ;f'x stack for  nterrupt (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


                    ;+
                    ;       Execute a Receive Data command and the check the response.
                    ;-
                    .MACRO  EXLCPRG Enamadr         ;Execute a Send Data command
                    B=B+1                           ;increment the CRN number
                    elptmp  \B,Enamadr              ;call variable A,B as if it where a number (\)
                    .ENDM

                    .MACRO  ELPTMP  B,Enamadr
                            .list
                    ELP'B:  bit     #bit15,cmdrng+2         ;test ownership of ring make sure we own 't
                            bne     ELP'B                   ;if we don't own it wait until we do
                            mov     #22,cmdlen              ;load lenght of packet to be send
                            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.elp,cmdpak+10       ;load up opcode
                            mov     #stdaln,cmdpak+12       ;stand alone modifier
                            mov     #6,r0                   ;6 letters transfer
                            mov     #cmdpak+14,r1           ;starting address to place program name
                            mov     #Enamadr,r2             ;start of Program Name
                    rfdj'B: movb    (r2)+,(r1)+             ;add 2 to bycnt then store
                            sob     r0,rfdj'B

                            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

```
                        ;+
                        ;       Execute a Receive Data command and the check the response.
                        ;-
                        .MACRO  EYCSUPPRG               ;Execute a Supplied program command
                        B=B+1                           ;increment the CRN number
                        esptmp  \B                      ;call variable A,B as if it where a number (\)
                        .ENDM

                        .MACRO  ESPTMP  B
                                .list
                        ESP'B:  bit     #bit15,cmdrng+2 ;test ownership of ring make sure we own it
                                bne     ESP'B                   ;if we don't own it wait until we do
                                mov     #50,cmdlen              ;load lenght of packet to be send
                                movb    #0,cmdlen+2             ;load msg type and credit value
                                movb    #dup.id,cmdlen+3        ;load DUP connection ID
                                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     #<autoend-autosz>,cmdpak+14 ;load length of prg into buffer
                                clr     cmdpak+16
                                mov     #autosz,cmdpak+20       ;starting address of downline load prg
                                clr     CMDpak+22
                                clr     CMDpak+24
                                clr     CMDpak+26
                                clr     CMDpak+30
                                clr     CMDpak+32

                                clr     CMDpak+34               ;overlay buffer descriptor
                                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 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
```

Word & Buffer defintions

```
                                .sbttl  Word & Buffer defintions

002274  000000          LOGUNIT: .WORD                  ;logunit number
002276  000000          LOCAL:   .WORD                  ;
002300  000000          PLOC:    .WORD                  ;p table address
002302  000000          ptbl:    .WORD                  ;p table address
002304  000000          UITadr:  .word

                        ;+
                        ; 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 f'lled
                        ;  with addresses 177770 and 177772 respectively.
                        ;-
002306  000000          IPreg:   .WORD   0              ;ADDRESS OF THE SA AND IP
002310  000000          Vector:  .word   0
002312  000000          Unit:    .word   0              ;unit number
002314  000123                   .word   123
002316  177777          sernbr:  .word   177777         ;serial number
002320  000000          UNTflgs: .word   0              ;flags, bit15 =auto mode, bit14 ="I'm sure bit"
                                                        ;bit13 =unknown model number
002322  000000          mdlnbr:  .word   0              ;model number of the controller as returned in step 4
002324  000000          mcdnbr:  .word   0              ;micorcode number of the controller as returned in step 4
002326  000000          UIN:     .word   0              ;this is a pointer to the correct UIT table

002330                  RSP1:    .BLKW   2              ;RESPONSE PACKET LENGTH
002334                  RSPPAK:  .BLKW   30.            ;RESPONSE PACKET
002430                  CMDLEN:  .BLKW   2              ;COMMAND PACKET LENGTH
002434                  CMDPAK:  .BLKW   20.            ;COMMAND PACKET

002504  000000          CINTR:   .WORD   0              ;COMMAND INTERRUPT INDICATOR
002506  000000          RINTR:   .WORD   0              ;RESPONSE INTERRUPT INDICATOR
002510  002334          RSPRNG:  .word   rsppak         :MESSAGE RING
002512  140000                   .word   140000
002514  002434          CMDRNG:  .word   cmdpak         ;COMMAND RING
002516  100000                   .word   100000
002520  177777                   .WORD   -1

002522  000000          LSTCRN:  .word   0              ;storage for unreturned command CRN
002524  000000          LSTCMD:  .word   0              ;storage for unreturned command opcode
002526  000000          LSTVCT:  .word   0              ;storage for unreturned command intterupt vector address
002530  000000          LOPRGI:  .word   0              ;Low word of the progress indicator
002532  000000          HIPRGI:  .word   0              ;High word of progress indicator

        .NLIST bin                                      ;data area
002534  DATARE: .asc'z  /%A12345678901234567890123456789012345678901234567890123456789012345678901234567890/
        .even
002660  PRGnam: .ascii  /FORMAT/        ;address of local format program name
002666          .byte   0               ;null for asciz
002667  XBN:    .ASCIZ  /0123456789/
002702  DBN:    .ASCIZ  /0123456789/
002715  LBN:    .ASCIZ  /0123456789/
002730  RBN:    .ASCIZ  /0123456789/
        .even
                        .LIST bin
```

DISK UNIT INFORMATION TABLE

```
                                        .sbttl  DISK UNIT INFORMATION TABLE
                                        ;+
                                        ; The following tables are made up of d'sk dr've parameters which w ll be
                                        ; feed to the FORMAT controller local program which will then use the
                                        ; information to format the d ives.
                                        ;-
                 003000                 .=3000
003000                                  UIT0:
                                        ;+
                                        ;       Unit Information table RD51 Seagate
                                        ;-
                                                                        ;/*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               ;/XBN 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  000462                                  .word   306.            ;/Cylinders per unit/
003026  000156                                  .word   110.            ;/Write precomp cylinder/
003030  000462                                  .word   306.            ;/Reduce write current cylinder /
003032  000000                                  .word   0               ;/Seek Rate/
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
000102                          UITsiz = .-UIT0
                                        .=3000+ UITsiz
003102                          UIT1:
                                        ;+
                                        ;       Unit Information table    RD52 Quantum drive
                                        ;
                                                                        ;/*Top of Unit Information table (UIT)
003102  000066                                  .word   54.             ;/XBN size (lo wrd) XBN size = 3*(1+sectors_per_track)/
003104  000000                                  .word   0               ;/XBN size (hi wrd)/
003106  000100                                  .word   64.             ;/DBN size (lo wrd)/
003110  000000                                  .word   0               ;/DBN size (hi wrd)/
```

DISK UNIT INFORMATION TABLE

```
1096 003112  166114                        .word   60492.          ;/LBN size (lo wrd)/
1097 003114  000000                        .word   0               ;/LBN size (hi wrd)/
1098 003116  000250                        .word   168.            ;/RBN size (lo wrd)/
1099 003120  000000                        .word   0               ;/RBN size (hi wrd)/
1100 003122  000021                        .word   17.             ;/Sectors per track/
1101 003124  000010                        .word   8.              ;/Surfaces per unit/
1102 003126  001000                        .word   512.            ;/Cylinders per unit/
1103 003130  000400                        .word   256.            ;/Write precomp cylinder/
1104 003132  001000                        .word   512.            ;/Reduce write current cylinder /
1105 003134  000000                        .word   0               ;/Seek Rate/
1106 003136  000001                        .word   1               ;/Use CRC or ECC/
1107 003140  000004                        .word   4               ;/RCT Size/
1108 003142  000003                        .word   3               ;/Number of RCT copies/
1109 003144  040064                        .word   ^B0100000000110100 ;^H4034;/Media (lo wrd)/
1110 003146  022544                        .word   ^B0010010101100100 ;^H2564;/Media (hi wrd)/
1111 003150  000001                        .word   1               ;/Sector Interleave (n-to-1)/
1112 003152  000002                        .word   2               ;/Surface to Surface Skew/
1113 003154  000015                        .word   13.             ;/Cylinder to Cylinder Skew/
1114 003156  000020                        .word   16.             ;/Gap size 0/
1115 003160  000020                        .word   16.             ;/Gap size 1/
1116 003162  000005                        .word   5.              ;/Gap size 2/
1117 003164  000050                        .word   40.             ;/Gap size 3/
1118 003166  000015                        .word   13.             ;/Sync size/
1119 003170  000001                        .word   1               ;/MSCP cylinders per Unit/
1120 003172  000001                        .word   1               ;/MSCP Groups per Cylinder/
1121 003174  000001                        .word   1               ;/MSCP Tracks per Group/
1122 003176  000012                        .word   10.             ;/Max allowed bad spots per surface/
1123 003200  000151                        .word   105.            ;/Bad spot tolerance (bytes)/
1124 003202  001000                        .word   512.            ;/auto recal cylinder
1125
1126         003204              .=3000+UITsiz+UITsiz
1127 003204                      UIT2:
1128                             :+
1129                             ;       Unit Information table  RD52 Atasi
1130                             ;-
1131                                                             ;/+Top of Unit Information table (UIT)
1132 003204  000066                        .word   54.             ;/XBN size (lo wrd) XBN size = 3*(1+sectors_per_track)/
1133 003206  000000                        .word   0               ;/XBN size (hi wrd)/
1134 003210  000100                        .word   64.             ;/DBN size (lo wrd)/
1135 003212  000000                        .word   0               ;/DBN size (hi wrd)/
1136 003214  166114                        .word   60492.          ;/LBN size (lo wrd)/
1137 003216  000000                        .word   0               ;/LBN size (hi wrd)/
1138 003220  000250                        .word   168.            ;/RBN size (lo wrd)/
1139 003222  000000                        .word   0               ;/RBN size (hi wrd)/
1140 003224  000021                        .word   17.             ;/Sectors per track/
1141 003226  000007                        .word   7.              ;/Surfaces per unit/
1142 003230  001205                        .word   645.            ;/Cylinders per unit/
1143 003232  000500                        .word   320.            ;/Write precomp cylinder/
1144 003234  001205                        .word   645.            ;/Reduce write current cylinder /
1145 003236  000000                        .word   0               ;/Seek Rate/
1146 003240  000001                        .word   1               ;/Use CRC or ECC/
1147 003242  000004                        .word   4               ;/RCT Size/
1148 003244  000003                        .word   3               ;/Number of RCT copies/
1149 003246  040064                        .word   ^B0100000000110100 ;^H4034;/Media (lo wrd)/
1150 003250  022544                        .word   ^B0010010101100100 ;^H2564;/Media (hi wrd)/
1151 003252  000001                        .word   1               ;/Sector Interleave (n-to-1)/
1152 003254  000002                        .word   2               ;/Surface to Surface Skew/
```

DISK UNIT INFORMATION TABLE

```
1153 003256  000007                        .word   7.              ;/Cylinder to Cylinder Skew/
1154 003260  000020                        .word   16.             ;/Gap size 0/
1155 003262  000020                        .word   16.             ;/Gap size 1/
1156 003264  000005                        .word   5.              ;/Gap size 2/
1157 003266  000050                        .word   40.             ;/Gap size 3/
1158 003270  000015                        .word   13.             ;/Sync size/
1159 003272  000001                        .word   1               ;/MSCP cylinders per Unit/
1160 003274  000001                        .word   1               ;/MSCP Groups per Cylinder/
1161 003276  000001                        .word   1               ;/MSCP Tracks per Group/
1162 003300  000024                        .word   20.             ;/Max allowed bad spots per surface/
1163 003302  000151                        .word   105.            ;/Bad spot tolerance (bytes)/
1164 003304  001206                        .word   646.            ;/auto recal cylinder
1165
1166         003306              .=3000+UITsiz+UITsiz+UITsiz
1167 003306                     UIT3:
1168                             ;+
1169                             ;       Unit Information table  RD53 Micropolis
1170                             ;-
1171                                                             ;/*Top of Unit Information table (UIT)
1172 003306  000066                        .word   54.             ;/XBN size (lo wrd) XBN size = 3*(1+sectors_per_track)/
1173 003310  000000                        .word   0               ;/XBN size (hi wrd)/
1174 003312  000057                        .word   47.             ;/DBN size (lo wrd)/
1175 003314  000000                        .word   0               ;/DBN size (hi wrd)/
1176 003316  016677                        .word   016677          ;/LBN size (lo wrd)/
1177 003320  000002                        .word   2               ;/LBN size (hi wrd)/
1178 003322  000524                        .word   340.            ;/RBN size (lo wrd)/
1179 003324  000000                        .word   0               ;/RBN size (hi wrd)/
1180 003326  000021                        .word   17.             ;/Sectors per track/
1181 003330  000010                        .word   8.              ;/Surfaces per unit/
1182 003332  002000                        .word   1024.           ;/Cylinders per unit/
1183 003334  002000                        .word   1024.           ;/Write precomp cylinder/
1184 003336  002000                        .word   1024.           ;/Reduce write current cylinder /
1185 003340  000000                        .word   0               ;/Seek Rate/
1186 003342  000001                        .word   1               ;/Use CRC or ECC/
1187 003344  000005                        .word   5               ;/RCT Size/
1188 003346  000003                        .word   3               ;/Number of RCT copies/
1189 003350  040065                        .word   ^B0100000000110101 ;^H4035;/Media (lo wrd)/
1190 003352  022544                        .word   ^B0010010101100100 ;^H2564;/Media (hi wrd)/
1191 003354  000001                        .word   1               ;/Sector Interleave (n-to-1)/
1192 003356  000002                        .word   2               ;/Surface to Surface Skew/
1193 003360  000010                        .word   8.              ;/Cylinder to Cylinder Skew/
1194 003362  000020                        .word   16.             ;/Gap size 0/
1195 003364  000020                        .word   16.             ;/Gap size 1/
1196 003366  000005                        .word   5.              ;/Gap size 2/
1197 003370  000050                        .word   40.             ;/Gap size 3/
1198 003372  000015                        .word   13.             ;/Sync size/
1199 003374  000001                        .word   1               ;/MSCP cylinders per Unit/
1200 003376  000001                        .word   1               ;/MSCP Groups per Cylinder/
1201 003400  000001                        .word   1               ;/MSCP Tracks per Group/
1202 003402  000040                        .word   32.             ;/Max allowed bad spots per surface/
1203 003404  000156                        .word   110.            ;/Bad spot tolerance (bytes)/
1204 003406  002000                        .word   1024.           ;/auto recal cylinder
1205
1206         003410              .=3000+UITsiz+UITsiz+UITsiz+UITsiz
1207 003410                     UIT4:
1208                             ;+
1209                             ;       Unit Information table
```

SEQ 0027

DISK UNIT INFORMATION TABLE

```
1210                                    ;
1211                                                    ;*Top of Unit Information table (UIT)
1212 003410  000066                     .word   54.     ;/XBN size (lo wrd) XBN size = 3*(1+sectors_per_track)/
1213 003412  000000                     .word   0       ;/XBN size (hi wrd)/
1214 003414  000057                     .word   47.     ;/DBN size (lo wrd)/
1215 003416  000000                     .word   0       ;/DBN size (hi wrd)/
1216 003420  016677                     .word   016677  ;/LBN size (lo wrd)/
1217 003422  000002                     .word   2       ;/LBN size (hi wrd)/
1218 003424  000524                     .word   340.    ;/RBN size (lo wrd)/
1219 003426  000000                     .word   0       ;/RBN size (hi wrd)/
1220 003430  000021                     .word   17.     ;/Sectors per track/
1221 003432  000010                     .word   8.      ;/Surfaces per unit/
1222 003434  002000                     .word   1024.   ;/Cylinders per unit/
1223 003436  002000                     .word   1024.   ;/Write precomp cylinder/
1224 003440  002000                     .word   1024.   ;/Reduce write current cylinder /
1225 003442  000000                     .word   0       ;/Seek Rate/
1226 003444  000001                     .word   1       ;/Use CRC or ECC/
1227 003446  000005                     .word   5       ;/RCT Size/
1228 003450  000003                     .word   3       ;/Number of RCT copies/
1229 003452  040065                     .word   ^B0100000000110101 ;^H4035;/Media (lo wrd)/
1230 003454  022544                     .word   ^B0010010101100100 ;^H2564;/Media (hi wrd)/
1231 003456  000001                     .word   1       ;/Sector Interleave (n-to-1)/
1232 003460  000002                     .word   2       ;/Surface to Surface Skew/
1233 003462  000010                     .word   8.      ;/Cylinder to Cylinder Skew/
1234 003464  000020                     .word   16.     ;/Gap size 0/
1235 003466  000020                     .word   16.     ;/Gap size 1/
1236 003470  000005                     .word   5.      ;/Gap size 2/
1237 003472  000050                     .word   40.     ;/Gap size 3/
1238 003474  000015                     .word   13.     ;/Sync size/
1239 003476  000001                     .word   1       ;/MSCP cylinders per Unit/
1240 003500  000001                     .word   1       ;/MSCP Groups per Cylinder/
1241 003502  000001                     .word   1       ;/MSCP Tracks per Group/
1242 003504  000040                     .word   32.     ;/Max allowed bad spots per surface/
1243 003506  000156                     .word   110.    ;/Bad spot tolerance (bytes)/
1244 003510  002000                     .word   1024.   ;/auto recal cylinder
1245
1246         003512                 .=3000+UITsiz+UITsiz+UITsiz+UITsiz+UITsiz
1247 003512                         UIT5:
1248                                 ;*
1249                                 ;       Unit Information table
1250                                 ;
1251                                                    ;/*Top of Unit Information table (UIT)
1252 003512  000066                     .word   54.     ;/XBN size (lo wrd) XBN size = 3*(1+sectors_per_track)/
1253 003514  000000                     .word   0       ;/XBN size (hi wrd)/
1254 003516  000057                     .word   47.     ;/DBN size (lo wrd)/
1255 003520  000000                     .word   0       ;/DBN size (hi wrd)/
1256 003522  016677                     .word   016677  ;/LBN size (lo wrd)/
1257 003524  000002                     .word   2       ;/LBN size (hi wrd)/
1258 003526  000524                     .word   340.    ;/RBN size (lo wrd)/
1259 003530  000000                     .word   0       ;/RBN size (hi wrd)/
1260 003532  000021                     .word   17.     ;/Sectors per track/
1261 003534  000010                     .word   8.      ;/Surfaces per unit/
1262 003536  002000                     .word   1024.   ;/Cylinders per unit/
1263 003540  002000                     .word   1024.   ;/Write precomp cylinder/
1264 003542  002000                     .word   1024.   ;/Reduce write current cylinder /
1265 003544  000000                     .word   0       ;/Seek Rate/
1266 003546  000001                     .word   1       ;/Use CRC or ECC/
```

DISK UNIT INFORMATION TABLE

```
 1267 003550  000005                        .word   5               ;/RCT Size/
 1268 003552  000003                        .word   3               ;/Number of RCT copies/
 1269 003554  040065                        .word   ^B0100000000110101 ;^H4035;/Media (lo wrd)/
 1270 003556  022544                        .word   ^B0010010101100100 ;^H2564;/Media (hi wrd)/
 1271 003560  000001                        .word   1               ;/Sector Interleave (n to-1)/
 1272 003562  000002                        .word   2               ;/Surface to Surface Skew/
 1273 003564  000010                        .word   8.              ;/Cylinder to Cylinder Skew/
 1274 003566  000020                        .word   16.             ;/Gap size 0/
 1275 003570  000020                        .word   16.             ;/Gap size 1/
 1276 003572  000005                        .word   5.              ;/Gap size 2/
 1277 003574  000050                        .word   40.             ;/Gap size 3/
 1278 003576  000015                        .word   13.             ;/Sync size/
 1279 003600  000001                        .word   1               ;/MSCP cylinders per Unit/
 1280 003602  000001                        .word   1               ;/MSCP Groups per Cylinder/
 1281 003604  000001                        .word   1               ;/MSCP Tracks per Group/
 1282 003606  000040                        .word   32.             ;/Max allowed bad spots per surface/
 1283 003610  000156                        .word   110.            ;/Bad spot tolerance (bytes)/
 1284 003612  002000                        .word   1024.           ;/auto recal cylinder
 1285
 1286         C03614             .=3000+UITsiz+UITsiz+UITsiz+UITsiz+UITsiz+UITsiz
 1287 003614                    UIT6:
 1288                            ;+
 1289                            ;       Unit Information table
 1290                            ;
 1291                                                            ;/*Top of Unit Information table (UIT)
 1292 003614  000066                        .word   54.             ;/XBN size (lo wrd) XBN size = 3*(1+sectors_per_track)/
 1293 003616  000000                        .word   0               ;/XBN size (hi wrd)/
 1294 003620  000057                        .word   47.             ;/DBN size (lo wrd)/
 1295 003622  000000                        .word   0               ;/DBN size (hi wrd)/
 1296 003624  016677                        .word   016677          ;/LBN size (lo wrd)/
 1297 003626  000002                        .word   2               ;/LBN size (hi wrd)/
 1298 003630  000524                        .word   340.            ;/RBN size (lo wrd)/
 1299 003632  000000                        .word   0               ;/RBN size (hi wrd)/
 1300 003634  000021                        .word   17.             ;/Sectors per track/
 1301 003636  000010                        .word   8.              ;/Surfaces per unit/
 1302 003640  002000                        .word   1024.           ;/Cylinders per unit/
 1303 003642  002000                        .word   1024.           ;/Write precomp cylinder/
 1304 003644  002000                        .word   1024.           ;/Reduce write current cylinder /
 1305 003646  000000                        .word   0               ;/Seek Rate/
 1306 003650  000001                        .word   1               ;/Use CRC or ECC/
 1307 003652  000005                        .word   5               ;/RCT Size/
 1308 003654  000003                        .word   3               ;/Number of RCT copies/
 1309 003656  040065                        .word   ^B0100000000110101 ;^H4035;/Media (lo wrd)/
 1310 003660  022544                        .word   ^B0010010101100100 ;^H2564;/Media (hi wrd)/
 1311 003662  000001                        .word   1               ;/Sector Interleave (n-to-1)/
 1312 003664  000002                        .word   2               ;/Surface to Surface Skew/
 1313 003666  000010                        .word   8.              ;/Cylinder to Cylinder Skew/
 1314 003670  000020                        .word   16.             ;/Gap size 0/
 1315 003672  000020                        .word   16.             ;/Gap size 1/
 1316 003674  000005                        .word   5.              ;/Gap size 2/
 1317 003676  000050                        .word   40.             ;/Gap size 3/
 1318 003700  000015                        .word   13.             ;/Sync size/
 1319 003702  000001                        .word   1               ;/MSCP cylinders per Unit/
 1320 003704  000001                        .word   1               ;/MSCP Groups per Cylinder/
 1321 003706  000001                        .word   1               ;/MSCP Tracks per Group/
 1322 003710  000040                        .word   32.             ;/Max allowed bad spots per surface/
 1323 003712  000156                        .word   110.            ;/Bad spot tolerance (bytes)/
```

SEQ 0029

DISK UNIT INFORMATION TABLE

```
1324 003714  002000                        .word   1024.           ;/auto recal cylinder
1325
1326         003716             .=3000+UITsiz+UITsiz+UITsiz+UITsiz+UITsiz+UITsiz+UITsiz
1327 003716                     UIT7:
1328                            ;+
1329                            ;       Unit Information table
1330                            ;-
1331                                                                ;/*Top of Unit Information table (UIT)
1332 003716  000066                        .word   54.             ;/XBN size (lo wrd) XBN size = 3*(1+sectors_per_track)/
1333 003720  000000                        .word   0               ;/XBN size (hi wrd)/
1334 003722  000057                        .word   47.             ;/DBN size (lo wrd)/
1335 003724  000000                        .word   0               ;/DBN size (hi wrd)/
1336 003726  016677                        .word   016677          ;/LBN size (lo wrd)/
1337 003730  000002                        .word   2               ;/LBN size (hi wrd)/
1338 003732  000524                        .word   340.            ;/RBN size (lo wrd)/
1339 003734  000000                        .word   0               ;/RBN size (hi wrd)/
1340 003736  000021                        .word   17.             ;/Sectors per track/
1341 003740  000010                        .word   8.              ;/Surfaces per unit/
1342 003742  002000                        .word   1024.           ;/Cylinders per unit/
1343 003744  002000                        .word   1024.           ;/Write precomp cylinder/
1344 003746  002000                         word   1024.           ;/Reduce write current cylinder /
1345 003750  000000                        .word   0               ;/Seek Rate/
1346 003752  000001                        .word   1               ;/Use CRC or ECC/
1347 003754  000005                        .word   5               ;/RCT Size/
1348 003756  000003                        .word   3               ;/Number of RCT copies/
1349 003760  040065                        .word   ^B0100000000110101 ;^H4035;/Media (lo wrd)/
1350 003762  022544                        .word   ^B0010010101100100 ;^H2564;/Media (hi wrd)/
1351 003764  000001                        .word   1               ;/Sector Interleave (n-to-1)/
1352 003766  000002                        .word   2               ;/Surface to Surface Skew/
1353 003770  000010                        .word   8.              ;/Cylinder to Cylinder Skew/
1354 003772  000020                        .word   16.             ;/Gap size 0/
1355 003774  000020                        .word   16.             ;/Gap size 1/
1356 003776  000005                        .word   5.              ;/Gap size 2/
1357 004000  000050                        .word   40.             ;/Gap size 3/
1358 004002  000015                        .word   13.             ;/Sync size/
1359 004004  000001                        .word   1               ;/MSCP cylinders per Unit/
1360 004006  000001                        .word   1               ;/MSCP Groups per Cylinder/
1361 004010  000001                        .word   1               ;/MSCP Tracks per Group/
1362 004012  000040                        .word   32.             ;/Max allowed bad spots per surface/
1363 004014  000156                        .word   110.            ;/Bad spot tolerance (bytes)/
1364 004016  002000                        .word   1024.           ;/auto recal cylinder
1365
1366         004020             .=3000+UITsiz+UITsiz+UITsiz+UITsiz+UITsiz+UITsiz+UITsiz+UITsiz
1367 004020                     UITdf:
1368                            ;+
1369                            ;       DEFAULT Unit Information table
1370                            ;-
1371                                                                ;/*Top of Unit Information table (UIT)
1372 004020  000066                        .word   54.             ;/XBN size (lo wrd) XBN size = 3*(1+sectors_per_track)/
1373 004022  000000                        .word   0               ;/XBN size (hi wrd)/
1374 004024  000100                        .word   64.             ;/DBN size (lo wrd)/
1375 004026  000000                        .word   0               ;/DBN size (hi wrd)/
1376 004030  024374                        .word   10492.          ;/LBN size (lo wrd)/
1377 004032  000001                        .word   1               ;/LBN size (hi wrd)/
1378 004034  000250                        .word   168.            ;/RBN size (lo wrd)/
1379 004036  000000                        .word   0               ;/RBN size (hi wrd)/
1380 004040  000021                        .word   17.             ;/Sectors per track/
```

SEQ 0030

DISK UNIT INFORMATION TABLE

```
1381 004042  000010                    .word   8.              ;/Surfaces per unit/
1382 004044  001000                    .word   512.            ;/Cylinders per unit/
1383 004046  000400                    .word   256.            ;/Write precomp cylinder/
1384 004050  001000                    .word   512             ;/Reduce write current cylinder /
1385 004052  000000                    .word   0               ;/Seek Rate/
1386 004054  000001                    .word   1               ;/Use CRC or ECC/
1387 004056  000004                    .word   4               ;/RCT Size/
1388 004060  000003                    .word   3               ;/Number of RCT copies/
1389 004062  040064                    .word   ^B0100000000110100 ;^H4034;/Media (lo wrd)/
1390 004064  022544                    .word   ^B0010010101100100 ;^H2564;/Media (hi wrd)/
1391 004066  000001                    .word   1               ;/Sector Interleave (n-to-1)/
1392 004070  000002                    .word   2               ;/Surface to Surface Skew/
1393 004072  000015                    .word   13.             ;/Cylinder to Cylinder Skew/
1394 004074  000020                    .word   16.             ;/Gap size 0/
1395 004076  000020                    .word   16.             ;/Gap size 1/
1396 004100  000005                    .word   5.              ;/Gap size 2/
1397 004102  000050                    .word   40.             ;/Gap size 3/
1398 004104  000015                    .word   13.             ;/Sync size/
1399 004106  000001                    .word   1               ;/MSCP cylinders per Unit/
1400 004110  000001                    .word   1               ;/MSCP Groups per Cylinder/
1401 004112  000001                    .word   1               ;/MSCP Tracks per Group/
1402 004114  000012                    .word   10.             ;/Max allowed bad spots per surface/
1403 004116  000151                    .word   105.            ;/Bad spot tolerance (bytes)/
1404 004120  002000                    .word   1024.           ;/auto recal cylinder
1405
```

DISK UNIT INFORMATION TABLE

```
1407          .nlist  bin
1408          .sbttl DISK PARAMETER QUESTIONS
1409          ;*
1410          ; P table Questions
1411          ;-
1412 004122  IP.adr:  .ASCIZ  /IP Address/
1413 004135  vec.adr: .ASCIZ  /Vector Address/
1414 004154  drv.nbr: .ASCIZ  /Logical Drive (0-255)/
1415 004202  ser.nbr: .ASCIZ  /Drive Serial Number(1 32000)/
1416 004237  auto.md: .ASCIZ  /Auto Format Mode/
1417 004260  warning: .ASCIZ /***** WARNING all the data on this drive will be DESTROYED ****/
1418 004360  do.cont: .ASCIZ  /Proceed to format the drive/
1419
1420 004414  DrvTxa: .asciz /%N%AUIT    Drive Name/
1421 004442  DrvTxb: .asciz /%N%A_____________________________________________________/
1422 004537  DrvTx0: .asciz /%N%A 0:    RD51 /
1423 004560  DrvTx1: .asciz /%N%A 1:    RD52 part # 30-21721-02 (1 light on front panel)/
1424 004654  DrvTx2: .asciz /%N%A 2:    RD52 part # 30-23227-02 (2 lights on front panel)/
1425 004751  DrvTx3: .asciz /%N%A 3:    RD53 /
1426 004772  DrvTx4: .asciz /%N%A 4:                                                  /
1427 005067  DrvTx5: .asciz /%N%A 5:                                                  /
1428 005164  DrvTx6: .asciz /%N%A 6:                                                  /
1429 005261  DrvTx7: .asciz /%N%A 7:                                                  /
1430 005356  DrvTxc: .asciz /%N%A10:    other%N%N/
1431
1432 005403  Unt.nbr: .ASCIZ  /Enter Unit Identifier Table (UIT)/
1433 005445  ask.prg: .ASCIZ  /What local program do you want to run/
1434 005513  ask.xbn: .ASCIZ  /Enter XBN size in decimal (upto 10 digits)/
1435 005566  ask.dbn: .ASCIZ  /Enter DBN size in decimal (upto 10 digits)/
1436 005641  ask.lbn: .ASCIZ  /Enter LBN size in decimal (upto 10 digits)/
1437 005714  ask.rbn: .ASCIZ  /Enter RBN size in decimal (upto 10 digits)/
1438
1439
1440          ;/*Top of Unit Information table (UIT)
1441 005767  TBQ0:    .ASCIZ  /XBN size (lo wrd) XBN size = 3*(1+sectors_per_track)/
1442 006054  TBQ1:    .ASCIZ  /XBN size (hi wrd)/
1443 006076  TBQ2:    .ASCIZ  /DBN size (lo wrd)/
1444 006120  TBQ3:    .ASCIZ  /DBN size (hi wrd)/
1445 006142  TBQ4:    .ASCIZ  /LBN size (lo wrd)/
1446 006164  TBQ5:    .ASCIZ  /LBN size (hi wrd)/
1447 006206  TBQ6:    .ASCIZ  /RBN size (lo wrd)/
1448 006230  TBQ7:    .ASCIZ  /RBN size (hi wrd)/
1449 006252  TBQ8:    .ASCIZ  /Sectors per track/
1450 006274  TBQ9:    .ASCIZ  /Surfaces per unit/
1451 006316  TBQ10:   .ASCIZ  /Cylinders per unit/
1452 006341  TBQ11:   .ASCIZ  /Write precomp cylinder/
1453 006370  TBQ12:   .ASCIZ  /Reduce write current cylinder /
1454 006427  TBQ13:   .ASCIZ  /Seek Rate/
1455 006441  TBQ14:   .ASCIZ  /Use CRC or ECC/
1456 006460  TBQ15:   .ASCIZ  /RCT Size/
1457 006471  TBQ16:   .ASCIZ  /Number of RCT copies/
1458 006516  TBQ17:   .ASCIZ  /Media (lo wrd)/
1459 006535  TBQ18:   .ASCIZ  /Media (hi wrd)/
1460 006554  TBQ19:   .ASCIZ  /Sector Interleave (n-to-1)/
1461 006607  TBQ20:   .ASCIZ  /Surface to Surface Skew/
1462 006637  TBQ21:   .ASCIZ  /Cylinder to Cylinder Skew/
1463 006671  TBQ22:   .ASCIZ  /Gap size 0/
```

SEQ 0032

DISK PARAMETER QUESTIONS

```
1464 006704  TBQ23:   .ASCIZ  /Gap size 1/
1465 006717  TBQ24:   .ASCIZ  /Gap size 2/
1466 006732  TBQ25:   .ASCIZ  /Gap size 3/
1467 006745  TBQ26:   .ASCIZ  /Sync size/
1468 006757  TBQ28:   .ASCIZ  /MSCP cylinders per Unit/
1469 007007  TBQ29:   .ASCIZ  /MSCP Groups per Cylinder/
1470 007040  TBQ30:   .ASCIZ  /MSCP Tracks per Group/
1471 007066  TBQ31:   .ASCIZ  /Max allowed bad spots per surface/
1472 007130  TBQ32:   .ASCIZ  /Bad spot tolerance (bytes)/
1473
1474 007163  DF1:     .ASCIZ  /Controller Initialization Timeout/
1475 007225  DF2:     .ASCIZ  /Controller never advanced tc next step/
1476 007274  DF3:     .ASCIZ  /Controller can not execute local programs or non STD DUP dialog program/
1477 007404  DF4:     .ASCIZ  /NXM Trap at controllers IP address/
1478         ;DF10:   .ASCIZ  /No Interrupt occurred after SA polled/
1479 007447  DF11:    .ASCIZ  /Bad Response Packet returned/
1480 007504  DF12:    .ASCIZ  /Fatal SA error ctlr offline/
1481 007540  DF13:    .ASCIZ  'No progress shown after a cmd had timed out/
1482 007614  DF14:    .ASCIZ  /GET DUST CMD time_out after another CMD time_out/
1483 007675  DF15:    .ASCIZ  /%N%AFatal error was reported when running local program/
1484 007765  DF16:    .ASCIZ  /%N%AA Special was reported when running local program don't know how to handle it/
1485 010107  SF0:     .ASCIZ  /DUP protocol Error, unexpected message/
1486 010156  SF1:     .ASCIZ  /%N%ASYSTEM is NOT in manual mode/
1487 010217  SF100:   .ASCIZ  /Unexpected or delayed Controller Interrupt/
1488 010272  HPD0:    .ASCIZ  /Fatal Format error/
1489 010315  SFT0:    .ASCIZ  /Controller in an unexpected ACTIVE state/
1490 010366  SFT1:    .ASCIZ  /Wrong Model Number on controller/
1491 010427  PB0:     .ASCIZ  /%N%AModel # listed %06/
1492 010456  PB1:     .ASCIZ  /%N%AExpected SA step bit %06%A,Received in SA %06/
1493 010540  PB3:     .ASCIZ  /%N%AAsking for Format Parameter table/
1494 010606  PB4:     .ASCIZ  /%N%AReceived valid Format Parameter table/
1495 010660  PB5:     .ASCIZ  /%N%AOn UNIT %06%A, %06 Bad Blks were found during Format/
1496 010751  PB6:     .ASCIZ  /%N%AOn UNIT %06%A, %06 Bad Blks were found during Verify pass %06/
1497 011053  PB7:     .ASCIZ  /%N%ADUP Message Type: %06/
1498 011105  PB8:     .ASCIZ  /%N%ADUP message number: %06/
1499 011141  PB9:     .ASCIZ  /%N%AMSCP Controller model # : %D3/
1500 011203  PB10:    .ASCIZ  /%N%A    Microcode version # : %D3/
1501 011245  PB11:    .ASCIZ  /%N%AController is IDLE when it should be ACTIVE running format program/
1502 011354  PB13:    .ASCIZ  /%N/
1503 011357  PF2:     .ASCIZ  /%N%N%AFinished local program without procedure error/
1504 011444  PBF0:    .ASCIZ  /%N%AFormat Parameter table entry at byte %06%N%Ais out of range/
1505 011544  PBF1:    .ASCIZ  /%N%AFormat Parameter table entry at byte %06%N%Ais incompatible with entry at byte %06/
1506 011673  PBF2:    .ASCIZ  /%N%AUNIT %06%A does not exist on controller/
1507 011747  PBF3:    .ASCIZ  /%N%AUNIT %06%A does exist but doesn't respond on controller/
1508 012043  PBF4:    .ASCIZ  /%N%AUNIT %06%A is write protected /
1509 012106  PBF5:    .ASCIZ  /%N%AWrite Fault detected on UNIT %06/
1510 012153  PBF6:    .ASCIZ  /%N%AAttempt to step hd %03%A at cyl %03%A failed on UNIT %06/
1511 012250  PBF7:    .ASCIZ  /%N%AAttempt to format hd %03%A at cyl %03%A failed on UNIT %06/
1512 012347  PBF8:    .ASCIZ  /%N%ATo many Bad Blocks              total Bad Blocks %06/
1513 012437  PBF9:    .ASCIZ  /%N%ADisk Controller model : %D3/
1514 012477  PBF10:   .ASCIZ  /%N%A    Microcode version : %D3/
1515 012537  PB11crn: .ASCIZ  /%N%AExpected CRN %06%A,Received CRN %06/
1516 012607  PB11op:  .ASCIZ  /%N%ACMDpkt Opcode %06%A,RSPpkt Opcode %06/
1517 012661  PB11sts: .ASCIZ  /%N%AResporse pkt status %06/
1518 012715  PB11end: .ASCIZ  /%N%ANo end bit(200) in response packet endcode/
1519 012774  PB11GDS: .ASCIZ  /%N%AGet Dust Status cmd/
1520 013024  PB11ESP: .ASCIZ  /%N%AExecute Supplied Prg cmd/
```

SEQ 0033

DISK PARAMETER QUESTIONS

```
1521 013061  PB11ELP: .ASCIZ  /%N%AExecute Local Prg cmd/
1522 013113  PB11SD: .ASCIZ  /%N%ASend Data cmd/
1523 013135  PB11RD: .ASCIZ  /%N%AReceive Data cmd/
1524 013162  PB11AP: .ASCIZ  /%N%AAbort Prg cmd/
1525 013204  pb11s0: .ASCIZ  /%N%Asts:  successful/
1526 013231  pb11s1: .ASCIZ  /%N%Asts:  Invalid Command/
1527 013263  pb11s2: .ASCIZ  /%N%Asts:  No Region Available/
1528 013321  pb11s3: .ASCIZ  /%N%Asts:  No Region Suitable/
1529 013356  pb11s4: .ASCIZ  /%N%Asts:  Program Not Known/
1530 013412  pb11s5: .ASCIZ  /%N%Asts:  Load Failure/
1531 013441  pb11s6: .ASCIZ  /%N%Asts:  Standalone/
1532 013466  pb11s9: .ASCIZ  /%N%Asts:  Host Buffer Access error/
1533 013531  pb11w0: .ASCIZ  /%N%AUnknown command OPCODE received in timeout loop/
1534 013615  pb11w1: .ASCIZ  /%N%AUnknown command CRN received in command timeout loop/
1535 013706  pb1201: .ASCIZ  /%N%ASA er: Envelope\packet Read (parity or timeout)/
1536 013772  pb1202: .ASCIZ  /%N%ASA er: Envelope\packet Write (parity or timeout)/
1537 014057  pb1203: .ASCIZ  /%N%ASA er: Controller ROM and RAM parity/
1538 014130  pb1204: .ASCIZ  /%N%ASA er: Controller RAM parity/
1539 014171  pb1205: .ASCIZ  /%N%ASA er: Controller ROM parity/
1540 014232  pb1206: .ASCIZ  /%N%ASA er: Queue Read (parity or timeout)/
1541 014304  pb1207: .ASCIZ  /%N%ASA er: Queue Write (parity or timeout)/
1542 014357  pb1208: .ASCIZ  /%N%ASA er: Interrupt Master/
1543 014413  pb1209: .ASCIZ  /%N%ASA er: Host Access Timeout (higher level protocol dependent)/
1544 014514  pb1210: .ASCIZ  /%N%ASA er: Credit Limit Exceeded /
1545 014556  pb1211: .ASCIZ  /%N%ASA er: Bus Master Error/
1546 014612  pb1212: .ASCIZ  /%N%ASA er: Diagnostic Controller Fatal error/
1547 014667  pb1213: .ASCIZ  /%N%ASA er: Instruction Loop Timeout/
1548 014733  pb1214: .ASCIZ  /%N%ASA er: Invalid Connection Identifier/
1549 015004  pb1215: .ASCIZ  /%N%ASA er: Interrupt Write Error/
1550 015045  pb1216: .ASCIZ  /%N%ASA er: MAINTENANCE READ\WRITE Invalid Region Identifier/
1551 015141  pb1217: .ASCIZ  /%N%ASA er: MAINTENANCE WRITE Load to non-loadable controller/
1552 015236  pb1218: .ASCIZ  /%N%ASA er: Controller RAM error (non-parity)/
1553 015313  pb1219: .ASCIZ  /%N%ASA er: INIT sequence error/
1554 015352  pb1220: .ASCIZ  /%N%ASA er: High level protocol incompatibility error/
1555 015437  pb1221: .ASCIZ  /%N%ASA er: Purge\poll hardware failure/
1556 015506  pb1222: .ASCIZ  /%N%ASA er: Mapping Register read error (parity or timeout)/
1557 015601  pb1223: .ASCIZ  /%N%ASA er: Attempt to set port data transfer mapping when option not present/
1558 015716  PB12:   .ASCIZ  /%N%ASA Value (oct) %06/
1559
1560 015745  PBsf0:  .ASCIZ  /%N%ADUP type %06%A message number %06/
1561 016013  DRPunt: .ASCIZ  /%N%ATEST UNIT %06%A, LOGICAL DRIVE %06%A is finished/
1562 016100  TYPASC: .ASCIZ  /%N%APLEASE TYPE ANSWER to controller question or just <return>/
1563          ;
```

```
                        .sbttl  FORMAT Messages

                ; queries

016177  qfuit:  ;.byte  2,,,b.spl                ; Unit Info Table? (spl #2)
016177           .asciz  '%N%AEntering UIT%O2%A: on drive number %D3%N'
016254  qfdat:  ;.byte  0,,,a.que                ; Date? (que #0)
016254           .asciz  'Enter date <MM-DD-YYYY>:'
016305  dfunt:  ;.byte  1,,,a.def                ; Unit? (def #1)
016305           .asciz  'Enter unit number to format <0>:'
016346  dfbad:  ;.byte  4,,,a.def                ; Use Bad? (def #4)
016346           .asciz  'Use existing bad block information <N>:'
016416  dfdwn:  ;.byte  5,,,a.def                ; Downline? (def #5)
016416           .asciz  'Use down-line load <Y>:'
016446  dfcon:  ;.byte  6,,,a.def                ; Continue? (def #6)
016446           .asciz  'Continue if bad block information is inaccessible <Y>:'
016535  qfser:  ;.byte  7,,,a.que                ; Serial #? (que #7)
016535           .asciz  'Enter non-zero serial number <8-10 digits>:'

                ; Informational Messages

016611  sfbegt: ;.byte  0,,,a.inf                ; Begin (inf #0)
016611           .asciz  '%N%AFormat Begun'
016632  sfdont: ;.byte  1,,,a.inf                ; Complete (inf #1)
016632           .asciz  '%N%AFormat complete'
016656  sfrevt: ;.byte  2,,,a.inf                ; # of Revectored LBNs (inf #2)
016656           .asciz  '% Revectored LBNS'
016700  sfr1t:  ;.byte  3,,,a.inf                ; # of primary ... (inf #3)
016700           .asciz  '% Primary revectored LBNS'
016732  sfr2t:  ;.byte  4,,,a.inf                ; # of secondary ... (inf #4)
016732           .asciz  '% Secondary/tertiary revectored LBNS'
016777  sfrcbt: ;.byte  5,,,a.inf                ; # of Bad RCT blocks ... (inf #5)
016777           .asciz  '% Bad blocks in the RCT area due to data errors'
017057  sfdbbt: ;.byte  7,,,a.inf                ; # of Bad DBNs ... (inf #7)
017057           .asciz  '% Bad blocks in the DBN area due to data errors'
017137  sfxbbt: ;.byte  9,,,a.inf                ; # of Bad XBNs ... (inf #9)
017137           .asciz  '% Bad blocks in the XBN area due to data errors'
017217  sftryt: ;.byte  11.,,a.inf               ; # of Retries (inf #11)
017217           .asciz  '% Blocks retried on the check pass'
017262  sfrbbt: ;.byte  14.,,a.inf               ; # of Bad RBNs ... (inf #14)
017262           .asciz  '% Bad RBNS'
017275  sfcylt: ;.byte  15.,,a.inf               ; Formatting Cyl (inf #15)
017275           .asciz  'Formatting Cyl %'
```

FORMAT Messages

```
                        ; Successful Termination Messages
                        ;
                            ;.byte      12.,.a.ter              ; Reformat Worked (ter #12)
017316  sffcut: .asciz  '%N%A'   used successfully'
                        ;
                        ;.byte  13.,.a.ter              ; Reconstruct Worked (ter #13)
017350  sffcnt: .asciz  '%N%AFCT was not used'
                        ;
                        ; Error messages
                        ;
017375  efstat: ;.byte  1.,.a.fat               ; Status Error (fat #1)
017375          .asciz  '%N%AGET STATUS failure'
                        ;
017424  efsndt: ;.byte  2.,.a.fat               ; Send Error (fat #2)
017424          .asciz  '%N%AQ-PORT send error'
                        ;
017452  efcmdt: ;.byte  3.,.a.fat               ; Command Error (fat #3)
017452          .asciz  '%N%AUnsuccessful command'
                        ;
017503  efrcvt: ;.byte  4 .,.a.fat               ; Receive Error (fat #4)
017503          .asciz  '%N%AQ-PORT receive error'
                        ;
017534  efbust: ;.byte  5.,.a.fat               ; Bus Error (fat #5)
017534          .asciz  '%N%AQ-Bus I/O error'
                        ;
017560  efinit: ;.byte  6.,.a.fat               ; Format Init Error (fat #6)
017560          .asciz  '%N%AFo matter initialization error'
                        ;
017623  efnut:  ;.byte  7.,.a. at               ; Unit nonexistent error (fat #7)
017623          .asciz  '%N%AN nexistent unit number'
                        ;
017657  efdxft: ;.byte  8.,.a.fat               ; DBN/XBN Format error (fat #8)
017657          .asciz  '%N%ADBN/XBN format error (drive FORMAT command failed)'
                        ;
017746  effcct: ;.byte  9.,.a.fat               ; FCT copies error (fat #9)
017746          .asciz  '%N%AFCT does not have enough good copies of each block'
                        ;
020035  efsekt: ;.byte  10.,.a.fat              ; Seek error (fat #10)
020035          .asciz  '%N%ASEEK error'
                        ;
020054  efrcct: ;.byte  11.,.a.fat              ; RCT copies error (fat #11)
020054          .asciz  '%N%ARCT does not have enough good copies of each block'
                        ;
020143  eflbft: ;.byte  12.,.a.fat              ; LBN format error (fat #12)
020143          .asciz  '%N%ALBN format error (drive FORMAT command failed)'
                        ;
020226  effcwt: ;.byte  13.,.a.fat              ; FCT write error (fat #13)
020226          .asciz  '%N%AFCT write error (check write protect switch)'
                        ;
020307  efrcrt: ;.byte  14.,.a.fat              ; RCT read error (fat #14)
020307          .asciz  '%N%ARCT read error'
                        ;
020332  efrcwt: ,.byte  15.,.a.fat              ; RCT write error (fat #15)
020332          .asciz  '%N%ARCT write error'
                        ;
020356  efrcft: ;.byte  16.,.a.fat              ; RCT full error (fat #16)
020356          .asciz  '%N%ARCT full'
```

FORMAT Messages

```
1666
1667 020373  effcrt: ;.byte  17.,.a.fat              ; FCT read error (fat #17)
1668 020373           .asciz  '%N%AFCT read error'
1669
1670 020416  effcnt: ;.byte  18.,.a.fat              ; FCT nonexistent error (fat #18)
1671 020416           .asciz  '%N%AFCT nonexistent'
1672
1673 020442  effcdt: ;.byte  19.,.a.fat              ; FCT downline load error (fat #19)
1674 020442           .asciz  '%N%AFCT Down-line load error'
1675
1676 020477  eftmot: ;.byte  20.,.a.fat              ; Drive timeout error (fat #20)
1677 020477           .asciz  '%N%ADrive 'nit timeout'
1678
1679 020526  efillt: ;.byte  21.,.a.fat              ; Illegal response error (fat #21)
1680 020526           .asciz  '%N%AIllegal response to start-up question'
1681
1682 020600  efwart: ;.byte  22.,.a.fat              ; Head error (fat #22)
1683 020600           .asciz  '%N%AWARNING - possible head addressing problem - run diagnostics'
1684
1685 020701  efinpt: ;.byte  23.,.a.fat              ; Input error (fat #23)
1686 020701           .asciz  '%N%AINPUT Error '
1687
1688 020722  efmedt: ;.byte  24.,.a.fat              ; Media error (fat #24)
1689 020722           .asc'z  '%N%AMedia degraded'
1690                                .list   bin
1691                                 .EVEN
```

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 timeout 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 find 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 soon 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
                                ;)>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>waiting for GETDUST status & timed out command responses, if
 no interrupt then fatal err
                                ;) )                              )interrupt received
                                ;) )                              )test for response to GET DUST or TIMED_OUT cmd
                                ;) )                              )handle response here
                                ;) )                              )if response was GET DUST check if we handle the TIMED OUT c
md already
                                ;) )    )<<<<<<<<<<<<<<<<<<<<<<<<)(yes, LSTCRN = 0) go back to DUP dialog mode by checking DU
P type and responding
                                ;) )<<<<<<<<<<<<<<<<<<<<<<<<<<<<)(no,  LSTCRN = 0) 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
                                ;*******************************************************************************
                                POLLWT:

                                pollw:  mtps    #140                            ;DROP CPU LEVEL TO THREE.
                                        tst     @IPreg                          ;TELL MUT TO START POLLING

                                        mov     #60.,r0                 ;outer wait loop
                                1$:     mov     #140,r1                 ;       ;
                                2$:     mtps    #340                    ;don't want interrupts while in other routines
                                        jsr     pc,BIT15T               ;       ;inner wait loop
                                        BREAK                           ;       ;
                                        mtps    #140                    ;turn on interrupts again after check
                                        sob     r1,2$                   ;       ;
                                        sob     r0,1$                   ;
                                                                        ;No response has been received
                                        jsr     pc,BIT15T               ;test SA make sure not a fatal error
                                        mov     cmdpak+10,r0            ;get opcode
                                        cmp     #op.gds,r0              ;if the command issued was a GETDUST STATUS and time
out big trouble
```

Global subroutines

```
   1750 021024  001006                          bne     GDS0                    ;if not go do a GET DUST to find out what the situat
ion is
   1751 021026                                  ERRDF   12,df14                 ;type no iterrupt after get dust status command cont
roller dead
   1752 021036  000137  034572                  jmp     dropunt                 ;drop unit and go or
   1753
   1754                                 ;GETDUST                                ;save timed out command information
   1755 021042  017737  161242  002526 GDS0:   mov     @vector,LSTVCT          ;store the vector address of timeout command
   1756 021050  013737  002434  002522          mov     cmdpak,LSTCRN           ;store the CRN of the timed out command
   1757 021056  013737  002444  002524          mov     cmdpak+10,LSTCMD        ;store the opcode of timed out command
   1758
   1759 021064  032737  100000  002516          bit     #bit15,cmdrng+2         ;test ownership of ring make sure we own it
   1760 021072  001363                          bne     GDS0                    ;if we don't own it wait until we do
   1761 021074  012737  000016  002430          mov     #14.,cmdlen             ;load lenght of packet to be send
   1762 021102  112737  000000  002432          movb    #0,cmdlen+2             ;load msg type and credit
   1763 021110  112737  000002  002433          movb    #dup.id,cmdlen+3        ;load DUP connection ID
   1764 021116  005237  002434                  inc     cmdpak                  ;load new CRN
   1765 021122  005037  002436                  clr     cmdpak+2
   1766 021126  005037  002440                  clr     cmdpak+4
   1767 021132  005037  002442                  clr     cmdpak+6
   1768 021136  012737  000001  002444          mov     #op.gds,cmdpak+10       ;load up opcode
   1769 021144  005037  002446                  clr     cmdpak+12               ;no modifiers
   1770
   1771 021150  012777  021212  161132          mov     #RFD0,@vector           ;NEW VECTOR PLACE
   1772 021156  012737  002334  002510          mov     #rsppak,rsprng          ;load response packet area into ring
   1773 021164  012737  002434  002514          mov     #cmdpak,cmdrng          ;load command packet area into ring
   1774 021172  012737  140000  002512          mov     #140000,RSPRNG+2        ;PORT OWNERSHIP BIT.
   1775 021200  012737  100000  002516          mov     #bit15,CMDRNG+2
   1776 021206  004737  020746                  jsr     pc,POLLWT                       ;GO and wait for interrupt
   1777
   1778
   1779
   1780                                 ;*************************************************************************
   1781 021212                             RFD0:                                ;INTR TO HERE if GETDUST or TIMED_OUT cmd
   1782                                 ;+
   1783                                 ; There is only 3 ways out code.
   1784                                 ;If GETDUST response and TIMED_OUT cmd response was hand'ed
   1785                                 ; if LSTCRN = 0 and RSPPAK+10 = OP.GDS+OP.END then
   1786                                 ; back to DUP dialog mode.
   1787                                 ;or
   1788                                 ; (TIMED_OUT cmd still hasn't returned but GETDUST has returned)
   1789                                 ; if LSTCRN = # and RSPPAK+10 = OP.GDS+OP.END then
   1790                                 ; check if idle or active. if idle then error
   1791                                 ; check for progress in progress indicator if no progress then error
   1792                                 ; load LSTVCT into @vector,LSTCRN into cmdpak, LSTCMD into cmdpak+10
   1793                                 ; set response ring ownership to Port Owned
   1794                                 ; jmp to pollwt.
   1795                                 ;or
   1796                                 ; (TIMED_OUT cmd response recieved before GETDUST response returned)
   1797                                 ; if LSTCRN = # and RSPPAK+10 not= OP.GDS+OP.END then
   1798                                 ; clear LSTCRN and
   1799                                 ; jmp to pollwt.
   1800
   1801 021212  013701  002434                  mov     cmdpak,r1               ;check command packet CRN
   1802 021216  013700  002334                  mov     rsppak,r0               ;check response packet CRN
   1803 021222  020001                          cmp     r0,r1                   ;Are they the SAME must be GETDUST cmd
   1804 021224  001101                          bne     3$                      ;if not it must be the TIMED_OUT cmd
   1805
   1806 021226  023727  002344  000201          cmp     rsppak+10,#op.gds+op.end         ;it should be a GETDUST lets make sure
```

Global subroutines

```
   1807 021234  001412                          beq     1$
   1808 021236                                  printf  #pb11w0         ;unexpected cmd response in time out loop
   1809 021256  000137  034556                  jmp     unkwn           ;error handler
   1810
   1811 021262  004737  026226          1$:     jsr     pc,RSPCHK       ;check the response
   1812 021266  005737  002522                  tst     LSTCRN          ;see if timed out command was already recieved (lstc
rn = 0)
   1813 021272  001002                          bne     2$
   1814 021274  000137  031406                  jmp     DUPDLG          ;if Timed out cmd was already received then goto DUP
 dialog mode
   1815
   1816 021300                          2$:                             ;if Timed out command was not received already (LSTC
RN not= 0)
   1817 021300  132737  000010  002353          bitb    #bit3,rsppak+17 ;if server idle then error
   1818 021306  001010                          bne     1002$           ;if not check for progress
   1819 021310                                  printf  #pb11           ;controller idle when it should be active
   1820
   1821 021330  013700  002354          1002$:  mov     rsppak+20,r0    ;check for progress in progress indicator
   1822 021334  013701  002356                  mov     rsppak+22,r1
   1823 021340  020037  002530                  cmp     r0,loprgi       ;see if low word of progress indicator is the same a
s older value
   1824 021344  001007                          bne     1001$           ;if it is then continue
   1825 021346  020137  002532                  cmp     r1,hiprgi       ;see if high vaule is the same
   1826 021352  001004                          bne     1001$
   1827 021354                                  ERRDF   11,DF13         ;no progress shown after cmd timeout
   1828
   1829 021364  010037  002530          1001$:  mov     r0,loprgi       ;update progress indicator
   1830 021370  010137  002532                  mov     r1,hiprgi
   1831 021374  013737  002522  002434          mov     LSTCRN,cmdpak   ;move TIMED_OUT cmd CRN into cmd
   1832 021402  013737  002524  002444          mov     LSTCMD,cmdpak+10 ;move TIMED_OUT cmd Opcode into cmd
   1833 021410  013777  002526  160672          mov     LSTVCT,@vector  ;load TIMED_OUT cmd interrupt handler address into v
ector
   1834 021416  012737  140000  002512          mov     #140000,RSPRNG+2 ;Port owned
   1835 021424  000137  020746                  jmp     pollw           ;wait for TIMED_OUT cmd response
   1836
   1837
   1838
   1839 021430  020037  002522          3$:     cmp     r0,LSTCRN       ;check the crn with the last CRN from the timeout co
mmand
   1840 021434  001412                          beq     4$              ;
   1841 021436                                  printf  #pb11w1         ;Unexpected cmd response in time out loop
   1842 021456  000137  034556                  jmp     unkwn           ;error handler
   1843
   1844                                                                 ;Timed out command recieved but Get Dust Status is s
till in Queue
   1845 021462  013737  002522  002434  4$:     mov     LSTCRN,cmdpak   ;load timed out command values for RSPCHK routine
   1846 021470  013737  002524  002444          mov     LSTCMD,cmdpak+10 ;load timed out command values for RSPCHK routine
   1847 021476  005037  002522                  clr     LSTCRN          ;if it is the timeout command clear LAST CRN registe
r
   1848 021502  004737  026226                  jsr     pc,RSPCHK       ;go check the command
   1849 021506  012737  140000  002512          mov     #140000,RSPRNG+2 ;PORT OWNERSHIP BIT.
   1850 021514  000137  020746                  jmp     POLLW           ;go wait for GETDUST intterupt
```

SEQ 0040

Global subroutines

```
   1852                                 ;*****************************************************************************
   1853                                 ;       HARD INITIALIZE
   1854                                 ; This routine hard initialize the disk controller so that DUP commands
   1855                                 ; can be issued. This routine is governed by the UQSSP spec.
   1856                                 ; This format starts by initializating
   1857                                 ;       r1 = ptable address
   1858                                 ;       r3 = step bit mask
   1859                                 ;       r4 = SA address
   1860                                 ;*****************************************************************************
   1861 021520                         HRDINT:
   1862 021520  106427  000140                 mtps    #140
   1863 021524  013704  002306                 mov     ipreg,r4
   1864 021530  012703  004000                 mov     #bit11,r3       ;initialization step reg
   1865 021534  005024                 1$:     clr     (r4)+           ; start initialaztion IP and put SA adrs into r4
   1866 021536                                 Break
   1867
   1868 021540  012700  040000                 mov     #40000,r0       ;looking for step 1 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>
   1869 021544  004737  027160         4$:     jsr     pc,bit15T       ;Look for fatal SAreg error
   1870 021550  030314                         bit     r3,(r4)         ;check sa reg for step 1        bit11 =0 >1
   1871 021552  001002                         bne     6$
   1872 021554  077005                         sob     r0,4$
   1873 021556  000560                         br      timout
   1874
   1875 021560  013700  002310         6$:     MOV     vector,r0       ;responding in step 1 <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<
   1876 021564  000241                         clc
   1877 021566  006200                         asr     r0
   1878 021570  006200                         asr     r0
   1879 021572  052700  100200                 bis     #<bit15+bit7>,r0 ;= VECTOR/4
   1880 021576  013701  002310                 mov     vector,r1
   1881 021602  012721  021642                 mov     #sp2int,(r1)+   ;load up interrupt location into vector
   1882 021606  012711  000140                 mov     #140,(r1)       ;lower the priority
   1883 021612  010014                         mov     r0,(r4)         ; Enable INTERRUPTS,set 1 cmd rng & 1 rsp rng
   1884
   1885 021614  006303                         asl     r3              ;next step
   1886 021616  012700  070000                 mov     #70000,r0       ;looking for step 2 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>
   1887 021622  106427  000340         8$:     mtps    #340
   1888 021626                                 break                   ;waste some time
   1889 021630                                 break                   ;waste some time
   1890 021632  106427  000140                 mtps    #140
   1891 021636  077007                         sob     r0,8$           ;wait ten seconds
   1892 021640  000527                         br      timout          ;timed out if here
   1893
   1894 021642  062706  000004         sp2int: add     #4,sp
   1895 021646  004737  027160                 jsr     pc,bit15T       ;look for sa error
   1896 021652  030314                         bit     r3,(r4)         ;check step
   1897 021654  001541                         beq     wrngstep
   1898
   1899 021656  012700  002510         12$:    mov     #RSPRNG,R0      ;responding in step 2 <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<
   1900 021662  042700  000001                 bic     #bit0,r0        ;no adapter purge interrupts
   1901 021666  013701  002310                 mov     vector,r1
   1902 021672  012721  021732                 mov     #sp3int,(r1)+   ;load up interrupt location into vector
   1903 021676  012711  000140                 mov     #140,(r1)       ;lower the priority
   1904 021702  010014                         mov     r0,(r4)         ;load low ringbase address of the communications area
   1905
   1906 021704  006303                         asl     r3              ;next step
   1907 021706  012700  070000                 mov     #70000,r0       ;looking for step 3 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>
   1908 021712  106427  000340         13$:    mtps    #340
```

```
021716                               break                       ;waste some time
021720                               break                       ;waste some time
021722  106427  000140               mtps     #140
021726  077007                       sob      r0,13$             ;wait ten seconds
021730  000473                       br       timout

021732  062706  000004       sp3int:add       #4,sp
021736  004737  027160               jsr      pc,bit15T
021742  030314                       bit      r3,(r4)
021744  001505                       beq      wrngstep

021746  013701  002310               mov      vector,r1          ;responding in step 3 <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< <<<<<<<
<<<<<<<<<<<<<<<<<
021752  012721  022012               mov      #sp4int,(r1)+      ;load up interrupt loacation into vector
021756  012711  000340               mov      #340,(r1)          ;after step four we want no interrupts until expected.
021762  005014                       clr      (r4)               ;load low ringbase address of the communications area

021764  006303                       asl      r3
021766  012701  070000               mov      #70000,r1          ;looking for step 4 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>
021772  106427  000340       18$:    mtps     #340
021776                               break                       ;waste some time
022000                               break                       ;waste some time
022002  106427  000140               mtps     #140
022006  077107                       sob      r1,18$             ;wait ten seconds
022010  000443                       br       timout

022012  062706  000004       sp4int: add      #4,sp
022016  004737  027160               jsr      pc,bit15T          ;check for SA error
022022  030314                       bit      r3,(r4)            ;check for step 4
022024  001455                       beq      wrngstep

022026  011401                       mov      (r4),r1            ;identify the controller number and microcode version number
022030  010102                       mov      r1,r2
022032  006201                       asr      r1                 ;shift version # out
022034  006201                       asr      r1
022036  006201                       asr      r1
022040  006201                       asr      r1
022042  042701  177400               bic      #177400,r1         ;clear top bits off
022046  010137  002322               mov      r1,mdlnbr          ;model number storage
022052  042702  177760               bic      #177760,r2         ;clear model number out
022056  010237  002324               mov      r2,mcdnbr
022062  122701  000007               cmpb     #Mrqdx1,r1         ;check for Model #
022066  001454                       beq      gobit
022070  122701  000023               cmpb     #Mrqdx3,r1         ;check for Model #
022074  001451                       beq      gobit

022076                               ERRSOFT 6,SFT1              ; DEVICE FATAL wrong model #,wrong controller
022106  052737  020000  002320       bis      #bit13,UNTflgs     ;set unknown model number in unit flags
022114  000137  022220               jmp      gobit              ;drop unit and go on

022120                       timout: ERRDF    5,DF1              ; DEVICE FATAL controller timeout during hard init
022130                               Printf   #pb1,r3,(r4)       ; Expected SA step bit xxxxx set, received yyyyyy
022154  000137  034572               jmp      dropunt                    ;drop unit and go on

022160                       wrngstep:
022160                               ERRDF    4,DF2              ; DEVICE FATAL wrong step bit set after interrupt
022170                               Printf   #pb1,r3,(r4)       ; Expected SA step bit xxxxx, received in SA yyyyyy
022214  000137  034572               jmp      dropunt                    ;drop unit and go on
```

Global subroutines

```
1966
1967 022220  012714  000001          GOBIT:  mov     #1,(r4)                 ;Controller 's NOW INITIALIZED
1968
1969 022224  012700  177777                  mov     #-1,r0
1970 022230  000240                  1$:     nop                             ;waste just a little t'me so program can terminate
1971 022232  077002                          sob     r0,1$
1972 022234                          GDScmd:
1973 022234                          GETDUST                                 ;Do a Get Dust Status command start things off
     022234  032737  100000  002516 GDS2:   bit     #bit15,cmdrng+2         ;test ownership of ring make sure we own it
     022242  001374                        bne     GDS2                    ;if we don't own it wait until we do
     022244  012737  000016  002430        mov     #14.,cmdlen             ;load lenght of packet to be send
     022252  112737  000000  002432        movb    #0,cmdlen+2             ;load msg type and credit
     022260  112737  000002  002433        movb    #dup.id,cmdlen+3        ;load DUP connection ID
     022266  005237  002434                inc     cmdpak                  ;load new CRN
     022272  005037  002436                clr     cmdpak+2
     022276  005037  002440                clr     cmdpak+4
     022302  005037  002442                clr     cmdpak+6
     022306  012737  000001  002444        mov     #op.gds,cmdpak+10       ;load up opcode
     022314  005037  002446                clr     cmdpak+12               ;no modifiers

     022320  012777  022362  157762        mov     #RFD2,@vector           ;NEW VECTOR PLACE
     022326  012737  002334  002510        mov     #rsppak,rsprng          ;load response packet area into ring
     022334  012737  002434  002514        mov     #cmdpak,cmdrng          ;load command packet area into ring
     022342  012737  140000  002512        mov     #140000,RSPRNG+2        ;PORT OWNERSHIP BIT.
     022350  012737  100000  002516        mov     #bit15,CMDRNG+2
     022356  004737  020746                jsr     pc,POLLWT               ;GO TO POLL AND WAIT ROUTINE.
                                  ;*************************************************************************
     022362                          RFD2:                                   ;INTR TO HERE.
     022362  062706  000006                add     #6,sp                   ;fix stack for interrupt (4), pollwt subrtn (2)
     022366  012777  030222  157714        mov     #intsrv,@vector         ;CHANGE VECTOR
     022374  004737  026226                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.
1974 022400  132737  000010  002353          bitb    #bit3,rsppak+17         ;is this server active already
1975 022406  001467                          beq     dnint                   ;branch to Execute Local Program
1976 022410                                  ERRSOFT 3,SFTO                  ;Soft Error "already active  will do an ABORT cmd'
1977 022420                          ABRT                                    ;Doing an ABRT do get into idle state
     022420  032737  10C00C  002516 ABRT3: bit     #bit15,cmdrng+2         ;test ownership of ring make sure we own it
     022426  001374                        bne     ABRT3                   ;if we don't own it wait until we do
     022430  012737  000016  002430        mov     #14.,cmdlen             ;load lenght of packet to be send
     022436  112737  000000  002432        movb    #0,cmdlen+2             ;load msg type and credit
     022444  112737  000002  002433        movb    #dup.id,cmdlen+3        ;load DUP connection ID
     022452  005237  002434                inc     cmdpak                  ;load new CRN
     022456  005037  002436                clr     cmdpak+2
     022462  005037  002440                clr     cmdpak+4
     022466  005037  002442                clr     cmdpak+6
     022472  012737  000006  002444        mov     #op.abrt,cmdpak+10      ;load up opcode
     022500  005C37  002446                clr     cmdpak+12               ;no modifiers

     022504  012777  022546  157576        mov     #RFD3,@vector           ;NEW VECTOR PLACE
     022512  012737  002334  002510        mov     #rsppak,rsprng          ;load response packet area into ring
     022520  012737  002434  002514        mov     #cmdpak,cmdrng          ;load command packet area into ring
     022526  012737  140000  002512        mov     #140000,RSPRNG+2        ;PORT OWNERSHIP BIT.
     022534  012737  100000  0C2516        mov     #bit15,CMDRNG+2
     022542  004737  020746                jsr     pc,POLLWT               ;GO TO POLL AND WAIT ROUTINE.
```

SEQ 0043

Global subroutines

```
                                ;*****************************************************************************
        022546                      RFD3:                                   ;INT.. TO HERE.
        022546  062706  000006              add     #6,sp                   ;fix stack for interrupt (4), pollwt subrtn (2)
        022552  012777  030222  157530      mov     #intsrv,@vector         ;CHANGE VECTOR
        022560  004737  026226              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.
  1978 022564  000623                      br      GDScmd                  ;branch back to make sure not busy
  1979 022566                       DNINT:
  1980 022566  000207                      rts     pc
```

Global subroutines

```
1982                                    ;*****************************************************************************
1983                                    ;                           AUTOsizer
1984                                    ;       This routine runs the Execute Supplied program called AUTOSZ
1985                                    ; This program is downline loaded into the controller to determine
1986                                    ; which drive is out in the controller. First you must tell which drive
1987                                    ; you want to format. After listing the drive number the program will load
1988                                    ; the program and figure which DEC drive it is and which UIT to load into
1989                                    ; the disk controller fo rthe format program.
1990                                    ;
1991                                    ;*****************************************************************************
1992 022570                            AUTOSizer:
1993 022570                                excSUPprg                       ;downline load the program autosz
     022570  032737  100000  002516  ESP4:   bit     #bit15,cmdrng+2 ;test ownership of ring make sure we own it
     022576  001374                          bne     ESP4                    ;if we don't own it wait until we do
     022600  012737  000050  002430          mov     #50,cmdlen              ;load lenght of packet to be send
     022606  112737  000000  002432          movb    #0,cmdlen+2             ;load msg type and credit value
     022614  112737  000002  002433          movb    #dup.id,cmdlen+3        ;load DUP connection ID
     022622  005037  002436                  clr     CMDpak+2
     022626  005037  002440                  clr     CMDpak+4
     022632  005037  002442                  clr     CMDpak+6
     022636  012737  000002  002444          mov     #op.esp,CMDpak+10       ;load up opcode
     022644  012737  000000  002446          mov     #0,CMDpak+12            ;no stand alone modifier
     022652  012737  000714  002450          mov     #<autoend-autosz>,cmdpak+14 ;load length of prg into buffer
     022660  005037  002452                  clr     cmdpak+16
     022664  012737  023240  002454          mov     #autosz,cmdpak+20       ;starting address of downline load prg
     022672  005037  002456                  clr     CMDpak+22
     022676  005037  002460                  clr     CMDpak+24
     022702  005037  002462                  clr     CMDpak+26
     022706  005037  002464                  clr     CMDpak+30
     022712  005037  002466                  clr     CMDpak+32

     022716  005037  002470                  clr     CMDpak+34               ;overlay buffer descriptor
     022722  005037  002472                  clr     CMDpak+36
     022726  005037  002474                  clr     CMDpak+40
     022732  005037  002476                  clr     CMDpak+42
     022736  005037  002500                  clr     CMDpak+44
     022742  005037  002502                  clr     CMDpak+46

     022746  012777  023010  157334          mov     #RFD4,@vector           ;NEW VECTOR PLACE
     022754  012737  002334  002510          mov     #rsppak,rsprng          ;load response packet area into ring
     022762  012737  002434  002514          mov     #cmdpak,cmdrng          ;load command packet area into ring
     022770  012737  140000  002512          mov     #140000,RSPRNG+2        ;PURT OWNERSHIP BIT.
     022776  012737  100000  002516          mov     #bit15,CMDRNG+2
     023004  004737  020746                  jsr     pc,POLLWT               ;GO TO POLL AND WAIT ROUTINE.
                                        ;*****************************************************************************
     023010                                 RFD4:                           ;INTR TO HERE.
     023010  062706  000006                  add     #6,sp                   ;fix stack for interrupt (4), pollwt subrtn (2)
     023014  012777  030222  157266          mov     #intsrv,@vector         ;CHANGE VECTOR
     023022  004737  026226                  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.
1994 023026                                 Recvdata    #msg,#msglen    ;get results of auto size
     023026  032737  100000  002516  RCD5:   bit     #bit15,cmdrng+2         ;test ownership of ring make sure we own it
     023034  001374                          bne     RCD5                    ;if we don't own it wait until we do
     023036  012737  000034  002430          mov     #34,cmdlen              ;load lenght of packet to be send
```

Global subroutines

```
023044  112737  000000  002432          movb    #0,cmdlen+2             ;load msg type and credit
023052  112737  000002  002433          movb    #dup.id,cmdlen+3        ;load DUP connection ID
023060  005237  002434                  inc     cmdpak                  ;load new CRN
023064  005037  002436                  clr     cmdpak+2
023070  005037  002440                  clr     cmdpak+4
023074  005037  002442                  clr     cmdpak+6
023100  012737  000005  002444          mov     #cp.rec,cmdpak+10       ;load up opcode
023106  005037  002446                  clr     cmdpak+12               ;no modifiers
023112  012737  000014  002450          mov     #msglen,cmdpak+14
023120  005037  002452                  clr     cmdpak+16
023124  012737  024140  002454          mov     #msg,cmdpak+20  ;load address of buffer describtor
023132  005037  002456                  clr     cmdpak+22
023136  005037  002460                  clr     cmdpak+24
023142  005037  002462                  clr     cmdpak+26
023146  005037  002464                  clr     cmdpak+30
023152  005037  002466                  clr     cmdpak+32

023156  012777  023220  157124          mov     #RFD5,@vector           ;NEW VECTOR PLACE
023164  012737  002334  002510          mov     #rsppak,rsprng          ;load response packet area into ring
023172  012737  002434  002514          mov     #cmdpak,cmdrng          ;load command packet area into ring
023200  012737  140000  002512          mov     #140000,RSPRNG+2        ;PORT OWNERSHIP BIT.
023206  012737  100000  002516          mov     #bit15,CMDRNG+2
023214  004737  020746                  jsr     pc,POLLWT               ;GO TO POLL AND WAIT ROUTINE.
                        ;*************************************************************************
023220                          RFD5:                                   ;INTR TO HERE.
023220  062706  000006                  add     #6,sp                   ;fix stack for interrupt (4), pollwt subrtn (2)
023224  012777  030222  157056          mov     #intsrv,@vector         ;CHANGE VECTOR
023232  004737  026226                  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.
1995 023236  000207                     rts     pc              ;return
1996
1997                            .sbttl autosz
1998                            ;*************************************************************************
1999                            ;                       AUTOsz
2000                            ;       This is the actual down line loaded code which is placed in
2001                            ; the RAM inside the RQDX3 controller. This code firgures out the
2002                            ; cylinder size of the drive. From the cylinder size we can determine
2003                            ; which drive it is.
2004                            ;+
2005                            ; SIZER - Determine Drive Type and Size (-1 for non-Winnie).
2006                            ;
2007                            ; Input:        None.
2008                            ;
2009                            ; Output:
2010                            ;               A Special Type Message:
2011                            ;
2012                            ;               +-----------------+-----------+
2013                            ;               )   Special Msg #10 (decimal)   ) +00
2014                            ;               +-----------------+-----------+
2015                            ;               )           Status              ) +02
2016                            ;               +-----------------------------+
2017                            ;               ) Innermost Cylinder for Unit 0 ) +04
2018                            ;               +-----------------------------+
2019                            ;               ) Innermost Cylinder for Unit 1 ) +06
```

```
 2020                                   ;               +-- ------- --- --  ------------+
 2021                                   ;               ) Innermost Cylinder for Unit 2 ) +10
 2022                                   ;               +------  - ---------------------+
 2023                                   ;               ) Innermost Cylinder for Unit 3 ) +12
 2024                                   ;               +------- --- ---  -------------+
 2025                                   ;
 2026                                   ;               where,  status      =   0 for success,
 2027                                   ;                                       1 for UDC never went done,
 2028                                   ;                                       2 for UDC never interrupted,
 2029                                   ;                                       3 for Seek Failed
 2030                                   ;
 2031                                   ;                       cylinder    =   0 to 2048 for Winnie,
 2032                                   ;                                       -1 for non-Winnie or "nothing"
 2033                                   ;
 2034                                   ; Note. The Unit Numbers will correspond to the numbers that the Host
 2035                                   ;       would use (i.e., not necessarily the DRVSEL numbers).  Thus,
 2036                                   ;       Winnies will always precede Floppies and "null devices".
 2037                                   ;-
 2038                                   ;*****************************************************************
 2039 023240                           AUTOsz:
 2040                                   .dsable AMA
 2041 023240  000714                            .word   <AUTOend-AUTOsz>        ;Byte count low         TEST HEADER
 2042 023242  000000                            .word   0                       ;byte count high
 2043 023244  000000                            .word   0                       ;overlay low
 2044 023246  000000                            .word   0                       ;overlay high
 2045 023250     101     125     124            .ascii  /AUTOSZ/                ;6 character asciz name
      023253     117     123     132
 2046                                           .even
 2047 023256  000001                            .word   1                       ;version number
 2048 023260     000                            .byte   0                       ;flags
 2049 023261     177                            .byte   177                     ;timeout
 2050 023262  000240                            nop                             ;start down line loaded test
 2051
 2052 023264                           AUTO::
 2053 023264  000240                            nop                             ;start down line loaded test
 2054
 2055                                   ; Executable Code Starts Here
 2056
 2057 023266  106427  000340                     mtps    #ps7                    ;;; Set up our own interrupts handlers
 2058 023272  005037  140004                     clr     @#w$fpi                 ;;; clear the leds
 2059 023276  013746  100002                     mov     @#i$udc,-(sp)           ;;; Save the MSCP handlers - UDC
 2060 023302  013746  100006                     mov     @#i$clk,-(sp)           ;;; ... Clock
 2061 023306  013746  100016                     mov     @#i$sec,-(sp)           ;;; ... Sector
 2062
 2063                                   ; Taken from RQDX3.MAC m$init code:
 2064
 2065 023312  112737  000000  140022             movb    #op.res,@#w$cmd         ;;; reset the smc9224 chip
 2066 023320  005067  000610                     clr     s$$bug                  ;;; assume the bug is not present
 2067 023324  032737  020000  140006             bit     #20000,@#r$fps          ;;; is the ECO wire there?
 2068 023332  001423                             beq     sizset                  ;;; definitely not
 2069 023334  112737  000111  140022             movb    #op.srp+11,@#w$cmd      ;;; enable interrupts
 2070 023342  112737  000040  140020             movb    #40,@#w$dat             ;;;
 2071 023350  112737  000001  140022             movb    #op.dd,@#w$cmd          ;;; deselect all drives
 2072
 2073 023356  012700  001000                     mov     #1000,r0                ;;; wait for a bit
 2074 023362                           sizwt:                                    ;;; ...
 2075 023362  005300                             dec     r0                      ;;; ...
```

Autosz

```
2075 023364  001376                    bne     sizwt                   ;;; ...
2077
2078 023366  032737  020000  140006    bit     #20000,@#r$fps          ;;; is the ECO wire there?
2079 023374  001002                    bne     sizset                  ;;; nope
2080 023376  005267  000532            inc     s$$bug                  ;;; say it is
2081
2082 023402                    sizset:                                 ;;; Set up handlers
2083 023402  010700                    mov     pc,r0                   ;;; ...
2084 023404  062700  000404            add     #<s$$udc-.>,r0          ;;; Use our own udc handler
2085 023410  010037  100002            mov     r0,@#i$udc              ;;; ...
2086 023414  010700                    mov     pc,r0                   ;;; ...
2087 023416  062700  000432            add     #<s$$rti-.>,r0          ;;;
2088 023422  010037  100006            mov     r0,@#i$clk              ;;; Make clock interrupt rti
2089 023426  010037  100016            mov     r0,@#i$sec              ;;; Make sector interrupt rti
2090 023432  106427  000000            mtps    #ps0                    ;;; Make it good
2091
2092                           ; Go Size the Drives
2093
2094 023436  010246                    mov     r2,-(sp)                ; Save Registers
2095 023440  010346                    mov     r3,-(sp)                ; ...
2096 023442  010702                    mov     pc,r2                   ; Point to Unit Descriptor Table
2097 023444  062702  000500            add     #<msgdat+2>-.,r2        ; ...
2098
2099 023450  010200                    mov     r2,r0                   ; ...
2100 023452  012703  000004            mov     #4.,r3                  ; Initialize all Unit Descriptors
2101 023456                    siznon:                                 ; ...
2102 023456  012720  177777            mov     #-1.,(r0)+              ; ... to "Not a Winnie"
2103 023462  077303                    sob     r3,siznon               ; ...
2104
2105 023464  005003                    clr     r3                      ; Set Drive Count
2106
2107 023466                    sizlop::                                ; ** Loop Until We Get All of Them **
2108 023466  010300                    mov     r3,r0                   ; Compute the right Winnie channel
2109 023470  042700  177776            bic     #^c<bit0>,r0            ; ... for the pllctl csr
2110 023474  006300                    asl     r0                      ; ...
2111 023476  062700  000010            add     #bit3,r0                ; ...
2112 023502  010037  140002            mov     r0,@#rw$pll             ; ...
2113 023506  012737  000104  140022    mov     #op.srp+4,@#w$cmd       ; Set up UDC registers
2114 023514  005037  140020            clr     @#w$dat                 ; ...
2115 023520  005037  140020            clr     @#w$dat                 ; ...
2116 023524  012737  000110  140022    mov     #op.srp+8.,@#w$cmd      ; ...
2117 023532  012737  000300  140020    mov     #rd.mode,@#w$dat        ; ...
2118 023540  010300                    mov     r3,r0                   ; Select the Drive
2119 023542  062700  000044            add     #op.sd.rd,r0            ; ...
2120 023546  004767  000300            jsr     pc,doudc                ; ...
2121 023552  005700                    tst     r0                      ; Okay?
2122 023554  001055                    bne     sizend                  ; Nope, something is screwed up
2123
2124 023556  032737  140000  140006    bit     #bit14+bit15,@#r$fps    ; Winnie?
2125 023564  001445                    beq     sizdrv                  ; If not, skip to next drive
2126
2127 023566  005012                    clr     (r2)                    ; It's a Winnie - Set Count to 0
2128 023570  012700  000003            mov     #op.rd,r0               ; Restore Drive
2129 023574  004767  000252            jsr     pc,doudc                ; Do UDC Command
2130 023600  005700                    tst     r0                      ; Okay?
2131 023602  001042                    bne     sizend                  ; Nope, something is screwed up
2132
```

```
2133 023604  012700  000003                mov     #ersek0,r0              ; Assume that seek to 0 failed
2134 023610  012737  000111  140022         mov     #op.srp+9.,@#w$cmd      ; At Cylinder 0?
2135 023616  013701  140010                mov     @#r$dat,r1              ; ...
2136 023622  032701  000020                bit     #bit4,r1                ; ...
2137 023626  001430                        beq     sizend                  ; Nope, something's wrong
2138
2139 023630                        sizin:                                  ; ** Step In Until Track 0 Found **
2140 023630  005212                        inc     (r2)                    ; Up Cylinder Count
2141 023632  012700  000005                mov     #op.si1,r0              ; Step In One Cylinder
2142 023636  004767  000210                jsr     pc,doudc                ; Do UDC Command
2143 023642  005700                        tst     r0                      ; Okay?
2144 023644  001021                        bne     sizend                  ; Nope, something is screwed up
2145
2146 023646  012737  000111  140022         mov     #op.srp+9.,@#w$cmd      ; At Cylinder 0?
2147 023654  013701  140010                mov     @#r$dat,r1              ; If so,
2148 023660  032701  000020                bit     #bit4,r1                ; ... skip to bump up
2149 023664  001003                        bne     sizrd                   ; ... descriptors
2150
2151 023666  021227  004000                cmp     (r2),#2048.             ; SMC Cylinder Limit Reached?
2152 023672  002756                        blt     sizin                   ; ** Bottom of Step In Loop **
2153
2154 023674                        sizrd:                                  ; ** This was a Winnie **
2155 023674  062702  000002                add     #untdsz,r2              ; Bump Pointer to Next Unit Descriptor
2156
2157 023700                        sizdrv:                                 ; ** Check Next Drive **
2158 023700  005203                        inc     r3                      ; Up Drive Count
2159 023702  020327  000004                cmp     r3,#4.                  ; All 4 Drives Checked?
2160 023706  002667                        blt     sizlop                  ; ** Bottom of Loop **
2161
2162 023710                        sizend:                                 ; ** Send Status and Table **
2163 023710  010067  000226                mov     r0,msgdat               ; Save status
2164 023714  012700  000001                mov     #op.dd,r0               ; Deselect Drive
2165 023720  004767  000126                jsr     pc,doudc                ; ...
2166 023724  012603                        mov     (sp)+,r3                ; Pop
2167 023726  012602                        mov     (sp)+,r2                ; ...
2168 023730  106427  000340                mtps    #ps7                    ;;; Put the MSCP Handlers Back
2169 023734  012637  100016                mov     (sp)+,@#i$sec           ;;; ...
2170 023740  012637  100006                mov     (sp)+,@#i$clk           ;;; ...
2171 023744  012637  100002                mov     (sp)+,@#i$udc           ;;; ...
2172 023750  106427  000000                mtps    #ps0                    ; ...
2173
2174 023754                      sizexi::                                  ; ** Okay, talk to the Host **
2175
2176                             ;PutData,msg,msglen - Send Response to Host
2177
2178 023754  010700                        mov     pc,r0                   ;figure the relative address
2179 023756  062700  000162                add     #msg-.,r0               ;... of the buffer
2180 023762  012746  000014                mov     #msglen,-(sp)           ;load lenght in bytes of the buffer
2181 023766  010046                        mov     r0,-(sp)                ;load relative address of the buffer
2182 023770  013746  000146                mov     @#146,-(sp)             ;load location of routine in microcode
2183 023774  004736                        jsr     pc,@(sp)+               ;call Put Data routine in Ucode
2184 023776  022626                        cmp     (sp)+,(sp)+             ;fix stack
2185
2186                             ; Terminate Supplied Program
2187
2188 024000  013700  000142                mov     @#142,r0                ;load location of routine in microcode
2189 024004  004710                        jsr     pc,(r0)                 ;call Terminate routine in Ucode
```

SEQ 0049

```
Autosz

   2190 024006  000207                          rts     pc                      ; .
```

Autosz

```
 2192                                   ;+
 2193                                   ; UDC Interrupt Handler
 2194                                   ;
 2195                                   ; Taken from RQDX3.MAC m$udc code:
 2196                                   ;-
 2197
 2198 024010                            s$$udc::                                ;;; UDC Handler
 2199 024010  005767  000120                    tst     s$$bug                  ;;; is the ECO wire there?
 2200 024014  001404                            beq     s$$udi                  ;;; nope
 2201 024016  032737  020000  140006            bit     #20000,@#r$fps          ;;; is the 9224 interrupt line set?
 2202 024024  001011                            bne     s$$rti                  ;;; if not, must be a bogus interrupt
 2203
 2204 024026                            s$$udi:                                 ;;; ...
 2205 024026  113746  140012                    movb    @#r$cmd,-(sp)           ;;; get interrupt status
 2206 024032  142716  000035                    bicb    #35,(sp)                ;;; clear bits of no interest
 2207 024036  122726  000240                    cmpb    #240,(sp)+              ;;; valid status?
 2208 024042  001002                            bne     s$$rti                  ;;; no, it's a bogus interrupt
 2209 024044  005267  000066                    inc     s$$flag                 ;;; set the flag
 2210
 2211                                   ;+
 2212                                   ; Return from Interrupt
 2213                                   ;-
 2214
 2215 024050                            s$$rti::                                        ;;; ...
 2216 024050  000002                            rti                             ;;; just quit
 2217
 2218                                   ;+
 2219                                   ; DOUDC - Do a UDC Command
 2220                                   ;
 2221                                   ; This routine sends a commands and waits an interrupt or
 2222                                   ; until timer expires.
 2223                                   ;
 2224                                   ; Input:        r0      = command
 2225                                   ; Output:       r0      = 0 for success, non zero for failure
 2226                                   ;               ... r0,r1 not preserved.
 2227                                   ;-
 2228
 2229          007570                   mseca = 30.*132.                        ; Max Step Rate * some *
 2230                                                                           ;  loop for 7.5 MHz T11 clock
 2231
 2232 024052                            doudc::                                 ; ** Do a UDC command **
 2233 024052  005067  000060                    clr     s$$flag                 ; Clear udc flag (interrupt pending)
 2234 024056  010037  140022                    mov     r0,@#w$cmd              ; Send the command
 2235 024062  012700  004000                    mov     #2048.,r0               ; Set the rom timer (max cylinders)
 2236
 2237 024066                            mswait:                                 ; ** Wait **
 2238 024066  012701  007570                    mov     #mseca,r1               ; set one millisecond counter
 2239 024072                            msin:                                   ; ** Top of Inner Loop **
 2240 024072  005767  000040                    tst     s$$flag                 ; 3.60 udc interrupted
 2241 024076  001005                            bne     msend                   ; 1.60 out if udc interrupted
 2242 024100  077104                            sob     r1,msin                 ; 2.40  Total: 7.60 @7.5MHz=>
 2243                                                                           ;                8.5457 @6.67MHz
 2244 024102  077007                            sob     r0,mswait               ; ** Bottom of Outer Loop **
 2245 024104  012700  000002                    mov     #eruint,r0              ; Never Interrupted
 2246 024110  000410                            br      douret                  ; ...
 2247
 2248 024112                            msend:                                  ; ** Interrupt Happened **
```

Autosz

```
 2249 024112  012700  000001                mov     #erudon,r0              ; Assume Never Done
 2250 024116  013701  140012                mov     @#r$cmd,r1              ; Get the return status
 2251 024122  032701  000040                bit     #bit5,r1                ; All done yet?
 2252 024126  001401                        beq     douret                  ; If so, pop out of th s
 2253
 2254 024130  005000                        clr     r0                      ; Assume everything's ok
 2255
 2256 024132                        douret:                                 ; ** Return **
 2257 024132  000207                        rts     pc                      ; Back to caller
```

SIZER Supplied Program Data

```
                                        .sbttl  SIZER Supplied Program Data
                                ;       .psect  c$data

                                ; Special Stuff

024134                          s$$bug: .blkw   1                       ; ECO Wire
024136                          s$$flag:.blkw   1                       ; UDC flag

                                ; Packet Area

024140      012     140         msg::   .byte   10.,.b.spl              ; Final Message
024142                          msgdat: .blkw   5.                      ; Status and Unit Descriptor Table
            000014              msglen  = .-msg                         ; Message Length (Byte Count)
            000002              untdsz  = 2.                            ; Unit Descriptor Length


                                .enable AMA
024154                          AUTOend:
                                ;*************************************************************************
                                ;
                                ;       This routine builds the UIT table or get the UIT table
                                ; depending who the questions are answered to the manual questions.
                                ; If the unit is a listed or regconizable drive we will use a prebuilt
                                ; UIT table. If not we will have to ask all the questions to build
                                ; a table.
                                ;
                                ;*************************************************************************
024154                          BLDUIT:
024154  032737  100000  002320          bit     #bit15,untflgs
024162  001402                        beq     manbld
024164  000137  024502                jmp     autobld

024170                          manbld: printf  #DrvTxa                 ;print out UIT tables and their related drives
024210                                  printf  #DrvTxb                 ;UIN    Drive
024230                                  printf  #DrvTx0                 ;0      rd51
024250                                  printf  #DrvTx1                 ;1      rd52
024270                                  printf  #DrvTx2                 ;2      etc
024310                                  printf  #DrvTx3                 ;3      etc
024330                                  printf  #DrvTx4                 ;4
024350                                  printf  #DrvTx5
024370                                  printf  #DrvTx6
024410                                  printf  #DrvTx7
024430                                  printf  #DrvTxc

024450                          GMANID unt.nbr,UIN,0,17,0,10,no         ;GET Unit identifier number (0-7)
                                                                        ;PLACE IN bits 0-3.
                                                                        ;no defaults person must know what Unit Identificati
on number.
024470  022737  000010  002326          cmp     #10,uin
024476  001514                        beq     tblbld                  ;ask twenty questions to build table
024500  000477                        br      uitloc                  ;get correct table address into UITadrs

024502                          autobld:
024502  013700  002312                mov     unit,r0                 ;get unit number
024506  006300                        asl     r0                      ;get the byte offset of tbl
024510  012737  000000  002326  1$:   mov     #0,uin                  ;pick UIT number 0
024516  023760  003100  024144          cmp     UIT0+UITsiz-2,msg+4(r0) ;if cylinder # equals UIT table # this is the correc
t UIT table
024524  001465                        beq     2$
```

SIZER Supplied Program Data

```
   2316 024526  012737  000001  002326          mov     #1,uin                  ;pick UIT number 1
   2317 024534  023760  003202  024144          cmp     UIT1-UITsiz-2,msg+4(r0) ;if cylinder # equals UIT table # this is the correc
t UIT table
   2318 024542  001456                          beq     2$
   2319 024544  012737  000002  002326          mov     #2,uin                  ;pick UIT number 2
   2320 024552  023760  003304  024144          cmp     UIT2-UITsiz-2,msg+4(r0) ;if cylinder # equals UIT table # this is the correc
t UIT table
   2321 024560  001447                          beq     2$
   2322 024562  012737  000003  002326          mov     #3,uin                  ;pick UIT number 3
   2323 024570  023760  003406  024144          cmp     UIT3-UITsiz-2,msg+4(r0) ;if cylinder # equals UIT table # this is the correc
t UIT table
   2324 024576  001440                          beq     2$
   2325 024600  012737  000004  002326          mov     #4,uin                  ;pick UIT number 4
   2326 024606  023760  003510  024144          cmp     UIT4-UITsiz-2,msg+4(r0) ;if cylinder # equals UIT table # this is the correc
t UIT table
   2327 024614  001431                          beq     2$
   2328 024616  012737  000005  002326          mov     #5,uin                  ;pick UIT number 5
   2329 024624  023760  003612  024144          cmp     UIT5-UITsiz-2,msg+4(r0) ;if cylinder # equals UIT table # this is the correc
t UIT table
   2330 024632  001422                          beq     2$
   2331 024634  012737  000006  002326          mov     #6,uin                  ;pick UIT number 6
   2332 024642  023760  003714  024144          cmp     UIT6-UITsiz-2,msg+4(r0) ;if cylinder # equals UIT table # this is the correc
t UIT table
   2333 024650  001413                          beq     2$
   2334 024652  012737  000007  002326          mov     #7,uin                  ;pick UIT number 7
   2335 024660  023760  004016  024144          cmp     UIT7-UITsiz-2,msg+4(r0) ;if cylinder # equals UIT table # this 's the correc
t UIT table
   2336 024666  001404                          beq     2$
   2337 024670  012737  000010  002326          mov     #10,uin                 ;if no UIT present then bu'ld a UIT 'n default table
 area
   2338 024676  000414                          br      tblbld                  ;go built a UIT 'n UIT default table
   2339 024700                          2$:
   2340 024700                          uitloc:
   2341 024700  013702  002326                  mov     UIN,r2                  ;get the correct UIT table address 'nto UITadr reg's
ter
   2342 024704  012703  003000                  mov     #UIT0,r3                ;r3 contains base address of UIT tables
   2343 024710  001403                          beq     11$                     ;if UIN=0 then set table to UIT0
   2344 024712  062703  000102          10$:    add     #UITsiz,r3              ;else multiply UIT size by the UIN number and add to
 base address
   2345 024716  077203                          sob     r2,10$
   2346 024720  010337  002304          11$:    mov     r3,UITadr               ;store the proper address of the UIT table
   2347 024724  000137  025716                  jmp     cont                    ;all done
   2348
   2349 024730                          tblbld:                                 ;We must build a UNIT INFORMATION TABLE
   2350 024730  012737  004020  002304          mov     #UITdf,UITadr           ;move the address of UIT table into a reg'ster
   2351 024736                          GMANID TBQ8,UITdf+20,D,-1,0, 1,yes              ;
   2352 024756                          GMANID TBQ9,UITdf+22,D,-1,0,-1,yes              ;
   2353 024776                          GMANID TBQ10,UITdf+24,D,-1,0,-1,yes             ;
   2354
   2355                                 ;GMANID TBQ0,UITdf+0,D, 1,0,-1,yes              ;XBN
   2356                                 ;GMANID TBQ1,UITdf+2,D, 1,0, 1,yes              ;
   2357 025016  013700  004040                  mov     UITdf+20,r0             ;/XBN size (lo wrd) XBN s'ze = 3+(1+sectors_
per_track)/
   2358 025022  005001                          clr     r1
   2359 025024  005200                          inc     r0
   2360 025026  060001                          add     r0,r1
   2361 025030  060001                          add     r0,r1
   2362 025032  060001                          add     r0,r1
   2363 025034  010137  004020                  mov     r1,UITdf+0
   2364 025040  005037  004022                  clr     UITdf+2
   2365
   2366                                 ;GMANID TBQ2,UITdf+4,D,-1,0,-1,yes              ;DBN
   2367                                 ;GMANID TBQ3,UITdf+6,D,-1,0, 1,yes              ;
   2368 025044                          askdbn:
   2369 025044                          GMANID ASK,DBN,DBN,A, 1,0.,10.,yes      ;ask for the User what local program he wants to run
   2370 025064  012701  002702                  mov     #dbn,r1                 ;address of ascii decimal data
```

```
                        jsr     pc,ASCDEC               ;call conversion routine
```



```
                        nop
                ;GMANID TBQ4,UITdf+10,D,-1,0,-1,yes             ;LBN
                ;GMANID TBQ5,UITdf+12,D,-1,0,-1,yes             ;
                asklbn:
                GMANID ASK,LBN,LBN,A,-1,0.,10.,yes      ;ask for the User what local program he wants to run
                        mov     #lbn,r1                 ;address of ascii decimal data
                        mov     #uitdf+10,r0            ;address to store octal conversion
                        jsr     pc,ASCDEC               ;call conversion routine
                        nop
                ;GMANID TBQ6,UITdf+14,D,-1,0,-1,yes             ;RBN
                ;GMANID TBQ7,UITdf+16,D,-1,0,-1,yes             ;
                askrbn:
                GMANID ASK,RBN,RBN,A,-1,0.,10.,yes      ;ask for the User what local program he wants to run
                        mov     #rbn,r1                 ;address of ascii decimal data
                        mov     #uitdf+14,r0            ;address to store octal conversion
                        jsr     pc,ASCDEC               ;call conversion routine
                        nop
                GMANID TBQ11,UITdf+26,D,-1,0,-1,yes             ;PRECOMP CYLINDER
                GMANID TBQ12,UITdf+30,D,-1,0,-1,yes             ;
                GMANID TBQ13,UITdf+32,D,-1,0,-1,yes             ;
                GMANID TBQ14,UITdf+34,D,-1,0,-1,yes             ;
                GMANID TBQ15,UITdf+36,D,-1,0,-1,yes             ;
                GMANID TBQ16,UITdf+40,D,-1,0,-1,yes             ;
                GMANID TBQ17,UITdf+42,D,-1,0,-1,yes             ;
                GMANID TBQ18,UITdf+44,D,-1,0,-1,yes             ;
                GMANID TBQ19,UITdf+46,D,-1,0, 1,yes             ;
                GMANID TBQ20,UITdf+50,D,-1,0,-1,yes             ;
                GMANID TBQ21,UITdf+52,D,-1,0,-1,yes             ;
                GMANID TBQ22,UITdf+54,D,-1,0,-1,yes             ;
                GMANID TBQ23,UITdf+56,D,-1,0,-1,yes             ;
                GMANID TBQ24,UITdf+60,D,-1,0,-1,yes             ;
                GMANID TBQ25,UITdf+62,D,-1,0,-1,yes             ;
                GMANID TBQ26,UITdf+64,D,-1,0,-1,yes             ;
                ;tbq27 purposely left out
                GMANID TBQ28,UITdf+66,D,-1,0,-1,yes             ;
                GMANID TBQ29,UITdf+70,D,-1,0,-1,yes             ;
                GMANID TBQ30,UITdf+72,D,-1,0,-1,yes             ;
                GMANID TBQ31,UITdf+74,D,-1,0,-1,yes             ;
                GMANID TBQ32,UITdf+76,D,-1,0,-1,yes             ;

                cont:   rts     pc                      ;go back
                ;*****************************************************************
                ;
                ;       Octal number to ASCII Decimal number
                ;       r1 = address of ascii decimal data
                ;       r0 = octal data word
                ;*****************************************************************
                OCTASC:
                        mov     r2, (sp)
                        mov     r3, (sp)
                        clr     r2                      ;clear the decimal table pointer
                1$:     clr     r3                      ;clear decimal d'g't
                2$:     inc     r3                      ;increment decimal digit
                        sub     dectbl(r2),r0           ;subtract a power of ten from accumulator
                        bge     2$                      ;'f not negative subtract another
                        add     dectbl(r2),r0           ;adjust accumulator so positive
                        dec     r3                      ;adjust decimal digit
```

```
2430 025746  062703  000060                add     #60,r3                  ;convert decimal to ascii
2431 025752  110321                        movb    r3,(r1)+                ;mov ascii digit text into buffer
2432 025754  005722                        tst     (r2)+                   ;increment table pointer
2433 025756  005762  025772                tst     dectbl(r2)              ;check if thats all
2434 025762  001361                        bne     1$
2435 025764  012603                        mov     (sp)+,r3
2436 025766  012602                        mov     (sp)+,r2
2437 025770  000207                        rts     pc
2438 025772                        dectbl:
2439 025772  023420                        .word   10000.
2440 025774  001750                        .word   1000.
2441 025776  000144                        .word   100.
2442 026000  000012                        .word   10.
2443 026002  000001                        .word   1.
2444 026004  000000                        .word   0
2445                               ;*****************************************************************
2446                               ;
2447                               ;       ASCII DECIMAL numbers to  Octal numbers
2448                               ;       r1 = address of ascii decimal data
2449                               ;       r0 = address to store octal data low word, h'gh word
2450                               ;*****************************************************************
2451 026006                        ASCDEC:
2452 026006  010546                        mov     r5,-(sp)
2453 026010  010446                        mov     r4,-(sp)
2454 026012  010346                        mov     r3,-(sp)
2455 026014  010246                        mov     r2,-(sp)
2456 026016  005004                        clr     r4
2457 026020  005003                        clr     r3
2458 026022  005002                        clr     r2
2459 026024  112104                3$:     movb    (r1)+,r4
2460 026026  001423                        beq     1$                      ;if digit equals null than all done
2461                               ;       cmp     r4,#60                  ;check for a real number value
2462                               ;       blt     asklbn                  ;wasn't a real number
2463                               ;       cmp     r4,#71
2464                               ;       bgt     asklbn                  ;wasn't a real number
2465
2466 026030  162704  000060                sub     #60,r4
2467 026034  010346                        mov     r3,-(sp)
2468 026036  010246                        mov     r2,-(sp)                ;save accum
2469
2470 026040  012705  000003                mov     #3,r5                   ;accum * 8
2471 026044  006302                4$:     asl     r2
2472 026046  006103                        rol     r3
2473 026050  077503                        sob     r5,4$
2474
2475 026052  006316                        asl     (sp)                    ;accum*2
2476 026054  006166  000002                rol     2(sp)
2477
2478 026060  000241                        clc                             ; accum*8 + accum*2
2479 026062  062602                        add     (sp)+,r2
2480 026064  005503                        adc     r3
2481 026066  062603                        add     (sp)+,r3
2482
2483 026070  060402                        add     r4,r2                   ;add presert digit to accum*10
2484 026072  005503                        adc     r3
2485 026074  000753                        br      3$
2486
```

```
 2487 026076  010220                 1$:     mov     r2,(r0)+                ;load lo number
 2488 026100  010310                         mov     r3,(r0)                 ;load hi number
 2489
 2490 026102  012602                         mov     (sp)+,r2                ;restore stack to 'ts orginal
 2491 026104  012603                         mov     (sp)+,r3
 2492 026106  012604                         mov     (sp)+,r4
 2493 026110  012605                         mov     (sp)+,r5
 2494 026112  000207                         rts     pc
 2495
 2496                                 ;*****************************************************************
 2497                                 ;
 2498                                 ; This routine types out the ASCII information passed
 2499                                 ; by the disk controller. This ASCII information is
 2500                                 ; contained in the buffer called DATARE and is offset
 2501                                 ; by 1 word. To fake the DRS macro routine a "%A" is
 2502                                 ; placed in front of the text.
 2503                                 ;*****************************************************************
 2504
 2505 026114                         typDUPbuf:
 2506 026114  012701  002534                 mov     #datare,r1      ;get data area address of ascii info
 2507 026120  063701  002350                 add     rsppak+14,r1    ;add the number of byte transfered
 2508 026124  105021                 1$:     clrb    (r1)+           ;put null characters into data buffer after end of ASCII inf
o
 2509 026126  020127  002660                 cmp     r1,#prgnam      ;
 2510 026132  001374                         bne     1$              ;we do this to fake out the DRS macro
 2511
 2512 026134  112737  000045  002534         movb    #45,datare      ;put the "%" delimiter for the DRS macro
 2513 026142  112737  000101  002535         movb    #101,datare+1   ;put the "A" for ascii info for the DRS macro
 2514 026150                                 printx  #PB13           ;New Line <cr><lf>
 2515 026170                                 printx  #datare         ;print the message returned from the controller
 2516
 2517 026210                         clrDUPbuf:
 2518 026210  012701  002534                 mov     #datare,r1      ;clear out entire data area
 2519 026214  105021                 2$:     clrb    (r1)+           ;
 2520 026216  020127  002660                 cmp     r1,#prgnam      ;
 2521 026222  001374                         bne     2$              ;
 2522 026224  000207                         rts     pc
 2523                                 ;*****************************************************************
 2524                                 ;
 2525                                 ;       THIS ROUTINE IS TO CHECK ON THE RESPONSE PACKET
 2526                                 ;       GOODNESS. THE COMMAND REFERENCE NUMBER, THE END CODE
 2527                                 ;       AND THE STATUS ARE TESTED.
 2528                                 ;*****************************************************************
 2529
 2530 026226                         RSPCHK:
 2531
 2532 026226  013701  002434             mov         cmdpak,r1
 2533 026232  013700  002334             mov         rsppak,r0
 2534 026236  020001                     cmp         r0,r1                   ;compare CRN numbers
 2535 026240  001014                     bne         1$
 2536 026242  013701  002444             mov         cmdpak+10,r1
 2537 026246  062701  000200             add         #200,r1
 2538 026252  013700  002344             mov         rsppak+10,r0
 2539 026256  020001                     cmp         r0,r1                   ;compare Opcodes
 2540 026260  001004                     bne         1$
 2541 026262  013701  002346             mov         rsppak+12,r1            ;check the status
 2542 026266  001001                     bne         1$
 2543 026270  000207                     rts         pc                      ;if all checks then return
```

SIZER Supplied Program Data

```
2544
2545
2546 026272                          1$:     ERRDF   10.df11                 ;if all doesn't check then a bad packet
2547 026302                          PRNTpkt:                                ;Bad response packet
2548 026302                                  Printb  @PB11crn,cmdpak,rsppak  ;Expected CRN XXXX ,Received CRN YYYY
2549 026332  013701  002344                 mov     rsppak+10,r1            ;check response opcode reply
2550 026336  032701  000200                 bit     #200,r1                 ;see if a end command response was send
2551 026342  001010                         bne     2$
2552 026344                                  printx  @PB11end                ;No end bit in response packet endcode
2553 026364  022701  000201          2$:     cmp     #201,r1
2554 026370  001010                         bne     3$                      ;check if Get Dust Status command
2555 026372                                  printx  @PB11GDS
2556 026412  022701  000202          3$:     cmp     #202,r1
2557 026416  001010                         bne     4$                      ;check if Execute Supplied Program
2558 026420                                  printx  @PB11ESP
2559 026440  022701  000203          4$:     cmp     #203,r1
2560 026444  001010                         bne     5$                      ;check if Execute Local Program
2561 026446                                  printx  @PB11ELP
2562 026466  022701  000204          5$:     cmp     #204,r1
2563 026472  001010                         bne     6$                      ;check if Send Data
2564 026474                                  printx  @PB11SD
2565 026514  022701  000205          6$:     cmp     #205,r1
2566 026520  001022                         bne     7$                      ;check if Receive Data
2567 026522                                  printx  @PB11RD
2568 026542                                  Printb  @PBSF0,r3,r5    ;"type xxx, message number xxxxx is unknow to this program"
2569 026566  022701  000206          7$:     cmp     #206,r1
2570 026572  001010                         bne     8$                      ;check if Abort Program
2571 026574                                  printx  @PB11AP
2572 026614                          8$:     Printb  @PB11op,cmdpak+10,rsppak+10
2573                                                                         ;CMDpkt opcode XXXX,RSPpkt opcode YYYYY
2574
2575 026644  013701  002346                 mov     rsppak+12,r1            ;find out what kind of status we have
2576 026650  022701  000000                 cmp     #0.,r1
2577 026654  001010                         bne     10$
2578 026656                                  printx  @pb11s0                 ;status:  successful
2579 026676  022701  000001          10$:    cmp     #1.,r1
2580 026702  001010                         bne     11$
2581 026704                                  printx  @pb11s1                 ;status:  Invalid Command
2582 026724  022701  000002          11$:    cmp     #2.,r1
2583 026730  001010                         bne     12$
2584 026732                                  printx  @pb11s2                 ;status:  No Region Available
2585 026752  022701  000003          12$:    cmp     #3.,r1
2586 026756  001010                         bne     13$
2587 026760                                  printx  @pb11s3                 ;status:  No Region Suitable
2588 027000  022701  000004          13$:    cmp     #4.,r1
2589 027004  001010                         bne     14$
2590 027006                                  printx  @pb11s4                 ;status:  Program Not Known
2591 027026  022701  000005          14$:    cmp     #5.,r1
2592 027032  001010                         bne     15$
2593 027034                                  printx  @pb11s5                 ;status:  Load Failure
2594 027054  022701  000006          15$:    cmp     #6.,r1
2595 027060  001010                         bne     16$
2596 027062                                  printx  @pb11s6                 ;status:  Standalone
2597 027102  022701  000011          16$:    cmp     #9.,r1
2598 027106  001010                         bne     19$
2599 027110                                  printx  @pb11s9                 ;status:  Host Buffer Access error
2600 027130                          19$:
```

SIZER Supplied Program Data

```
2601 027130                                     Printb  @PB11sts.rsppak+12      ;Response packet status XXXX
2602 027154  000137  034572                     jmp     dropunt                 ;drop unit and go on
2603
2604
2605                                    ;*************************************************************************
2606                                    ;
2607                                    ;                       BIT FIFTEEN TEST
2608                                    ;*************************************************************************
2609 027160                             BIT15T:
2610 027160  032714  100000                     bit     #bit15,(r4)
2611 027164  001001                             bne     100$
2612 027166  000207                             rts     pc
2613 027170                             100$:   ERRDF   9,df12          ;Fatal SA error
2614 027200  011401                             mov     (r4),r1
2615 027202  022701  001000                     cmp     #1000,r1
2616 027206  001010                             bne     1$
2617 027210                                     printx  #pb1201         ;
2618 027230  022701  100001             1$:     cmp     #100001,r1
2619 027234  001010                             bne     2$
2620 027236                                     printx  #pb1202         ;
2621 027256  022701  100002             2$:     cmp     #100002,r1
2622 027262  001010                             bne     3$
2623 027264                                     printx  #pb1203         ;
2624 027304  022701  100003             3$:     cmp     #100003,r1
2625 027310  001010                             bne     4$
2626 027312                                     printx  #pb1204         ;
2627 027332  022701  100004             4$:     cmp     #100004,r1
2628 027336  001010                             bne     5$
2629 027340                                     printx  #pb1205         ;
2630 027360  022701  100005             5$:     cmp     #100005,r1
2631 027364  001010                             bne     6$
2632 027366                                     printx  #pb1206         ;
2633 027406  022701  100006             6$:     cmp     #100006,r1
2634 027412  001010                             bne     7$
2635 027414                                     printx  #pb1207         ;
2636 027434  022701  100007             7$:     cmp     #100007,r1
2637 027440  001010                             bne     8$
2638 027442                                     printx  #pb1208         ;
2639 027462  022701  100010             8$:     cmp     #100010,r1
2640 027466  001010                             bne     9$
2641 027470                                     printx  #pb1209         ;
2642 027510  022701  100011             9$:     cmp     #100011,r1
2643 027514  001010                             bne     10$
2644 027516                                     printx  #pb1210         ;
2645 027536  022701  100012             10$:    cmp     #100012,r1
2646 027542  001010                             bne     11$
2647 027544                                     printx  #pb1211         ;
2648 027564  022701  100013             11$:    cmp     #100013,r1
2649 027570  001010                             bne     12$
2650 027572                                     printx  #pb1212         ;
2651 027612  022701  100014             12$:    cmp     #100014,r1
2652 027616  001010                             bne     13$
2653 027620                                     printx  #pb1213         ;
2654 027640  022701  100015             13$:    cmp     #100015,r1
2655 027644  001010                             bne     14$
2656 027646                                     printx  #pb1214         ;
2657 027666  022701  100016             14$:    cmp     #100016,r1
```

SIZER Supplied Program Data

```
2658 027672  001010                        bne     15$
2659 027674                                printx  #pb1215         ;
2660 027714  022701  100017          15$:   cmp     #100017,r1
2661 027720  001010                        bne     16$
2662 027722                                printx  #pb1216         ;
2663 027742  022701  100020          16$:   cmp     #100020,r1
2664 027746  001010                        bne     17$
2665 027750                                printx  #pb1217         ;
2666 027770  022701  100021          17$:   cmp     #100021,r1
2667 027774  001010                        bne     18$
2668 027776                                printx  #pb1218         ;
2669 030016  022701  100022          18$:   cmp     #100022,r1
2670 030022  001010                        bne     19$
2671 030024                                printx  #pb1219         ;
2672 030044  022701  100023          19$:   cmp     #100023,r1
2673 030050  001010                        bne     20$
2674 030052                                printx  #pb1220         ;
2675 030072  022701  100024          20$:   cmp     #100024,r1
2676 030076  001010                        bne     21$
2677 030100                                printx  #pb1221         ;
2678 030120  022701  100025          21$:   cmp     #100025,r1
2679 030124  001010                        bne     22$
2680 030126                                printx  #pb1222         ;
2681 030146  022701  100026          22$:   cmp     #100026,r1
2682 030152  001010                        bne     23$
2683 030154                                printx  #pb1223         ;
2684 030174                          23$:
2685 030174                                printb  #pb12,r1        ;SA value: xxxxx
2686 030216  000137  034572                jmp     dropunt                 ;drop unit and go on
2687
2688                                 ;*************************************************************************
2689                                 ;       Unexpected Interrupt Server
2690                                 ;
2691                                 ;*************************************************************************
2692 030222                          intsrv:
2693
2694 030222                                ERRSF   8,sf100 ;Fatal SA error
2695 030232                                docln                   ;do clean up and quit
2696 030234  000137  034572                jmp     dropunt         ;drop test unit and end pass
2697
2698
```

SEQ 0060

SIZER Supplied Program Data

```
 2700 030240                                BGNPROT
 2701 030240  177777                        .WORD -1
 2702 030242  177777                        .WORD -1
 2703 030244  177777                        .WORD -1
 2704 030246                                ENDPROT
 2705
 2706 030246                                BGNINIT                         ;SEQUENTIAL EXAMPLE
 2707 030246                                READEF      @EF.CONTINUE        ;Continue COMMAND?
 2708 030254                                BCOMPLETE   conton              ;YES, GET NO P-TABLE but still initialize
 2709 030256                                READEF      @EF.NEW             ;NEW PASS
 2710 030264                                BNCOMPLETE  next                ;if not new then go to next unit number
 2711 030266  012737  177777  002274  SETUP: mov       #-1,LOGUNIT         ;INITIALIZE LOGICAL UNIT NBR
 2712 030274  005237  002274        NEXT:   inc       LOGUNIT                     ;POINT TO NEXT LOGICAL UNIT
 2713 030300  023737  002274  002012        cmp       LOGUNIT,L$UNIT      ;HAVE WE PASSED MAXIMUM?
 2714 030306  001002                        bne       1$
 2715 030310  000137  030520                jmp       ABORT               ;YES, ABORT THE PASS
 2716 030314                          1$:   GPHARD LOGUNIT,PLOC             ;GET THE P-TABLE
 2717 030326                                BNCOMPLETE NEXT                 ;if not available get next unit
 2718
 2719 030330  013700  002300                mov       ploc,r0
 2720 030334  010037  002302                mov       r0,ptbl                     ;store the Ptable address for unit
 2721 030340  012037  002306                mov       (r0)+,ipreg                 ;store IPreg address into register
 2722 030344  012037  002310                mov       (r0)+,vector                ;store vector
 2723 030350  012037  002312                mov       (r0)+,unit                  ;store logical drive number
 2724 030354  012037  002316                mov       (r0)+,sernbr                ;store the serial number
 2725 030360  012037  002320                mov       (r0)+,untflgs
 2726
 2727 030364  005037  002522          conton: clr     LSTCRN                      ;basic initialization stuff
 2728 030370  005037  002526                clr       LSTVCT
 2729 030374  005037  002530                clr       LOPRGI
 2730 030400  005037  002532                clr       HIPRGI
 2731
 2732 030404  032737  100000  002320        bit       @bit15,untflgs
 2733 030412  001411                        beq       1$                          ;
 2734 030414  032737  040000  002320        bit       @bit14,untflgs
 2735 030422  001005                        bne       1$
 2736 030424                                dodu      logunit                     ;if in auto mode and warning flag isn't acknowledge
drop unit
 2737 030432  000137  030520                jmp       abort
 2738
 2739 030436  013746  000004          1$:   mov       @#4,-(sp)                   ;test to see if controller is there
 2740 030442  012737  030456  000004        mov       #$2,@#4
 2741 030450  005077  151632                clr       @IPreg                      ;get controller into know state
 2742 030454  000410                        br        $3
 2743
 2744 030456                          $2:   ERRDF     7,DF4                       ;NXM trap at controller IP address
 2745 030466                                dodu      LOGUNIT                     ;drop unit
 2746 030474  000677                        br        next                        ;get new unit
 2747
 2748 030476  012637  000004          $3:   mov       (sp)+,@#4                   ;move value back intc location 4
 2749
 2750 030502  012700  000076                mov       #76,r0              ;clean out all packets and interrupt flags
 2751 030506  012701  002330                mov       #rsp1,r1            ;and the command area
 2752 030512  005021                  $4:   clr       (r1)+
 2753 030514  077002                        sob       r0,$4
 2754
 2755 030516  000401                        br        end
 2756
```

SEQ 0061

SIZER Supplied Program Data

```
 2757 030520                          ABORT:  DOCLN                   ;DO CLEAN-UP AND ABORT THE PASS
 2758 030522                           END:   ENDINIT                 ;FINISHED
 2759
 2760
 2761 030524                          BGNAUTO
 2762 030524                          DODU LOGUNIT
 2763 030532                          ENDAUTO
 2764
 2765 030534                          BGNCLN
 2766 030534  005077  151546                 clr     @IPreg          ;get controller into know state
 2767 030540                                 Break                   ;waste some time
 2768 030542                          ENDCLN
 2769
 2770 030544                          BGNDU
 2771 030544                                 printf @DRPunt,LOGUNIT,un't
 2772 030574                          ENDDU
 2773
```

SIZER Supplied Program Data

```
   2775 030576                                  BGNTST 1
   2776 030576  004737  021520                         jsr     pc,hrdint               ;init the controller
   2777 030602  122737  000023  002322                 cmpb    @#rqdx3,mdlnbr          ;check if RQDX3 controller
   2778 030610  001403                                 beq     2$
   2779 030612  042737  100000  002320                 bic     @#bit15,untflgs         ;if other then RQDX3 than impossible to run auto siz
er or in auto mo
   2780 030620  032737  100000  002320  2$:            bit     @#bit15,untflgs         ;test if auto mode is enabled
   2781 030626  001402                                 beq     1$                      ;if not skip the auto sizer routine
   2782 030630  004737  022570                         jsr     pc,AUTOsizer            ;if it is then run AUTO SIZER on the controller
   2783 030634                                  1$:
   2784
   2785 030634  004737  021520                         jsr     pc,hrdint               ;reinitialize controller incase it was in a weird st
ate
   2786 030640                                         printb  @#pb9,mdlnbr            ;print the disk controller model number
   2787 030664                                         printb  @#pb10,mcdnbr     ;print microcode version number in decimal
   2788
   2789 030710                                  ELPcmd:
   2790
   2791 030710  032737  100000  002320                 bit     @#bit15,untflgs         ;test if auto mode is enabled
   2792 030716  001011                                 bne     1$                      ;branch if in auto mode else
   2793 030720                                  GMANID ASK.prg,PRGnam,A,-1,6.,6.,yes    ;ask for the User what local program he wants to run
   2794 030740  000411                                 br      2$
   2795 030742                                  1$:
   2796 030742  012737  047506  002660                 mov     @#"FO,PRGnam            ;place "FORMAT" into ascii buffer if in auto mode
   2797 030750  012737  046522  002662                 mov     @#"RM,PRGnam+2
   2798 030756  012737  052101  002664                 mov     @#"AT,PRGnam+4
   2799 030764                                  2$:
   2800 030764                                  EXLCPRG PRGnam                          ;Execute Local program "FORMAT" or what ever they wr
ote
        030764  032737  100000  002516  ELP6:  bit     @#bit15,cmdrng+2        ;test ownership of ring make sure we own it
        030772  001374                         bne     ELP6                    ;if we don't own it wait until we do
        030774  012737  000022  002430         mov     #22,cmdlen              ;load lenght of packet to be send
        031002  112737  000000  002432         movb    #0,cmdlen+2             ;load msg type and credit
        031010  112737  000002  002433         movb    @dup.id,cmdlen+3        ;load DUP connection ID
        031016  005237  002434                 inc     cmdpak                  ;load new CRN
        031022  005037  002436                 clr     cmdpak+2
        031026  005037  002440                 clr     cmdpak+4
        031032  005037  002442                 clr     cmdpak+6
        031036  012737  000003  002444         mov     #op.elp,cmdpak+10       ;load up opcode
        031044  012737  000001  002446         mov     #stdaln,cmdpak+12       ;stand alone modifier
        031052  012700  000006                 mov     #6,r0                   ;6 letters transfer
        031056  012701  002450                 mov     #cmdpak+14,r1           ;starting address to place program name
        031062  012702  002660                 mov     #PRGnam,r2              ;start of Program Name
        031066  112221                 rfdj5:  movb    (r2)+,(r1)+             ;add 2 to bycnt then store
        031070  077002                         sob     r0,rfdj6

        031072  012777  031134  151210         mov     #RFD6,@vector           ;NEW VECTOR PLACE
        031100  012737  002334  002510         mov     #rsppak,rsprng          ;load response packet area into ring
        031106  012737  002434  002514         mov     #cmdpak,cmdrng          ;load command packet area into ring
        031114  012737  140000  002512         mov     #140000,RSPRNG+2        ;PORT OWNERSHIP BIT.
        031122  012737  100000  002516         mov     #bit15,CMDRNG+2
        031130  004737  020746                 jsr     pc,POLLWT               ;GO TO POLL AND WAIT ROUTINE.
                                        ;*****************************************************************************
        031134                             RFD6:                                ;INTR TO HERE.
        031134  062706  000006                 add     #6,sp                   ;fix stack for interrupt (4), pollwt subrtn (2)
        031140  012777  030222  151142         mov     #intsrv,@vector         ;CHANGE VECTOR
        031146  004737  026226                 jsr     pc,RSPCHK
                                                                               ;GO TO ROUTINE THAT WILL CHECK ON
                                                                               ;THE RESPONSE RECVD FROM THE MUT.
                                                                               ;IT WILL CHECK THE CMD REF
```

SIZER Supplied Program Data

```
                                                                        ;NUM, THE ENDCODE AND STATUS.
 2801
 2802 031152  122737  000011  002353          cmpb    @bit3+bit0,rsppak+17    ;is this program a standalone,DUP dialog type
 2803 031160  001406                          beq     1$
 2804 031162                                  ERRDF   2,DF3                   ;"Device Fatal can't do remote programs"
 2805 031172  000137  034572                  jmp     dropunt                 ;drop unit and go on
 2806 031176                          1$:
 2807 031176                          RCDcmd:
 2808 031176                          RECVDAT @datare,^80.
      031176  032737  100000  002516  RCD7:   bit     @bit15,cmdrng+2         ;test ownership of ring make sure we own it
      031204  001374                          bne     RCD7                    ;if we don't own it wait until we do
      031206  012737  000034  002430          mov     @34,cmdlen              ;load lenght of packet to be send
      031214  112737  000000  002432          movb    @0,cmdlen+2             ;load msg type and credit
      031222  112737  000002  002433          movb    @dup.id,cmdlen+3        ;load DUP connection ID
      031230  005237  002434                  inc     cmdpak                  ;load new CRN
      031234  005037  002436                  clr     cmdpak+2
      031240  005037  002440                  clr     cmdpak+4
      031244  005037  002442                  clr     cmdpak+6
      031250  012737  000005  002444          mov     @op.rec,cmdpak+10       ;load up opcode
      031256  005037  002446                  clr     cmdpak+12               ;no modifiers
      031262  012737  000120  002450          mov     @80.,cmdpak+14
      031270  005037  002452                  clr     cmdpak+16
      031274  012737  002534  002454          mov     @datare,cmdpak+20       ;load address of buffer describtor
      031302  005037  002456                  clr     cmdpak+22
      031306  005037  002460                  clr     cmdpak+24
      031312  005037  002462                  clr     cmdpak+26
      031316  005037  002464                  clr     cmdpak+30
      031322  005037  002466                  clr     cmdpak+32

      031326  012777  031370  150754          mov     @RFD7,@vector           ;NEW VECTOR PLACE
      031334  012737  002334  002510          mov     #rsppak,rsprng          ;load response packet area into ring
      031342  012737  002434  002514          mov     #cmdpak,cmdrng          ;load command packet area into ring
      031350  012737  140000  002512          mov     @140000,RSPRNG+2        ;PORT OWNERSHIP BIT.
      031356  012737  100000  002516          mov     @bit15,CMDRNG+2
      031364  004737  020746                  jsr     pc,POLLWT               ;GO TO POLL AND WAIT ROUTINE.
                                       ;*****************************************************************
      031370                              RFD7:                               ;INTR TO HERE.
      031370  062706  000006                  add     @6,sp                   ;fix stack for interrupt (4), pollwt subrtn (2)
      031374  012777  030222  150706          mov     @intsrv,@vector         ;CHANGE VECTOR
      031402  004737  026226                  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.
 2809                                 ;+
 2810                                 ;       get
 2811                                 ;       r3 = type
 2812                                 ;       r4 = SA adrs
 2813                                 ;       r5 = sub number
 2814                                 ;-
 2815 031406  113703  002535          DUPDLG: movb    datare+1,r3     ;get dup type info
 2816 031412  006203                          asr     r3
 2817 031414  006203                          asr     r3
 2818 031416  006203                          asr     r3
 2819 031420  006203                          asr     r3
 2820 031422  042703  177760                  bic     @type,r3        ;mask off all but DUP type
 2821                                 ;       printx  @PB7,r3         ; received DUP command type XX"
```

SIZER Supplied Program Data

```
 2822 031426  013705  002534                  mov     datare,r5       ;get duo message number info
 2823 031432  042705  170000                  bic     #msgnbr,r5      ;clear out top 4 bits
 2824                               ;         printx  #PB8,r5         ;"received dup message number XX'
 2825
 2826
 2827                               ;+
 2828                               ; Check for the type.
 2829                               ; if QUESTION type, it will be answered by sending
 2830                               ; an answer through a Send command which will be followed
 2831                               ; by a Receive command to await further instructions.
 2832                               ;
 2833                               ; If a DEFAULT QUESTION type is given an answer will
 2834                               ; either be given or a blank send command returned.
 2835                               ; Either way we will do a Send command followed by a
 2836                               ; Receive command.
 2837                               ;
 2838                               ; if INFORMATIONAL type, check message number and type
 2839                               ; inforrmation according to message numb   given.
 2840                               ;
 2841                               ; if FATAL ERROR type, check message number and print
 2842                               ; error message accordingly. No other commands will
 2843                               ; be given following this type of command.
 2844                               ;
 2845                               ; If TERMINATION type check the message number and print the
 2846                               ; correct message. Ussually this implies a succesful
 2847                               ; end to the formatter. After this command we exit the program
 2848                               ;
 2849                               ; If SPECIAL type we are asking for the FCT table to be passed
 2850                               ; to the RQDX3 controller. We will send the table with a Send
 2851                               ; command and then to a Receive command to proceed.
 2852                               ;
 2853 031436  022703  000001       qstn:   cmp     #Question,r3    ;test for "question" subtype
 2854 031442  001117                       bne     dfqstn          ;if not branch
 2855 031444  032737  020000  002320       bit     #bit13,untflgs  ;see if we are working on a known controller
 2856 031452  001077                       bne     qnbra           ;if not type out ascii
 2857 031454  122737  000106  002660       cmpb    #'F,prgnam      ;if running the format program then print info
 2858 031462  001073                       bne     qnbra           ;else just go for an answer
 2859
 2860 031464  004737  026210       qnbr0:  jsr     pc,clrDUPbuf    ;clear out data buffer so DRS macros don't show default
 2861 031470  022705  000000                cmp     #0,r5           ;check for message number
 2862 031474  001036                       bne     qnbr7           ;check for next message number
 2863 031476  032737  100000  002320       bit     #bit15,untflgs
 2864 031504  001011                       bne     1$
 2865 031506                               GMANID  qfdat,DATARE,A,177777,10.,10.,no                        ;DATE MM-DD-YYYY ?
 2866 031526  000417                       br      2$
 2867 031530  012737  033060  002534  1$:  mov     #"06,datare     ;The date is not used anyway so any date will do
 2868 031536  012737  030455  002536       mov     #"-1,datare+2   ;I'll be celebrating this day
 2869 031544  012737  026467  002540       mov     #"7-,datare+4
 2870 031552  012737  034461  002542       mov     #"19,datare+6
 2871 031560  012737  033070  002544       mov     #"86,datare+10
 2872 031566  000137  032270       2$:     jmp     SDTcmd          ;branch to Send Data command
 2873
 2874 031572  022705  000007       qnbr7:  cmp     #7,r5           ;check for message number
 2875 031576  001025                       bne     qnbra           ;check for next message number
 2876 031600  032737  100000  002320       bit     #bit15,untflgs
 2877 031606  001011                       bne     1$
 2878 031610                               GMANID  qfser,DATARE,A,177777,8.,10.,NO                 ;SERIAL NUMBER 9 digits ?
```

SIZER Supplied Program Data

```
 2879 031630  000406                        br      2$
 2880 031632  013700  002316          1$:   mov     sernbr,r0
 2881 031636  012701  002534                mov     @datare,r1      ;place to stick ascii
 2882 031642  004737  025720                jsr     pc,OCTASC       ;convert octal to decimal ascii
 2883 031646  000137  032270          2$:   jmp     SDTcmd
 2884
 2885 031652  004737  026114          qnbra: jsr    pc,typDUPbuf    ;type out ASCII sent by disk controller
 2886 031656                                GMANID  ASK.ANSWER,DATARE,A,177777,0.,10.,YES                    ;give it an answer
 2887 031676  000137  032270                jmp     SDTcmd          ;branch to Send Data command
 2888
 2889
 2890
 2891
 2892 031702  022703  000002          dfqstn: cmp   @DefQuest,r3    ;test for "Default Question" subtype
 2893 031706  001402                        beq     1$
 2894 031710  000137  032504                jmp     infrm           ;if not branch
 2895 031714  032737  020000  002320  1$:   bit     @bit13,untflgs   see if we are working on a known controller
 2896 031722  001150                        bne     dqnbra          ;if not type out ascii
 2897 031724  122737  000106  002660        cmpb    @'F,prgnam      ;if running the format program then print info
 2898 031732  001144                        bne     dqnbra          ;else just go for an answer
 2899
 2900 031734  004737  026210          dqnbr1: jsr   pc,clrDUPbuf    ;clear out data buffer so DRS macros don't show default
 2901 031740  022705  000001                cmp     @1,r5           ;check for message number
 2902 031744  001043                        bne     dqnbr4          ;check for next message number
 2903                                                                ;put in message number
 2904 031746  032737  100000  002320        bit     @bit15,untflgs
 2905 031754  001011                        bne     3$
 2906 031756                                GMANID  dfunt,DATARE,A,177777,0,3,YES   ;Ask for UNIT NUMBER 0-255 ?
 2907 031776  000406                        br      4$
 2908 032000  013700  002312          3$:   mov     unit,r0         ;get unit number if in auto mode from Hardware P table
 2909 032004  012701  002534                mov     @datare,r1      ;store decimal ascii conversion in data area
 2910 032010  004737  025720                jsr     pc,OCTASC       ;convert octal to ascii decimal in data area
 2911
 2912 032014  012701  002534          4$:   mov     @datare,r1      ;address of ascii decimal data
 2913 032020  012700  002312                mov     @unit,r0        ;address to store octal conversion
 2914 032024  004737  026006                jsr     pc,ASCDEC       ;convert ascii decimal to octal
 2915 032030  022737  000003  002312  2$:   cmp     @3,unit         ;make sure unit number is less than 4 or between 0-3
 2916 032036  002004                        bge     1$
 2917 032040  162737  000004  002312        sub     @4,unit         ;subtract 4 until unit is less than four
 2918 032046  000770                        br      2$
 2919 032050  000137  032270          1$:   jmp     SDTcmd          ;branch to Send Data command
 2920
 2921 032054  022705  000004          dqnbr4: cmp   @4,r5           ;check for message number
 2922 032060  001021                        bne     dqnbr5          ;check for next message number
 2923 032062  012737  000116  002534        mov     @'N,datare      ;set the default for NO
 2924 032070  032737  100000  002320        bit     @bit15,untflgs
 2925 032076  001010                        bne     1$
 2926 032100                                GMANID  dfbad,DATARE,A,177777,0,1,YES                   ;Use existing bad block info
rmation (Y or N)?
 2927 032120  000137  032270          1$:   jmp     SDTcmd          ;branch to Send Data command
 2928
 2929 032124  022705  000005          dqnbr5: cmp   @5,r5           ;check for message number
 2930 032130  001021                        bne     dqnbr6          ;check for next message number
 2931 032132  012737  000131  002534        mov     @'Y,datare      ;Set the default for YES
 2932 032140  032737  100000  002320        bit     @bit15,untflgs
 2933 032146  001010                        bne     1$
 2934 032150                                GMANID  dfdwn,DATARE,A,177777,0,1,YES                   ;Use Down Line Load (Y or N)
?
 2935 032170  000137  032270          1$:   jmp     SDTcmd          ;branch to Send Data command
```

```
   2936
   2937 032174  022705  000006          dqnbr6: cmp     #6,r5           ;check for message number
   2938 032200  001021                          bne     dqnbra          ;check for next message number
   2939 032202  012737  000131  002534          mov     # 1,datare       ;set the default for YES
   2940 032210  032737  100000  002320          bit     #bit15,untflgs
   2941 032216  001010                          bne     1$
   2942 032220                                  GMANID  dfcon,DATARE,A,177777,0,1,YES                   ;Continue if bad block infor
mation is inaccessable (Y or N)?
   2943 032240  000137  032270          1$:     jmp     SDTcmd
   2944
   2945                                                                 ;if unknown use default and continue
   2946                                                                 ;who knows maybe it will be useful some day
   2947 032244  004737  026114          dqnbra: jsr     pc,typDUPbuf    ;type out ASCII sent by disk controller
   2948 032250                                  GMANID  ASK.ANSWER,DATARE,A,177777,0..10.,YES                   ;give it an answer
   2949 032270                          SDTcmd:
   2950 032270                                  SENDDAT #datare,#10.    ;sent the answer
        032270  032737  100000  002516  SDT10:  bit     #bit15,cmdrng+2         ;test ownership of ring make sure we own it
        032276  001374                          bne     SDT10                   ;if we don't own it wait until we do
        032300  012737  000034  002430          mov     #34,cmdlen              ;load lenght of packet to be send
        032306  112737  000000  002432          movb    #0,cmdlen+2             ;load msg type and credit
        032314  112737  000002  002433          movb    #dup.id,cmdlen+3        ;load DUP connection ID
        032322  005237  002434                  inc     cmdpak                  ;load new CRN
        032326  005037  002436                  clr     cmdpak+2
        032332  005037  002440                  clr     cmdpak+4
        032336  005037  002442                  clr     cmdpak+6
        032342  012737  000004  002444          mov     #op.sen,cmdpak+10       ;load up opcode
        032350  005037  002446                  clr     cmdpak+12               ;no modifiers
        032354  012737  000012  002450          mov     #10.,cmdpak+14
        032362  005037  002452                  clr     cmdpak+16
        032366  012737  002534  002454          mov     #datare,cmdpak+20       ;load address of buffer descriptor
        032374  005037  002456                  clr     cmdpak+22
        032400  005037  002460                  clr     cmdpak+24
        032404  005037  002462                  clr     cmdpak+26
        032410  005037  002464                  clr     cmdpak+30
        032414  005037  002466                  clr     cmdpak+32

        032420  012777  032462  147662          mov     #RFD10,@vector          ;NEW VECTOR PLACE
        032426  012737  002334  002510          mov     #rsppak,rsprng          ;load response packet area into ring
        032434  012737  002434  002514          mov     #cmdpak,cmdrng          ;load command packet area into ring
        032442  012737  140000  002512          mov     #140000,RSPRNG+2        ;PORT OWNERSHIP BIT.
        032450  012737  100000  002516          mov     #bit15,CMDRNG+2
        032456  004737  020746                  jsr     pc,POLLWT               ;GO TO POLL AND WAIT ROUTINE.
                                        ;*****************************************************************
        032462                              RFD10:                              ;INTR TO HERE.
        032462  062706  000006                  add     #6,sp                   ;fix stack for interrupt (4), pollwt subrtn (2)
        032466  012777  030222  147614          mov     #intsrv,@vector         ;CHANGE VECTOR
        032474  004737  026226                  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.
   2951 032500  000137  031176                  jmp     RCDcmd          ;do another receive cmd
   2952
   2953
   2954
   2955 032504  022703  000003          infrm:  cmp     #Inform,r3      ;test for "Informational" subtype
   2956 032510  001046                          bne     term            ;if not branch
   2957 032512  032737  020000  002320          bit     #bit13,untflgs  ;see if we are working on a known controller
```

```
 2958 032520  0C1036                          bne     inbra           ;if not type out asc'i
 2959 032522  122737  000106  002660          cmpb    @'F,prgnam      ;if running the format program then print info
 2960 032530  001032                          bne     'nbra
 2961
 2962 032532  022705  000000          inbr0:  cmp     @0,r5           ;check for message number
 2963 032536  001012                          bne     inbr1           ;check for next message number
 2964 032540  004737  026210                  jsr     pc,clrDUPbuf    ;clear out DUP buffer so there 's no echo on last ASCII
 2965 032544                                  printf  @sfbegt         ;format begun
 2966 032564  022705  000001          inbr1:  cmp     @1,r5           ;check for message number
 2967 032570  001012                          bne     inbra           ;check for next message number
 2968 032572  004737  026210                  jsr     pc,clrDUPbuf    ;clear out DUP buffer so there 's no echo on last ASCII
 2969 032576                                  printf  @sfdont         ;format complete
 2970
 2971 032616  004737  026114          'nbra:  jsr     pc,typDUPbuf    ;type out ASCII sent by d'sk controller
 2972 032622  000137  031176                  jmp     RCDcmd          ;do another receive command
 2973
 2974
 2975
 2976 032626  022703  000004          term:   cmp     @terminat,r3    ;test for terminat'on type
 2977 032632  001056                          bre     ftler           ;if not branch
 2978 032634  032737  020000  002320          bit     @bit13,untflgs  ;see if we are working on a known controller
 2979 032642  001036                          bne     tnbra           ;if not type out ascii
 2980 032644  122737  000106  002660          cmpb    @'F,prgnam      ;if runn'ng the format program then branch to error routine
 2981 032652  001032                          bne     tnbra
 2982
 2983 032654  022705  000014          tnbr12: cmp     @12.,r5         ;test for sub number @1
 2984 032660  001012                          bne     tnbr13          ;branch if not sub number @1
 2985 032662                                  printf  @sffcut
 2986 032702  000137  034572                  jmp     dropunt         ;drop test unit and end pass
 2987
 2988 032706  022705  000015          tnbr13: cmp     @13.,r5         ;test for msg number
 2989 032712  001012                          bne     tnbra           ;branch if not right number
 2990 032714                                  printf  @sffcnt
 2991 032734  000137  034572                  jmp     dropunt         ;drop test unit and end pass
 2992
 2993 032740  004737  026114          tnbra:  jsr     pc,typDUPbuf    ;type out ASCII sent by disk controller
 2994 032744                                  printf  @PF2            ;print finished local program without procedure error
 2995 032764  000137  034600                  jmp     etst            ;end DUP diaglog but stay 'n test loop
 2996
 2997
 2998
 2999 032770  022703  000005          ftler:  cmp     @Ftlerr,r3      ;test for "Fatal Error" subtype
 3000 032774  001402                          beq     1$
 3001 032776  000137  034252                  jmp     spcl            ;if not branch
 3002 033002  032737  020000  002320  1$:     bit     @bit13,untflgs  ;see if we are working on a known controller
 3003 033010  001004                          bne     3$              ;if not type out ascii
 3004 033012  122737  C00106  002660          cmpb    @'F,prgnam      ;if running the format program then branch to error rout'ne
 3005 033020  001414                          beq     2$
 3006 033022  004737  026114          3$:     jsr     pc,typDUPbuf    ;type out ASCII sent by disk controller
 3007 033026                                  printf  @DF15           ;Fatal error reported when running local program
 3008 033046  000137  034572                  jmp     dropunt         ;drop unit and end pass
 3009
 3010 033052                          2$:     ERRHRD  1,HRDO          ;Hard device  error
 3011
 3012 033062  022705  000001          fnbr1:  cmp     @1,r5           ;test for sub number @1
 3013 033066  001012                          bne     fnbr2           ;branch if not sub number @1
 3014 033070                                  printb  @efstat         ;"GET STATUS failure"
```

SIZER Supplied Program Data

```
3015 033110  000137  034600                    jmp     etst            ;end DUP d'aglog but stay in test loop
3016
3017 033114  022705  000002          fnbr2:    cmp     #2.,r5          ;test for msg number
3018 033120  001012                            bne     fnbr3           ;branch if not right number
3019 033122                                    printf  #efsndt         ;
3020 033142  000137  034600                    jmp     etst            ;end DUP diaglog but stay in test loop
3021
3022 033146  022705  000003          fnbr3:    cmp     #3.,r5          ;test for msg number
3023 033152  001012                            bne     fnbr4           ;branch if not right number
3024 033154                                    printf  #efcmdt         ;
3025 033174  000137  034600                    jmp     etst            ;end DUP diaglog but stay in test loop
3026
3027 033200  022705  000004          fnbr4:    cmp     #4.,r5          ;test for msg number
3028 033204  001012                            bne     fnbr5           ;branch if not right number
3029 033206                                    printf  #efrcvt         ;
3030 033226  000137  034600                    jmp     etst            ;end DUP diaglog but stay in test locp
3031
3032 033232  022705  000005          fnbr5:    cmp     #5.,r5          ;test for msg number
3033 033236  001012                            bne     fnbr6           ;branch if not right number
3034 033240                                    printf  #efbust         ;
3035 033260  000137  034600                    jmp     etst            ;end DUP diaglog but stay in test loop
3036
3037 033264  022705  000006          fnbr6:    cmp     #6.,r5          ;test for msg number
3038 033270  001012                            bne     fnbr7           ;branch if not right number
3039 033272                                    printf  #efinit         ;
3040 033312  000137  034600                    jmp     etst            ;end DUP diaglog but stay in test loop
3041
3042 033316  022705  000007          fnbr7:    cmp     #7.,r5          ;test for msg number
3043 033322  001012                            bne     fnbr8           ;branch if not right number
3044 033324                                    printf  #efnut          ;'Q-PORT send error "
3045 033344  000137  034600                    jmp     etst            ;end DUP diaglog but stay in test loop
3046
3047 033350  022705  000010          fnbr8:    cmp     #8.,r5          ;test for msg number
3048 033354  001012                            bne     fnbr9           ;branch if not right number
3049 033356                                    printf  #efdxft
3050 033376  000137  034600                    jmp     etst            ;end DUP diaglog but stay in test loop
3051
3052 033402  022705  000011          fnbr9:    cmp     #9.,r5          ;test for msg number
3053 033406  001012                            bne     fnbr10          ;branch if not right number
3054 033410                                    printf  #effcct         ;"Q-PORT send error "
3055 033430  000137  034600                    jmp     etst            ;end DUP diaglog but stay in test loop
3056
3057 033434  022705  000012          fnbr10:   cmp     #10.,r5         ;test for msg number
3058 033440  001012                            bne     fnbr11          ;branch if not right number
3059 033442                                    printf  #efsekt         ;"Q-PORT send error "
3060 033462  000137  034600                    jmp     etst            ;end DUP diaglog but stay in test loop
3061
3062 033466  022705  000013          fnbr11:   cmp     #11.,r5         ;test for msg number
3063 033472  001012                            bne     fnbr12          ;branch if not right number
3064 033474                                    printf  #efrcct         ;"Q-PORT send error "
3065 033514  000137  034600                    jmp     etst            ;end DUP diaglog but stay in test loop
3066
3067 033520  022705  000014          fnbr12:   cmp     #12.,r5         ;test for msg number
3068 033524  001012                            bne     fnbr13          ;branch if not right number
3069 033526                                    printf  #eflbft         ;"Q-PORT send error "
3070 033546  000137  034600                    jmp     etst            ;end DUP diaglog but stay in test loop
3071
```

```
3072 033552  022705  000015          fnbr13: cmp     #13.,r5         ;test for msg number
3073 033556  001012                         bne     fnbr14          ;branch if not right number
3074 033560                                 printf  #effcwt         ;"Q PORT send error '
3075 033600  000137  034600                 jmp     etst            ;end DUP diaglog but stay in test loop
3076
3077 033604  022705  000016          fnbr14: cmp     #14.,r5         ;test for msg number
3078 033610  001012                         bne     fnbr15          ;branch if not right number
3079 033612                                 printf  #efrcrt         ;"Q-PORT send error "
3080 033632  000137  034600                 jmp     etst            ;end DUP diaglog but stay in test loop
3081
3082 033636  022705  000017          fnbr15: cmp     #15.,r5         ;test for msg number
3083 033642  001012                         bne     fnbr16          ;branch if not right number
3084 033644                                 printf  #efrcwt         ;"Q-PORT send error
3085 033664  000137  034600                 jmp     etst            ;end DUP diaglog but stay in test loop
3086
3087 033670  022705  000020          fnbr16: cmp     #16.,r5         ;test for msg number
3088 033674  001012                         bne     fnbr17          ;branch if not right number
3089 033676                                 printf  #efrcft         ;"Q PORT send error "
3090 033716  000137  034600                 jmp     etst            ;end DUP diaglog but stay in test loop
3091
3092 033722  022705  000021          fnbr17: cmp     #17.,r5         ;test for msg number
3093 033726  001012                         bne     fnbr18          ;branch if not right number
3094 033730                                 printf  #effcrt         ;"Q-PORT send error "
3095 033750  000137  034600                 jmp     etst            ;end DUP diaglog but stay in test loop
3096
3097 033754  022705  000022          fnbr18: cmp     #18.,r5         ;test for msg number
3098 033760  001012                         bne     fnbr19          ;branch if not right number
3099 033762                                 printf  #effcnt         ;
3100 034002  000137  034600                 jmp     etst            ;end DUP diaglog but stay in test loop
3101
3102 034006  022705  000023          fnbr19: cmp     #19.,r5         ;test for msg number
3103 034012  001012                         bne     fnbr20          ;branch if not right number
3104 034014                                 printf  #effcdt         ;
3105 034034  000137  034600                 jmp     etst            ;end DUP diaglog but stay in test loop
3106
3107 034040  022705  000024          fnbr20: cmp     #20.,r5         ;test for msg number
3108 034044  001012                         bne     fnbr21          ;branch if not right number
3109 034046                                 printf  #eftmot         ;"Q-PORT send error "
3110 034066  000137  034600                 jmp     etst            ;end DUP diaglog but stay in test loop
3111
3112 034072  022705  000025          fnbr21: cmp     #21.,r5         ;test for msg number
3113 034076  001012                         bne     fnbr22          ;branch if not right number
3114 034100                                 printf  #efillt         ;"Q-PORT send error "
3115 034120  000137  034600                 jmp     etst            ;end DUP diaglog but stay in test loop
3116
3117 034124  022705  000026          fnbr22: cmp     #22.,r5         ;test for msg number
3118 034130  001012                         bne     fnbr23          ;branch if not right number
3119 034132                                 printf  #efwart         ;"Q-PORT send error "
3120 034152  000137  034600                 jmp     etst            ;end DUP diaglog but stay in test loop
3121
3122 034156  022705  000027          fnbr23: cmp     #23.,r5         ;test for msg number
3123 034162  000412                         br      fnbr24          ;branch if not right number
3124 034164                                 printf  #efinpt         ;"Q-PORT send error "
3125 034204  000137  034600                 jmp     etst            ;end DUP diaglog but stay in test loop
3126
3127
3128 034210  022705  000030          fnbr24: cmp     #24.,r5         ;test for msg number
```

SEQ 0070

SIZER Supplied Program Data

```
  3129 034214  001012                        bne     1$
  3130 034216                                printf  @efmedt
  3131 034236  000137  034600                jmp     etst            ;end DUP diaglog but stay in test loop
  3132
  3133 034242  004737  026114        1$:     jsr     pc,typDUPbuf    ;type out ASCII sent by disk controller
  3134 034246  000137  034600                jmp     etst            ;end DUP diaglog but stay in test loop
  3135
  3136
  3137
  3138
  3139 034252  022703  000006        spcl:   cmp     @specl,r3       ;test for special type
  3140 034256  001137                        bne     unkwn           ;branch if not known
  3141 034260  032737  020000  002320        bit     @bit13,untflgs  ;see if we are working on a known controller
  3142 034266  001004                        bne     2$              ;if not type out ascii
  3143 034270  122737  000106  00266C        cmpb    @'F,prgnam      ;if running the format program then print info
  3144 034276  001414                        beq     1$
  3145 034300  004737  026114        2$:     jsr     pc,typDUPbuf    ;type out ASCII sent by disk controller
  3146 034304                                printf  @DF16           ;special command issued by local program did not know how to
handle
  3147 034324  000137  034556                jmp     unkwn           ;report error
  3148
  3149 034330  022705  000002        1$:     cmp     @2,r5           ;test for message number 1
  3150 034334  001110                        bne     unkwn           ;branch if not known
  3151 034336  004737  024154                jsr     pc,biduit       ;go get or build UIT table
  3152                               ;       printx  @qfuit,uin,unit ;"Entering UITO: on drive number 2"
  3153 034342                                SENDDAT UITadr,@UITsiz  ;sent Unit Information table
       034342  032737  100000  002516  SDT11:  bit     @bit15,cmdrng+2         ;test ownership of ring make sure we own it
       034350  001374                        bne     SDT11                   ;if we don't own it wait until we do
       034352  012737  000034  002430        mov     @34,cmdlen              ;load lenght of packet to be send
       034360  112737  000000  002432        movb    @0,cmdlen+2             ;load msg type and credit
       034366  112737  000002  002433        movb    @dup.id,cmdlen+3        ;load DUP connection ID
       034374  005237  002434                inc     cmdpak                  ;load new CRN
       034400  005037  002436                clr     cmdpak+2
       034404  005037  002440                clr     cmdpak+4
       034410  005037  002442                clr     cmdpak+6
       034414  012737  000004  002444        mov     @op.sen,cmdpak+10       ;load up opcode
       034422  005037  002446                clr     cmdpak+12               ;no modifiers
       034426  012737  000102  002450        mov     @UITsiz,cmdpak+14
       034434  005037  002452                clr     cmdpak+16
       034440  013737  002304  002454        mov     UITadr,cmdpak+20        ;load address of buffer describtor
       034446  005037  002456                clr     cmdpak+22
       034452  005037  002460                clr     cmdpak+24
       034456  005037  002462                clr     cmdpak+26
       034462  005037  002464                clr     cmdpak+30
       034466  005037  002466                clr     cmdpak+32

       034472  012777  034534  145610        mov     @RFD11,@vector          ;NEW VECTOR PLACE
       034500  012737  002334  002510        mov     @rsppak,rsprng          ;load response packet area into ring
       034506  012737  002434  002514        mov     @cmdpak,cmdrng          ;load command packet area into ring
       034514  012737  140000  002512        mov     @140000,RSPRNG+2        ;PORT OWNERSHIP BIT.
       034522  012737  100000  002516        mov     @bit15,CMDRNG+2
       034530  004737  020746                jsr     pc,POLLWT               ;GO TO POLL AND WAIT ROUTINE.
                                  ;*************************************************************************
       034534                        RFD11:                                  ;INTR TO HERE.
       034534  062706  000006                add     @6,sp                   ;fix stack for interrupt (4), pollwt subrtn (2)
       034540  012777  030222  145542        mov     @intsrv,@vector         ;CHANGE VECTOR
       034546  004737  026226                jsr     pc,RSPCHK
                                                                             ;GO TO ROUTINE THAT WILL CHECK ON
```

SEQ 0071

SIZER Supplied Program Data

```
                                                        ;THE RESPONSE RECVD FROM THE MUT.
                                                        ;IT WILL CHECK THE CMD REF
                                                        ;NUM, THE ENDCODE AND STATUS.
 3154 034552  000137  031176                jmp     RCDcmd          ;do another receive cmd
 3155
 3156
 3157 034556                        unkwn:  ERRSF   0,SFO            ; system error unkown response
 3158 034566  004737  026302                jsr     pc,PRNTpkt      ;type out packet information
 3159
 3160 034572                        dropunt:
 3161 034572                                DODU    LOGUNIT         ;drop the unit
 3162
 3163 034600                        etst:
 3164 034600                                docln                   ;take controller offline
 3165 034602                        ENDTST
```

SIZER Supplied Program Data

```
   3167 034604                          BGNHRD
   3168
   3169 034606                               GPRMA ip.adr,0,0,160000,177776,YES   ;GET IP REG ADDR (170000 177776)
   3170                                                                           ;PLACE IN WORD 2 OF THE TABLE
   3171                                                                           ;DEFAULT VALUE IS FROM DEFAULT
   3172                                                                           ;TABLE.
   3173
   3174 034616                               GPRMA vec.adr,2,0,0,776,YES          ;GET THE VECTOR ADDR (OCTAL 0-776)
   3175                                                                           ;PLACE IN WORD
   3176                                                                           ;DEFAULT VALUE IS FROM DEFAULT
   3177                                                                           ;TABLE.
   3178 034626                               GPRML auto.md,10,bit15,YES           ;ask if they want to go in to auto mode
   3179                                                                           ;This will format the drive using the autosizer
   3180
   3181 034634                               XFERF label0                         ;IF LAST GPRML INPUT IS FALSE (N) TRANSFER
   3182                                                                           ;CONTROL TO LABEL.
   3183
   3184 034636                               GPRMD drv.nbr,4,D,-1,0,255.,YES       ;GET THE LOGICAL DRIVE (Decimal 0-255)
   3185                                                                          ;PLACE IN WORD
   3186                                                                          ;DEFAULT VALUE IS FROM DEFAULT
   3187                                                                          ;TABLE.
   3188
   3189
   3190 034650                               GPRMD ser.nbr,6,D,-1,1,012345.,YES    ;GET THE DRIVE SERIAL NUMBER
   3191                                                                          ;PLACE IN WORD
   3192                                                                          ;DEFAULT VALUE IS FROM DEFAULT
   3193                                                                          ;TABLE.
   3194
   3195 034662                               DISPLAY warning                      ;The information on this drive will be totally des
troyed
   3196
   3197 034666                               GPRML do.cont,10,bit14,YES           ;ask if they want to go on even though info will b
e creamed
   3198
   3199
   3200 034674                          label0:                                   ;We don't ask the warning question if they go into
 manual mode.
   3201                                                                           ;They'll have to answer the question again
   3202 034674                          exit hrd
   3203 034676                           ENDHRD
   3204
   3205
   3206 034676                           LASTAD
        034702                          L$LAST::
   3207 034702                          ENDMOD
   3208          000001                  .END
```

Symbol table

```
A       = 000000          CONTON  030364          C$RPT = 000025          EFFCRT  020373          FNBR9   033402
ABORT   030520            C$AU    = 000052          C$SEFG= 000046          EFFCWT  020226          FTLER   032770
ABRT3   022420            C$AUTO= 000061          C$SPRI= 000041          EFILLT  020526          FTLERR= 000005
ADR     = 000020 G        C$BRK   = 000022          C$SVEC= 000037          EFINIT  017560          F$AU    = 000015
ASCDEC  026006            C$BSEG= 000004          C$TOME= 000076          EFINFT  020701          F$AUTO= 000020
ASKDBN  025044            C$BSUB= 000002          DATARE  002534          EFL9FT  020143          F$BGN   = 000040
ASKLBN  025102            C$LLCK= 000062          DBN     002702          EFMEDT  020722          F$CLEA= 000007
ASKRBN  025140            C$CLEA= 000012          DECTBL  025772          EFNUT   017623          F$DU    = 000016
ASK.AN= ****** GX         C$CLOS= 000035          DEFQUE= 000002          EFRCCT  020054          F$END   = 000041
ASK.DB  005566            C$CLP1= 000006          DFBAD   016346          EFRCFT  020356          F$HARD= 000004
ASK.LB  005641            C$CPBF= 000074          DFCON   016446          EFRCRT  020307          F$HW    = 000013
ASK.PR  005445            C$CPME= 000075          DFDWN   016416          EFRCVT  017503          F$INIT= 000006
ASK.RB  005714            C$CVEC= 000036          DFPTBL  002262 G        EFRCWT  020332          F$JMP   = 000050
ASK.XB  005513            C$DCLN= 000044          DFQSTN  031702          EFSEKT  020035          F$MOD   = 000000
ASSEMB= 000010            C$DODU= 000051          DFUNT   016305          EFSNDT  017424          F$MSG   = 000011
AUTO    023264 G          C$DRPT= 000024          DF1     007163          EFSTAT  017375          F$PROT= 000021
AUTOBL  024502            C$DU    = 000053          DF11    007447          EFTMOT  020477          F$PWR   = 000017
AUTOEN  024154            C$EDIT= 000003          DF12    007504          EFWART  020600          F$RPT   = 000012
AUTOSI  022570            C$ERDF= 000055          DF13    007540          EF.CON= 000036 G        F$SEG   = 000003
AUTOSZ  023240            C$ERHR= 000056          DF14    007614          EF.NEW= 000035 G        F$SOFT= 000005
AUTO.M  004237            C$ERRO= 000060          DF15    007675          EF.PWR= 000034 G        F$SRV   = 000010
B       = 000011          C$ERSF= 000054          DF16    007765          EF.RES= 000037 G        F$SUB   = 000002
BIT0    = 000001 G        C$ERSO= 000057          DF2     007225          EF.STA= 000040 G        F$SW    = 000014
BIT00   = 000001 G        C$ESCA= 000010          DF3     007274          ELPCMD  030710          F$TEST= 000001
BIT01   = 000002 G        C$ESEG= 000005          DF4     007404          ELP6    030764          GDSCMD  022234
BIT02   = 000004 G        C$ESUB= 000003          DIAGMC= 000000          END     030522          GDS0    021042
BIT03   = 000010 G        C$ETST= 000001          DNINT   022566          ERSEK0= 000003          GDS2    022234
BIT04   = 000020 G        C$EXIT= 000032          DOUDC   024052 G        ERUDON= 000001          GOBIT   022220
BIT05   = 000040 G        C$FREQ= 000101          DOURET  024132          ERUINT= 000002          G$CNTO= 000200
BIT06   = 000100 G        C$FRME= 000100          DO.CON  004360          ESP4    022570          G$DELM= 000372
BIT07   = 000200 G        C$GETB= 000026          DQNBRA  032244          ETST    034600          G$DISP= 000003
BIT08   = 000400 G        C$GETW= 000027          DQNBR1  031734          EVL     = 000004 G      G$EXCP= 000400
BIT09   = 001000 G        C$GMAN= 000043          DQNBR4  032054          E$END   = 002100        G$HILI= 000002
BIT1    = 000002 G        C$GPHR= 000042          DQNBR5  032124          E$LOAD= 000035          G$LOLI= 000001
BIT10   = 002000 G        C$GPRI= 000040          DQNBR6  032174          FNBR1   033062          G$NO    = 000000
BIT11   = 004000 G        C$INIT= 000011          DROPUN  034572          FNBR10  033434          G$OFFS= 000400
BIT12   = 010000 G        C$INLP= 000020          DRPUNT  016013          FNBR11  033466          G$OFSI= 000376
BIT13   = 020000 G        C$MANI= 000050          DRVTXA  004414          FNBR12  033520          G$PRMA= 000001
BIT14   = 040000 G        C$MAP   = 000102          DRVTXB  004442          FNBR13  033552          G$PRMD= 000002
BIT15   = 100000 G        C$MEM   = 000031          DRVTXC  005356          FNBR14  033604          G$PRML= 000000
BIT15T  027160            C$MMU   = 000103          DRVTXO  004537          FNBR15  033636          G$RADA= 000140
BIT2    = 000004 G        C$MSG   = 000023          DRVTX1  004560          FNBR16  033670          G$RADB= 000000
BIT3    = 000010 G        C$OPNR= 000034          DRVTX2  004654          FNBR17  033722          G$RADD= 000040
BIT4    = 000020 G        C$OPNW= 000104          DRVTX3  004751          FNBR18  033754          G$RADL= 000120
BIT5    = 000040 G        C$PNTB= 000014          DRVTX4  004772          FNBR19  034006          G$RADO= 000020
BIT6    = 000100 G        C$PNTF= 000017          DRVTX5  005067          FNBR2   033114          G$XFER= 000004
BIT7    = 000200 G        C$PNTS= 000016          DRVTX6  005164          FNBR20  034040          G$YES   = 000010
BIT8    = 000400 G        C$PNTX= 000015          DRVTX7  005261          FNBR21  034072          HIPRGI  002532
BIT9    = 001000 G        C$PUTB= 000072          DRV.NB  004154          FNBR22  034124          HOE     = 100000 G
BLDUIT  024154            C$PUTW= 000073          DUPDLG  031406          FNBR23  034156          HRDINT  021520
BOE     = 000400 G        C$QIO   = 000377          DUP.ID= 000002          FNBR24  034210          HRDO    010272
CINTR   002504            C$RDBU= 000007          EFBUST  017534          FNBR3   033146          IBE     = 010000 G
CLRDUP  026210            C$REFG= 000047          EFCMDT  017452          FNBR4   033200          IDU     = 000040 G
CMDLEN  002430            C$REL   = 000077          EFDXFT  017657          FNBR5   033232          IER     = 020000 G
CMDPAK  002434            C$RESE= 000033          EFFCCT  017746          FNBR6   033264          INBRA   032616
CMDRNG  002514            C$REVI= 000003          EFFCDT  020442          FNBR7   033316          INBRO   032532
CONT    025716            C$RFLA= 000021          EFFCNT  020416          FNBR8   033350          INBR1   032564
```

Symbol table

```
INFORM= 000003          L$ENVI  002044 G        OP.END= 000200          PB1202  013772          RBN     002730
INFRM   032504          L$ETP   002102 G        OP.ESP= 000002          PB1203  014057          RCDCMD  031176
INTSRV  030222          L$EXP1  002046 G        OP.GDS= 000001          PB1204  014130          RCD5    023026
IPREG   002306          L$EXP4  002064 G        OP.RD = 000003          PB1205  014171          RCD7    031176
IP.ADR  004122          L$EXP5  002066 G        OP.REC= 000005          PB1206  014232          RD.MOD= 000300
ISR    = 000100 G       L$HARD  034606 G        OP.RES= 000000          PB1207  014304          RFDJ6   031066
IXE    = 004000 G       L$HIME  002120 G        OP.SD.= 000044          PB1208  014357          RFD0    021212
I$AU   = 000041         L$HPCP  002016 G        OP.SEN= 000004          PB1209  014413          RFD10   032462
I$AUTO= 000041          L$HPTP  002022 G        OP.SI1= 000005          PB1210  014514          RFD11   034534
I$CLK  = 100006         L$HW    002262 G        OP.SRP= 000100          PB1211  014556          RFD2    022362
I$CLN  = 000041         L$ICP   002104 G        O$APTS= 000000          PB1212  014612          RFD3    022546
I$DU   = 000041         L$INIT  030246 G        O$AU   = 000000         PB1213  014667          RFD4    023010
I$HRD  = 000041         L$LADP  002026 G        O$BGNR= 000000          PB1214  014733          RFD5    023220
I$INIT= 000041          L$LAST  034702 G        O$BGNS= 000000          PB1215  015004          RFD6    031134
I$MOD  = 000041         L$LOAD  002100 G        O$DU   = 000001         PB1216  015045          RFD7    031370
I$MSG  = 000041         L$LUN   002074 G        O$ERRT= 000000          PB1217  015141          RINTR   002506
I$PROT= 000040          L$MREV  002050 G        O$GNSW= 000000          PB1218  015236          RSPCHK  026226
I$PTAB= 000041          L$NAME  002000 G        O$POIN= 000001          PB1219  015313          RSPPAK  002334
I$PWR  = 000041         L$PRIO  002042 G        O$SETU= 000001          PB1220  015352          RSPRNG  002510
I$RPT  = 000041         L$PROT  030240 G        PBF0    011444          PB1221  015437          RSP1    002330
I$SEC  = 100016         L$PRT   002112 G        PBF1    011544          PB1222  015506          RW$PLL= 140002
I$SEG  = 000041         L$REPP  002062 G        PBF10   012477          PB1223  015601          R$CMD  = 140012
I$SETU= 000041          L$REV   002010 G        PBF2    011673          PB13    011354          R$DAT  = 140010
I$SRV  = 000041         L$SPC   002056 G        PBF3    011747          PB3     010540          R$FPS  = 140006
I$SUB  = 000041         L$SPCP  002020 G        PBF4    012043          PB4     010606          SDTCMD  032270
I$TST  = 000041         L$SPTP  002024 G        PBF5    012106          PB5     010660          SDT10   032270
I$UDC  = 100002         L$STA   002030 G        PBF6    012153          PB6     010751          SDT11   034342
J$JMP  = 000167         L$TEST  002114 G        PBF7    012250          PB7     011053          SERNBR  002316
LABEL0  034674          L$TIML  002014 G        PBF8    012347          PB8     011105          SER.NB  004202
LBN     002715          L$UNIT  002012 G        PBF9    012437          PB9     011141          SETUP   030266
LOCAL   002276          L10000  002274          PBSF0   015745          PF2     011357          SFBEGT  016611
LOE    = 040000 G       L10002  030522          PB0     010427          PLOC    002300          SFCYLT  017275
LOGUNI  002274          L10003  030532          PB1     010456          PNT    = 001000 G       SFDBBT  017057
LOPRGI  002530          L10004  030542          PB10    011203          POLLW   020746          SFDONT  016632
LOT    = 000010 G       L10005  030574          PB11    011245          POLLWT  020746          SFFCNT  017350
LSTCMD  002524          L10006  034602          PB11AP  013162          PRGNAM  002660          SFFCUT  017316
LSTCRN  002522          L10007  034676          PB11CR  012537          PRI    = 002000 G       SFRBBT  017262
LSTVCT  002526          MANBLD  024170          PB11EL  013061          PRI00  = 000000 G       SFRCBT  016777
L$ACP   002110 G        MAXDRV= 000004          PB11EN  012715          PRI01  = 000040 G       SFREVT  016656
L$APT   002036 G        MCDNBR  002324          PB11ES  013024          PRI02  = 000100 G       SFR1T   016700
L$AUT   002070 G        MDLNBR  002322          PB11GD  012774          PRI03  = 000140 G       SFR2T   016732
L$AUTO  030524 G        MOD1    002000 G        PB11OP  012607          PRI04  = 000200 G       SFTRYT  017217
L$CCP   002106 G        MRQDX1= 000007          PB11RD  013135          PRI05  = 000240 G       SFT0    010315
L$CLEA  030534 G        MRQDX3= 000023          PB11SD  013113          PRI06  = 000300 G       SFT1    010366
L$CO    002032 G        MSECA  = 007570         PB11ST  012661          PRI07  = 000340 G       SFXBBT  017137
L$DEPO  002011 G        MSEND   024112          PB11S0  013204          PRNTPK  026302          SF0     010107
L$DESC  002126 G        MSG     024140 G        PB11S1  013231          PS0    = 000000         SF1     010156
L$DESP  002076 G        MSGDAT  024142          PB11S2  013263          PS7    = 000340         SF100   010217
L$DEVP  002060 G        MSGLEN= 000014          PB11S3  013321          PTBL    002302          SIZDRV  023700
L$DISP  002124 G        MSGNBR= 170000          PB11S4  013356          QFDAT   016254          SIZEND  023710
L$DLY   002116 G        MSIN    024072          PB11S5  013412          QFSER   016535          SIZEXI  023754 G
L$DTP   002040 G        MSWAIT  024066          PB11S6  013441          QFUIT   016177          SIZIN   023630
L$DTYP  002034 G        NEXT    030274          PB11S9  013466          QNBRA   031652          SIZLOP  023466 G
L$DU    030544 G        OCTASC  025720          PB11W0  013531          QNBR0   031464          SIZNON  023456
L$DUT   002072 G        OP.ABR= 000006          PB11W1  013615          QNBR7   031572          SIZRD   023674
L$DVTY  002160 G        OP.DD  = 000001          PB12    015716          QSTN    031436          SIZSET  023402
L$EF    002052 G        OP.ELP= 000003          PB1201  013706          QUESTI= 000001          SIZWT   023362
```

SCQ 0075

Symbol table

```
SPCL    034252          TBQ18   006535          TYPASC  016100          T$TEST= 000001          UNIT    002312
SPECL = 000006          TBQ19   006554          TYPDUP  026114          T$TSTM= 177777          UNKWN   034556
SP2INT  021642          TBQ2    006076          TYPE   = 177760          T$TSTS= 000001          UNTDSZ= 000002
SP3INT  021732          TBQ20   006607          T$ARGC= 000001          T$$AUT= 010003          UNTFLG  002320
SP4INT  022012          TBQ21   006637          T$CODE= 001004          T$$CLE= 010004          UNT.NB  005403
STDALN= 000001          TBQ22   006671          T$ERRN= 000000          T$$DU = 010005          VECTOR  002310
SVCGBL= 000000          TBQ23   006704          T$EXCP= 000000          T$$HAR= 010007          VEC.AD  004135
SVCINS= 177777          TBQ24   006717          T$FLAG= 000041          T$$HW = 010000          WARNIN  004260
SVCSUB= 177777          TBQ25   006732          T$FREE= ****** GX       T$$INI= 010002          WRNGST  022160
SVCTAG= 177777          TBQ26   006745          T$GMAN= 000000          T$$PRO= 010001          W$CMD = 140022
SVCTST= 177777          TBQ28   006757          T$HILI= 030071          T$$TES= 010006          W$DAT = 140020
S$LSYM= 010000          TBQ29   007007          T$LAST= 000001          T1      030576 G        W$FPL = 140004
S$$BUG  024134          TBQ3    006120          T$LOLI= 000001          UAM   = 000200 G        XBN     002667
S$$FLA  024136          TBQ30   007040          T$LSYM= 010000          UIN     002326          X$ALWA= 000000
S$$RTI  024050 G        TBQ31   007066          T$LTNO= 000001          UITADR  002304          X$FALS= 000040
S$$UDC  024010 G        TBQ32   007130          T$NEST= 177777          UITDF   004020          X$OFFS= 000400
S$$UDI  024026          TBQ4    006142          T$NS0 = 000000          UITLOC  024700          X$TRUE= 000020
TBLBLD  024730          TBQ5    006164          T$NS1 = 000004          UITOTH= 000010          $2      030456
TBQ0    005767          TBQ6    006206          T$PTHV= ****** GX       UITSIZ= 000102          $3      030476
TBQ1    006054          TBQ7    006230          T$PTNU= 000000          UIT0    003000          $4      030512
TBQ10   006316          TBQ8    006252          T$SAVL= 177777          UIT1    003102          .A.DEF= 000040
TBQ11   006341          TBQ9    006274          T$SEGL= 177777          UIT2    003204          .A.FAT= 000120
TBQ12   006370          TERM    032626          T$SIZE= ****** GX       UIT3    003306          .A.INF= 000060
TBQ13   006427          TERMIN= 000004          T$SUBN= 000000          UIT4    003410          .A.QUE= 000020
TBQ14   006441          TIMOUT  022120          T$TAGL= 177777          UIT5    003512          .A.TER= 000100
TBQ15   006460          TNBRA   032740          T$TAGN= 010010          UIT6    003614          .A.TYP= 000020
TBQ16   006471          TNBR12  032654          T$TEMP= 000000          UIT7    003716          .B.SPL= 000140
TBQ17   006516          TNBR13  032706

. ABS.  034702    000   (RW,I,GBL,ABS,OVR)
        000000    001   (RW,I,LCL,REL,CON)
Errors detected:  0

*** Assembler statistics


Work  file  reads: 363
Work  file writes: 363
Size of work file: 39520 Words  ( 155 Pages)
Size of core pool: 19402 Words  ( 74 Pages)
Operating  system: RSX-11M/PLUS (Under VAX/VMS)

Elapsed time: 00:07:12.15
ZRQCB1,ZRQCB1.LST/CR/-SP=SVC35R.MLB/ML,ZRQCB1.MAC
```