```
.REM %(

                    IDENTIFICATION
                     --- -- -----

PRODUCT CODE:   AC FNFAA MC

PRODUCT NAME:   CZRQFA0  RQDX3 RX33 FLP FRMTR

PRODUCT DATE:   JUNE 6,  1986

MAINTAINER:     DIAGNOSTIC ENGINEERING

AUTHOR:         Richard Dietz
```

TABLE OF CONTENTS

1. ABSTRACT - What is it?
2. How to run it?
2.1 Hardware Requirements
2.2 Software Requirements
2.3 Questions asked and their answers
2.3.1 Hardware Questions from diagnostic software
2.3.2 Manual Questions from controller firmware
2.3.3 UIT tables
2.4 Program messages and format completion
2.5 Execution time
3. Errors
4. Program design and flow
5. Modification of UIT for additional drives
6. GLOSSARY
7. BIBLIOGRAPHY
8. REVISION HISTORY

1.0 ABSTRACT

This formatter was written to format RX33 floppy dr'ves attached to the RQDX3 disk controller. All RX33 media must be formated first before being used by the disk controller. Once the media is formatted than the diskette can be brought onl'ne through MSCP protocol or by an operating system.

This RX33 format utility w'll only work on an RQDX3 disk controller and will never work on an RQDX1 or RQDX2. This formatter uses the DUP protocol to answer questions asked by the format program in the microcode. The actual routine that does the formatting exists in the microcode. Th's ut'lity just passes information back and forth to the controller local format routine.

2.0 HOW TO RUN IT?

2.1 HARDWARE REQUIREMENTS

An RQDX3 disk controller and a RX33 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. It is also possible to run the formatter under APT.

2.3 QUESTIONS ASKED AND THEIR ANSWERS
2.3.1 HARDWARE QUESTIONS FROM DIAGNOSTIC SOFTWARE

The diagnostic 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.

Typical Diagnost'c Script:

```
boot up XXDP
.RUN ZRQF??
ZRQFA0.BIN

DRSXM-A0
ZRQF-A-0
RQDX3 RX33 Floppy Format utility
Unit is RX33
Restart Address is 141656
DR>START
```

```
Change HW ?  Y
# Units ?  1

IP Address 172150 ? <rtn>
Vector Address 154 ? <rtn>
Logical Drive (0 255) 1 ? <rtn>
```

2.3.2 MANUAL QUESTIONS ASKED

These questions where installed mainly to protect the person trying to format a diskette on the same drive as their boot media. If the drive doing the formatting is not the boot drive then please ignore the warnings.

Completion report:

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

FCT was not used
Format Completed

Do you want to format another diskette?

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

RQDX DRIVE xxxx finished.
pass aborted for this unit
ZRQC EOP 1
    0 Cumulative errors
```

Note that the pass is aborted for the unit. It doesn't mean that it failed. It just means that the test unit is done its sequence.

2.5 EXECUTION TIME

The execution time for this diagnostic is fairly short. The floppies only take 3 minutes to format.

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. This could happen if you answered the questions wrong.
                        Example you answer UNIT X and un't X was a winchester instead of
                        an RX33.

                1,HRD0  ;Fatal DUP type returned
                        Error with Format program check detailed error message more then
                        likely this will be a drive error or dr've 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 offl'ne,wr'te
                        protected, RX50 instead of an RDxx, power plug us loose, jumpers are
                        wrong.

                2,DF3   ;Can't do remote programs"
                        Wrong controller or bad microcode controller error.

                3,SFT0  ;"already active  will do an ABORT cmd"
                        Wrong controller or bad microcode controller error. The controller
                        was expected to be in an 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 0,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 't is
                        working very slow or is stuck. Leave the program running for a
                        couple minutes. If this message repeats then the drive is l'kely
                        broken.
```

```
                        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 ring and 1 response ring. For every command send there is expected 1 response. If the command sent times out a "Get D'JST Status" command 's 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 'nitialized 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 formatt'ng all the logical units.

5.0 GLOSSARY

ZRQFa0 follows the module name format described in the XXDP Programmer's Guide.

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

--F-- 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.

---a- 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:[diaglib.xxdp])

                                 8.0  REVISION HISTORY

                                Revision A.0
                                        Diagnostic created for PDP 11/53 first volume
                                sh'p with the RX33.
                        )%
```

```
                                .MCALL SVC
000000                          SVC
000000                          .ENABLE ABS,AMA
                000052          .=52
000052  010000                     .word       b't12            ;setup for extended XXDP monitor
                002000          .=2000
002000                          BGNMOD MOD1
002000                          POINTER BGNDU,BGNCLN,BGNPROT,BGNSETUP
002000                          HEADER ZRQF,A,0,600,0
002122                          DISPATCH 1
002126                          DESCRIPT <RQDX3 RX33 Format Utility>
002160                          DEVTYPE <RX33 *** Answer "Y" to "Change HW (L) ?" ***>

```

```
  313 002236                        BGNHW DFPTBL
  314 002240  172150                      .WORD   172150          ;IP address
  315 002242  000154                      .WORD   154             ;Vector address
  316 002244  000001                      .WORD   1               ;un't one as defualt drive
  317 002246                        ENDHW
  318
```

```
320 002246                          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
                        .sbttl L'terals

                        ;+
                        ; Mask values to mask out specified flags
                        ;-
000010                          UITothr = 10            ;UIT other
                                                        ;if UIT doesn't exist

                        ;+
                        ; Misc.
                        ;-
000004                          MaxDrv  = 4             ;Maximum Number of drives
000002                          DUP.id  = bit1          ;DUP connection ID
000007                          Mrqdx1  = 7.            ;model number for RQDX1
000023                          Mrqdx3  = 19.           ;model number for RQDX3
000001                          stdaln  = bit0          ;stand-alone modifier
000367                          retry   = 367           ;Number of retries UDC

                        ;+
                        ; Opcodes for DUP commands
                        ;-
000001                          op.gds  = 1
000006                          op.abrt = 6
000004                          op.sen  = 4
000005                          op.rec  = 5
000003                          op.elp  = 3
000002                          op.esp  = 2
000200                          op.end  = 200

                        ;+
                        ; Message type masks
                        ;-
000001                          Question = 1
000002                          DefQuest = 2
000003                          inform   = 3
000004                          terminat = 4
000005                          ftlerr   = 5
000006                          specl    = 6

177760                          type    = 177760
170000                          msgnbr  = 170000

                        ;+
                        ;Auto sizer literals
                        ;

                        ; Interrupt Service Routines and Pr'or'ty Levels
```

Literals

```
                                ;
100002          i$udc   =       100002          ; Pointer to UDC interrupt handler
100006          i$clk   =       100006          ; Pointer to Clock interrupt handler
100016          i$sec   =       100016          ; Pointer to Sector Done Interrupt handler
000000          ps0     =       0               ; Allow Any Interrupts
000340          ps7     =       340             ; Inhibit Interrupts
                                ;
                                ; CSRs
                                ;
140002          rw$pll  =       140002
140004          w$fpl   =       140004
140006          r$fps   =       140006
140010          r$dat   =       140010
140012          r$cmd   =       140012
140020          w$dat   =       140020
140022          w$cmd   =       140022
                                ;
                                ; RECEIVE DATA ASCII reply message types:
                                ;
000020          .a.typ  =       20                      ; ASCII Message Type Multiplier
000020          .a.que  =       1*.a.typ                ; Question
000040          .a.def  =       2*.a.typ                ; Default question
000060          .a.inf  =       3*.a.typ                ; Information
000100          .a.ter  =       4*.a.typ                ; Termination
000120          .a.fat  =       5*.a.typ                ; Fatal error
                                ;
                                ; RECEIVE DATA binary message types.
                                ;
000140          .b.spl  =       6*.a.typ                ; Special
                                ;
                                ; Status Codes returned by SIZER (Success is zero)
                                ;
000001          erudon          =       1               ; UDC Never Done
000002          eruint          =       2               ; UDC Never Interrupted
000003          ersek0          =       3               ; Couldn't Restore to Cyl 0
                                ;
                                ; UDC Commands
                                ;
000000          op.res          =       0               ; Reset 9224
000001          op.dd           =       1               ; Deselect Drive
000003          op.rd           =       3               ; Restore Drive
000005          op.si1          =       5               ; Step In One Cylinder
000007          op.so1          =       7               ; Step Out One Cylinder
000044          op.srd          =       44              ; Select Winchester Drive
000054          op.srx          =       54              ; Select Floppy Drive
000100          op.srp          =       100             ; Set Register Pointer
000300          rd.mode         =       300             ; RD Mode
                                ;
                                ;
```

Macro Definitions

```
                                .sbttl  Macro Definitions


                                ;+
                                ;       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
```

Macro Definitions

```
                        ;+
                        ;       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
                                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 endcode and status.
                                .nlist
                        .ENDM
```

Macro Definitions

```
                        ;+
                        ;       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,Sb,tcnt
                                .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

                                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 ard status.
                                .nlist
                        .ENDM
```

Macro Definit'ons

```
                        ;+
                        ;       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 'f 't 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 describtor
                                clr     cmdpak+22
                                clr     cmdpak+24
                                clr     cmdpak+26
                                clr     cmdpak+30
                                clr     cmdpak+32

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

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

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

Macro Definitions

```
614
615
616                             ;+
617                             ;       Execute a Execute Local Program command and the check the response.
618                             ;
619
620
621                             .MACRO  EXLCPRG Enamadr                 ;Execute a Send Data command
622                             B=B+1                                   ;'ncrement the CRN number
623                             elptmp  \B,Enamadr                      ;call variable A,B as 'f it where a number (.)
624                             .ENDM
625
626                             .MACRO  ELPTMP  B,Enamadr
627                                     .list
628                             ELP'B:  bit     #bit15,cmdrng+2         ;test ownership of ring make sure we own 't
629                                     bne     ELP'B                   ;if we don't own it wait until we do
630                                     mov     #22,cmdlen              ;load lenght of packet to be send
631                                     movb    #0,cmdlen+2             ;load msg type and credit
632                                     movb    #dup.id,cmdlen+3        ;load DUP connection ID
633                                     inc     cmdpak                  ;load new CRN
634                                     clr     cmdpak+2
635                                     clr     cmdpak+4
636                                     clr     cmdpak+6
637                                     mov     #op.elp,cmdpak+10       ;load up opcode
638                                     mov     #stdaln,cmdpak+12       ;stand alone modifier
639                                     mov     #6,r0                   ;6 letters transfer
640                                     mov     #cmdpak+14,r1           ;starting address to place program name
641                                     mov     #Enamadr,r2             ;start of Program Name
642                             rfdj'B: movb    (r2)+,(r1)+             ;add 2 to bycnt then store
643                                     sob     r0,rfdj'B
644
645                                     mov     #RFD'B,@vector          ;New vector place
646                                     mov     #rsppak,rsprng          ;load response packet area into ring
647                                     mov     #cmdpak,cmdrng          ;load command packet area into ring
648                                     mov     #140000,RSPRNG+2        ;Port ownership bit.
649                                     mov     #bit15,CMDRNG+2
650                                     jsr     pc,POLLWT               ;Go to poll and wait routine.
651
652                             ;*************************************************************************
653
654                                 RFD'B:                              ;Intr to here.
655                                     add     #6,sp                   ;fix stack for interrupt (4), pollwt subrtn (2)
656                                     mov     #intsrv,@vector         ;Change vector
657                                     jsr     pc,RSPCHK               ;Go to routine that will check on
658                                                                     ;the response recvd from the mut.
659                                                                     ;it will check the cmd ref
660                                                                     ;num, the endcode and status.
661                                     .nlist
662                             .ENDM
663
664
```

```
                                .sbttl  Word & Buffer defint'ons

002246  000000          LOGUNIT: .WORD                  ;logun't number
002250  000000          LOCAL:   .WORD                  ;
002252  000000          PLOC:    .WORD                  ;p table address
002254  000000          ptbl:    .WORD                  ;p table address
00225r  000000          UITadr:  .word
002260  000000          BOOT:    .word                  ;bootable media

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

002262  000000          IPreg:   .WORD   0              ;Address of the SA and IP registers
002264  000000          Vector:  .word   0
002266  000000          Unit:    .word   0              ;unit number
002270  000000          UNTflgs: .word   0              ;flags, bit15 =auto mode, bit14 ="I'm sure bit"
                                                        ;bit13 =unknown model number,bit12 -park heads only
002272  000000          mdlnbr:  .word   0              ;model number of the controller as returned in step 4
002274  000000          mcdnbr:  .word   0              ;micorcode number of the controller as returned in step 4

002276                  RSP1:    .BLKW   2              ;Response packet length
002302                  RSPPAK:  .BLKW   30.            ;Response packet
002376                  CMDLEN:  .BLKW   2              ;Command packet length
002402                  CMDPAK:  .BLKW   20.            ;Command packet

002452  000000          CINTR:   .WORD   0              ;Command interrupt indicator
002454  000000          RINTR:   .WORD   0              ;Response interrupt indicator
002456  002302          RSPRNG:  .word   rsppak         ;Message ring
002460  140000                   .word   140000
002462  002402          CMDRNG:  .word   cmdpak         ;Command ring
002464  100000                   .word   100000
002466  177777                   .WORD   -1

002470  000000          LSTCRN:  .word   0              ;storage for unreturned command CRN
002472  000000          LSTCMD:  .word   0              ;storage for unreturned command opcode
002474  000000          LSTVCT:  .word   0              ;storage for unreturned command intterupt vector address
002476  000000          LOPRGI:  .word   0              ;Low word of the progress indicator
002500  000000          HIPRGI:  .word   0              ;High word of progress indicator

                        .nlist bin                      ;data area
002502  DATARE: .asciz  /%A12345678901234567890123456789012345678901234567890123456789012345678901234567890/
                .even
002626  PRGnam: .ascii  /FORMAT/          ;address of local format program name
002634           .byte   0                 ;null for asciz
                .even
                                .list bin
```

DISK PARAMETER QUESTIONS

```
                                .sbttl DISK PARAMETER QUESTIONS
        .nlist  bin

        ;*
        ; P table Questions
        ;-

002636  IP.adr:  .ASCIZ  /IP Address/
002651  vec.adr: .ASCIZ  /Vector Address/
002670  prk.hds: .ASCIZ  /Just park the heads/
002714  drv.nbr: .ASCIZ  /Logical Drive (0-255)/
002742  ser.nbr: .ASCIZ  /Dr ve Serial Number(1-32000)/
002777  auto.md: .ASCIZ  /Auto Format Mode/
003020  warning: .ASCIZ /***** WARNING all the data on this drive will be DESTROYED ***/
003117           .byte  0

003120  do.cont: .ASCIZ  /Proceed to format the drive/

003154  DrvTxa: .asciz /%N%AUIT#  Drive Name%N/
003203  DrvTxb: .asciz /%A___ _____  ----------------------------------------%N/
003277  DrvTx0: .asciz /%A  0:  RD51                                          %N/
003373  DrvTx1: .asciz /%A  1:  RD52 part # 30-21721-02 (1 light on front panel) %N/
003467  DrvTx2: .asciz /%A  2:  RD52 part # 30-23227-02 (2 lights on front panel)%N/
003563  DrvTx3: .asciz /%A  3:  RD53                                          %N/
003657  DrvTx4: .asciz /%A  4:  RD31                                          %N/
003753  DrvTx5: .asciz /%A  5:  RD54                                          %N/
004047  DrvTx6: .asciz /%A  6:                                                %N/
004142  DrvTx7: .asciz /%A  7:                                                %N/
004235  DrvTxc: .asciz /%A 10:                                                %N/

004331  ASMSG1: .ASCIZ  /%N%AUnt  Cyl#  UIT#     Drive Name/
004374  ASMSG2: .ASCIZ  /%A  %D1%A  %D4%A  /
004417  ASMSG3: .ASCIZ  /%N%AAUTOSIZER RETURNED FAILURE STATUS CODE %D1%A:/
004501  ASMSG4: .ASCIZ  /%N%A    CONTROLLER CHIP NEVER WENT DONE/
004551  ASMSG5: .ASCIZ  /%N%A    CONTROLLER CHIP NEVER INTERRUPTED/
004623  ASMSG6: .ASCIZ  /%N%A    SEEK FAILED/
004647  ASMSG7: .ASCIZ  /%N%A    UNIT %D1%A NONEXISTENT/
004706  ASMSG8: .ASCIZ  /%N%A    UNIT %D1%A RX50 FLOPPY (UNFORMATABLE)/
004764  ASMSG9: .ASCIZ  /%N%A    UNIT %D1%A RX33 FLOPPY (FORMATABLE)/
005040  ASMSGT: .ASCIZ  /%N/
005043  parkdrv: .ASCIZ  /%N%APLEASE wait .... parking disk heads./

005114  Unt.nbr: .ASCIZ  /Enter Unit Identifier Table (UIT)/
005156  ask.prg: .ASCIZ  /What local program do you want to run/
005224  ask.xbn: .ASCIZ  /Enter XBN size in decimal (upto 10 digits)/
005277  ask.dbn: .ASCIZ  /Enter DBN size in decimal (upto 10 digits)/
005352  ask.lbn: .ASCIZ  /Enter LBN size in decimal (upto 10 digits)/
005425  ask.rbn: .ASCIZ  /Enter RBN size in decimal (upto 10 digits)/


005500  bot.dev: .ASCII <15><12>/WARNING - Remove boot diskette if in drive to be formatted and/
005600           .ASCII <15><12>/          insert a diskette to be formatted./
005656           .ASCIZ <15><12>/WARNING - All data on drive will be DESTROYED, do you want to continue?/
005770  bot.rep: .ASCIZ /If boot drive, reinsert boot diskette & press <RETURN>./
006060  bot.con: .ASCIZ <15><12>/Do you want to format another diskette?/

        ; Top of Unit Information table (UIT)
```

DISK PARAMETER QUESTIONS

```
    774
    775 006132  TBQ0:    .ASCIZ  /XBN size (lo wrd) XBN s'ze = 5*(1+sectors_per_track)/
    776 006217  TBQ1:    .ASCIZ  /XBN size (hi wrd)/
    777 006241  TBQ2:    .ASCIZ  /DBN size (lo wrd)/
    778 006263  TBQ3:    .ASCIZ  /DBN size (hi wrd)/
    779 006305  TBQ4:    .ASCIZ  /LBN size (lo wrd)/
    780 006327  TBQ5:    .ASCIZ  /LBN size (hi wrd)/
    781 006351  TBQ6:    .ASCIZ  /RBN size (lo wrd)/
    782 006373  TBQ7:    .ASCIZ  /RBN size (hi wrd)/
    783 006415  TBQ8:    .ASCIZ  /Sectors per track/
    784 006437  TBQ9:    .ASCIZ  /Surfaces per unit/
    785 006461  TBQ10:   .ASCIZ  /Cylinders per unit/
    786 006504  TBQ11:   .ASCIZ  /Write precomp cylinder/
    787 006533  TBQ12:   .ASCIZ  /Reduce write current cylinder /
    788 006572  TBQ13:   .ASCIZ  /Seek Rate/
    789 006604  TBQ14:   .ASCIZ  /Use CRC or ECC/
    790 006623  TBQ15:   .ASCIZ  /RCT Size/
    791 006634  TBQ16:   .ASCIZ  /Number of RCT cop'es/
    792 006661  TBQ17:   .ASCIZ  /Media (lo wrd)/
    793 006700  TBQ18:   .ASCIZ  /Media (hi wrd)/
    794 006717  TBQ19:   .ASCIZ  /Sector Interleave (n-to-1)/
    795 006752  TBQ20:   .ASCIZ  /Surface to Surface Skew/
    796 007002  TBQ21:   .ASCIZ  /Cylinder to Cylinder Skew/
    797 007034  TBQ22:   .ASCIZ  /Gap size 0/
    798 007047  TBQ23:   .ASCIZ  /Gap size 1/
    799 007062  TBQ24:   .ASCIZ  /Gap size 2/
    800 007075  TBQ25:   .ASCIZ  /Gap size 3/
    801 007110  TBQ26:   .ASCIZ  /Sync size/
    802 007122  TBQ28:   .ASCIZ  /MSCP cylinders per Unit/
    803 007152  TBQ29:   .ASCIZ  /MSCP Groups per Cylinder/
    804 007203  TBQ30:   .ASCIZ  /MSCP Tracks per Group/
    805 007231  TBQ31:   .ASCIZ  /Max allowed bad spots per surface/
    806 007273  TBQ32:   .ASCIZ  /Bad spot tolerance (bytes)/
    807
    808 007326  DF1:     .ASCIZ  /Controller Initialization Timeout/
    809 007370  DF2:     .ASCIZ  /Controller never advanced to next step/
    810 007437  DF3:     .ASCIZ  /Controller can not execute local programs or non STD DUP dialog program/
    811 007547  DF4:     .ASCIZ  /NXM Trap at controllers IP address/
    812         ;DF10:   .ASCIZ  /No Interrupt occurred after SA polled/
    813 007612  DF11:    .ASCIZ  /Bad Response Packet returned/
    814 007647  DF12:    .ASCIZ  /Fatal SA error ctlr offline/
    815 007703  DF13:    .ASCIZ  /No progress shown after a cmd had timed out/
    816 007757  DF14:    .ASCIZ  /GET DUST CMD time_out after another CMD time_out/
    817 010040  DF15:    .ASCIZ  /%N%AFatal error was reported when running local program/
    818 010130  DF16:    .ASCIZ  /%N%AA Special was reported when running local program don't know how to handle it/
    819 010252  SF0:     .ASCII  /DUP protocol Error, unexpected message/
    820 010320          .ASCIZ  <15><12>/Check unit, it is probable not an RX33/
    821 010371  SF1:     .ASCIZ  /%N%ASYSTEM is NOT in manual mode/
    822 010432  SF100:   .ASCIZ  /Unexpected or delayed Controller Interrupt/
    823 010505  HRD0:    .ASCIZ  /Fatal Format error/
    824 010530  SFT0:    .ASCIZ  /Controller in an unexpected ACTIVE state/
    825 010601  SFT1:    .ASCIZ  /Wrong Model Number on controller/
    826 010642  PB0:     .ASCIZ  /%N%AModel # listed %O6/
    827 010671  PB1:     .ASCIZ  /%N%AExpected SA step bit %O6%1,Received in SA %O6/
    828 010753  PB3:     .ASCIZ  /%N%AAsking for Format Parameter table/
    829 011021  PB4:     .ASCIZ  /%N%AReceived valid Format Parameter table/
    830 011073  PB5:     .ASCIZ  /%N%AOn UNIT %O6%A, %O6 Bad Blks were found during Format/
```

DISK PARAMETER QUESTIONS

```
831 011164  PB6:     .ASCIZ  /%N%AOn UNIT %06%A, %06 Bad Blks were found during Verify pass %06/
832 011266  PB7:     .ASCIZ  /%N%ADUP Message Type: %06/
833 011320  PB8:     .ASCIZ  /%N%ADUP message number: %06/
834 011354  PB9:     .ASCIZ  /%N%AMSCP Controller model # : %D3/
835 011416  PB10:    .ASCIZ  /%N%A    Microcode version # : %D3/
836 011460  PB11:    .ASCIZ  /%N%AController is IDLE when it should be ACTIVE running format program/
837 011567  PB13:    .ASCIZ  /%N/
838 011572  PF2:     .ASCIZ  /%N%N%AFinished local program without procedure error/
839 011657  PBF0:    .ASCIZ  /%N%AFormat Parameter table entry at byte %06%N%Ais out of range/
840 011757  PBF1:    .ASCIZ  /%N%AFormat Parameter table entry at byte %06%N%Ais incompatible with entry at byte %06/
841 012106  PBF2:    .ASCIZ  /%N%AUNIT %06%A does not exist on controller/
842 012162  PBF3:    .ASCIZ  /%N%AUNIT %06%A does exist but doesn't respond on controller/
843 012256  PBF4:    .ASCIZ  /%N%AUNIT %06%A 's write protected /
844 012321  PBF5:    .ASCIZ  /%N%AWrite Fault detected on UNIT %06/
845 012366  PBF6:    .ASCIZ  /%N%AAttempt to step hd %03%A at cyl %03%A failed on UNIT %06/
846 012463  PBF7:    .ASCIZ  /%N%AAttempt to format hd %03%A at cyl %03%A failed on UNIT %06/
847 012562  PBF8:    .ASCIZ  /%N%ATo many Bad Blocks                 total Bad Blocks %06/
848 012652  PBF9:    .ASCIZ  /%N%ADisk Controller model : %D3/
849 012712  PBF10:   .ASCIZ  /%N%A    Microcode version : %D3/
850 012752  PB11crn: .ASCIZ  /%N%AExpected CRN %06%A,Received CRN %06/
851 013022  PB11op:  .ASCIZ  /%N%ACMDpkt Opcode %06%A,RSPpkt Opcode %06/
852 013074  PB11sts: .ASCIZ  /%N%AResponse pkt status %06/
853 013130  PB11end: .ASCIZ  /%N%ANo end bit(200) in response packet endcode/
854 013207  PB11GDS: .ASCIZ  /%N%AGet Dust Status cmd/
855 013237  PB11ESP: .ASCIZ  /%N%AExecute Supplied Prg cmd/
856 013274  PB11ELP: .ASCIZ  /%N%AExecute Local Prg cmd/
857 013326  PB11SD:  .ASCIZ  /%N%ASend Data cmd/
858 013350  PB11RD:  .ASCIZ  /%N%AReceive Data cmd/
859 013375  PB11AP:  .ASCIZ  /%N%AAbort Prg cmd/
860 013417  pb11s0:  .ASCIZ  /%N%Asts:  successful/
861 013444  pb11s1:  .ASCIZ  /%N%Asts:  Invalid Command/
862 013476  pb11s2:  .ASCIZ  /%N%Asts:  No Region Available/
863 013534  pb11s3:  .ASCIZ  /%N%Asts:  No Region Suitable/
864 013571  pb11s4:  .ASCIZ  /%N%Asts:  Program Not Known/
865 013625  pb11s5:  .ASCIZ  /%N%Asts:  Load Failure/
866 013654  pb11s6:  .ASCIZ  /%N%Asts:  Standalone/
867 013701  pb11s9:  .ASCIZ  /%N%Asts:  Host Buffer Access error/
868 013744  pb11w0:  .ASCIZ  /%N%AUnknown command OPCODE received in timeout loop/
869 014030  pb11w1:  .ASCIZ  /%N%AUnknown command CRN received in command timeout loop/
870 014121  pb1201:  .ASCIZ  /%N%ASA er: Envelope\packet Read (parity or timeout)/
871 014205  pb1202:  .ASCIZ  /%N%ASA er: Envelope\packet Write (parity or timeout)/
872 014272  pb1203:  .ASCIZ  /%N%ASA er: Controller ROM and RAM parity/
873 014343  pb1204:  .ASCIZ  /%N%ASA er: Controller RAM parity/
874 014404  pb1205:  .ASCIZ  /%N%ASA er: Controller ROM parity/
875 014445  pb1206:  .ASCIZ  /%N%ASA er: Queue Read (parity or timeout)/
876 014517  pb1207:  .ASCIZ  /%N%ASA er: Queue Write (parity or timeout)/
877 014572  pb1208:  .ASCIZ  /%N%ASA er: Interrupt Master/
878 014626  pb1209:  .ASCIZ  /%N%ASA er: Host Access Timeout (higher level protocol dependent)/
879 014727  pb1210:  .ASCIZ  /%N%ASA er: Credit Limit Exceeded /
880 014771  pb1211:  .ASCIZ  /%N%ASA er: Bus Master Error/
881 015025  pb1212:  .ASCIZ  /%N%ASA er: Diagnostic Controller Fatal error/
882 015102  pb1213:  .ASCIZ  /%N%ASA er: Instruction Loop Timeout/
883 015146  pb1214:  .ASCIZ  /%N%ASA er: Invalid Connection Identifier/
884 015217  pb1215:  .ASCIZ  /%N%ASA er: Interrupt Write Error/
885 015260  pb1216:  .ASCIZ  /%N%ASA er: MAINTENANCE READ\WRITE Inval'd Region Identifier/
886 015354  pb1217:  .ASCIZ  /%N%ASA er: MAINTENANCE WRITE Load to non-loadable controller/
887 015451  pb1218:  .ASCIZ  /%N%ASA er: Controller RAM error (non-parity)/
```

```
    888 015526  pb1219: .ASCIZ  /%N%ASA er: INIT sequence error/
    889 015565  pb1220: .ASCIZ  /%N%ASA er: High level protocol 'ncompatib'l'ty error/
    890 015652  pb1221: .ASCIZ  /%N%ASA er: Purge\poll hardware failure/
    891 015721  pb1222: .ASCIZ  /%N%ASA er: Mapping Register read error (parity or timeout)/
    892 016014  pb1223: .ASCIZ  /%N%ASA er: Attempt to set port data transfer mapping when option not present/
    893 016131  PB12:   .ASCIZ  /%N%ASA Value (oct) %06/
    894
    895 016160  PBsf0:  .ASCIZ  /%N%ADUP type %06%A message number %06/
    896 016226  DRPunt: .ASCIZ  /%N%N%ARQDX DRIVE %06%A is fin'shed/
    897 016271  TYPASC: .ASCIZ  /%N%APLEASE TYPE ANSWER to controller question or just <return>/
    898
    899         ;mmm
    900         ;
```

```
                        .sbttl  FORMAT Messages

                ; quer'es

016370  qfuit:  ;.byte  2,,,b.spl                ; Unit Info Table? (spl #2)
016370           .asciz  '%N%AEntering UIT%O2%A: on drive number %D3%N'
016445  qfdat:  ;.byte  0,,,a.que                ; Date? (que #0)
016445           .asciz  'Enter date <MM DD-YYYY>:'
016476  dfunt:  ;.byte  1,,,a.def                ; Unit? (def #1)
016476           .asciz  'Enter unit number to format <0>:'
016537  dfbad:  ;.byte  4,,,a.def                ; Use Bad? (def #4)
016537           .asciz  'Use existing bad block information <N>:'
016607  dfdwn:  ;.byte  5,,,a.def                ; Downline? (def #5)
016607           .asciz  'Use down line load <Y>:'
016637  dfcon:  ;.byte  6,,,a.def                ; Continue? (def #6)
016637           .asciz  'Continue if bad block information is inaccessible <N>:'
016726  qfser:  ;.byte  7,,,a.que                ; Serial #? (que #7)
016726           .asciz  'Enter non zero serial number <8-10 digits>:'

                ; Informational Messages

017002  sfbegt: ;.byte  0,,,a.inf                ; Begin (inf #0)
017002           .asciz  '%N%AFormat Begun'
017023  sfdont: ;.byte  1,,,a.inf                ; Complete (inf #1)
017023           .asciz  '%N%AFormat complete'
017047  sfrevt: ;.byte  2,,,a.inf                ; # of Revectored LBNs (inf #2)
017047           .asciz  '% Revectored LBNS'
017071  sfr1t:  ;.byte  3,,,a.inf                ; # of primary ... (inf #3)
017071           .asciz  '% Primary revectored LBNS'
017123  sfr2t:  ;.byte  4,,,a.inf                ; # of secondary ... (inf #4)
017123           .asciz  '% Secondary/tertiary revectored LBNS'
017170  sfrcbt: ;.byte  5,,,a.inf                ; # of Bad RCT blocks ... (inf #5)
017170           .asciz  '% Bad blocks in the RCT area due to data errors'
017250  sfdbbt: ;.byte  7,,,a.inf                ; # of Bad DBNs ... (inf #7)
017250           .asciz  '% Bad blocks in the DBN area due to data errors'
017330  sfxbbt: ;.byte  9,,,a.inf                ; # of Bad XBNs ... (inf #9)
017330           .asciz  '% Bad blocks in the XBN area due to data errors'
017410  sftryt: ;.byte  11,,,a.inf               ; # of Retries (inf #11)
017410           .asciz  '% Blocks retried on the check pass'
017453  sfrbbt: ;.byte  14,,,a.inf               ; # of Bad RBNs ... (inf #14)
017453           .asciz  '% Bad RBNS'
017466  sfcylt: ;.byte  15,,,a.inf               ; Formatting Cyl (inf #15)
017466           .asciz  'Formatting Cyl %'
```

```
                ; Successful Termination Messages
                
                    ;.byte      12...a.ter              ; Reformat Worked (ter #12)
017507  sffcut: .asciz  '%N%AFCT used successfully'
                
                        ;.byte  13...a.ter              ; Reconstruct Worked (ter #13)
017541  sffcnt: .ascii  '%N%AFCT was not used'
017565          .asciz  '%N%AFormat Completed'
                
                ; Error messages
                
017612  efstat: ;.byte  1...a.fat                ; Status Error (fat #1)
017612          .asciz  '%N%AGET STATUS failure'
                
017641  efsndt: ;.byte  2...a.fat                ; Send Error (fat #2)
017641          .asciz  '%N%AQ-PORT send error'
                
017667  efcmdt: ;.byte  3...a.fat                ; Command Error (fat #3)
017667          .asciz  '%N%AUnsuccessful command'
                
017720  efrcvt: ;.byte  4...a.fat                ; Receive Error (fat #4)
017720          .asciz  '%N%AQ-PORT receive error'
                
017751  efbust: ;.byte  5...a.fat                ; Bus Error (fat #5)
017751          .asciz  '%N%AQ-Bus I/O error'
                
017775  efinit: ;.byte  6...a.fat                ; Format Init Error (fat #6)
017775          .asciz  '%N%AFormatter initialization error'
                
020040  efnut:  ;.byte  7...a.fat                ; Unit nonexistent error (fat #7)
020040          .asciz  '%N%ANonexistent unit number'
                
020074  efdxft: ;.byte  8...a.fat                ; DBN/XBN Format error (fat #8)
020074          .asciz  '%N%ADBN/XBN format error (drive FORMAT command failed)'
                
020163  effcct: ;.byte  9...a.fat                ; FCT copies error (fat #9)
020163          .asciz  '%N%AFCT does not have enough good copies of each block'
                
020252  efsekt: ;.byte  10...a.fat               ; Seek error (fat #10)
020252          .asciz  '%N%ASEEK error'
                
020271  efrcct: ;.byte  11...a.fat               ; RCT copies error (fat #11)
020271          .asciz  '%N%ARCT does not have enough good copies of each block'
                
020360  eflbft: ;.byte  12...a.fat               ; LBN format error (fat #12)
020360          .ascii  '%N%ALBN format err (drv FORMAT cmd failed)'
020432          .asciz  '%N%AChk unit, RX50 is NOT formattable'
                
020500  effcwt: ;.byte  13...a.fat               ; FCT write error (fat #13)
020500          .asciz  '%N%AFCT write error (check write protect switch)'
                
020561  efrcrt: ;.byte  14...a.fat               ; RCT read error (fat #14)
020561          .asciz  '%N%ARCT read error'
                
020604  efrcwt: ;.byte  15...a.fat               ; RCT write error (fat #15)
020604          .asciz  '%N%ARCT write error'
```

M2

FORMAT Messages

```
1003
1004 020630  efrcft: ;.byte  16...a.fat              ; RCT full error (fat #16)
1005 020630          .asciz  '%N%ARCT full
1006
1007 020645  effcrt: ;.byte  17...a.fat              ; FCT read error (fat #17)
1008 020645          .asciz  '%N%AFCT read error'
1009
1010 020670  effcnt: ;.byte  18...a.fat              ; FCT nonexistent error (fat #18)
1011 020670          .asciz  '%N%AFCT nonexistent'
1012
1013 020714  effcdt: ;.byte  19...a.fat              ; FCT downline load error (fat #19)
1014 020714          .asciz  '%N%AFCT Down line load error'
1015
1016 020751  eftmot: ;.byte  20...a.fat              ; Drive timeout error (fat #20)
1017 020751          .asciz  '%N%ADrive init timeout'
1018
1019 021000  efillt: ;.byte  21...a.fat              ; Illegal response error (fat #21)
1020 021000          .asciz  '%N%AIllegal response to start-up question'
1021
1022 021052  efwart: ;.byte  22...a.fat              ; Head error (fat #22)
1023 021052          .asciz  '%N%AWARNING - possible head addressing problem - run diagnostics'
1024
1025 021153  efinpt: ;.byte  23...a.fat              ; Input error (fat #23)
1026 021153          .asciz  '%N%AINPUT Error '
1027
1028 021174  efmedt: ;.byte  24...a.fat              ; Media error (fat #24)
1029 021174          .asciz  '%N%AMedia degraded'
1030
1031 021217  efunrg: ;.byte  1...a.fat               ; Status Error (fat #1)
1032 021217          .asciz  '%N%AUnrecogonized drive'
1033
1034                         .list   bin
1035                         .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 go ng
                                ;The other course of the routine is to handled timeout commands. Th's  s
                                ;
                                ;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 wh'ch 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
o type and responding
                                ;) )<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<)(no,  LSTCRN = #) load TIMED OUT cmd Vector,Opcode,CRN goto
 top of routine
                                ;)<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<)if response was TIMED_OUT cmd clear LSTCRN and wa't for GET
DUST response
                                ;       )
                                ;       ) go to DUP dialog routine
                                ;*****************************************************************************


                                POLLWT:

021250                          

021250  106427  000140        pollw:  mtps    #140                    ;Drop cpu level to three.
021254  005777  161002                tst     @IPreg                  ;Tell mut to start polling.

021260  012700  000074                mov     #60.,r0                 ;outer wait loop
021264  012701  004000        1$:     mov     #4000,r1                ;       ;
021270  106427  000340        2$:     mtps    #340                    ;don't want interrupts while in other routines
021274  004737  024350                jsr     pc,BIT15T               ;       ;inner wait loop
021300                                BREAK                           ;check for control C
021302  106427  000140                mtps    #140                    ;turn on interrupts again after check
021306  077110                        sob     r1,2$                   ;       ;
021310  077013                        sob     r0,1$                   ;
                                                                      ;No response has been received
```

```
Global subroutines

   1094 021312  106427  000340                  mtps    #340                    ;don't want interrupts while setting up for cmd
   1095 021316  004737  024350                  jsr     pc,BIT15T               ;test SA make sure not a fatal error
   1096 021322  013700  002412                  mov     cmdpak+10,r0            ;get opcode
   1097 021326  022700  000001                  cmp     #op.gds,r0              ;if the command issued was a GETDUST STATUS and time
out big trouble
   1098 021332  001006                          bne     GDS0                    ;if not go do a GET DUST to find out what the situat
ion is
   1099 021334                                  ERRDF   12,df14                 ;type no 'terrupt after get dust status command cont
roller dead
   1100 021344  000137  030606                  jmp     dropunt                 ;drop unit and go on
   1101
   1102                                 ;GETDUST                                ;save timed out command information
   1103
   1104 021350  017737  160710  002474 GDS0:    mov     @vector,LSTVCT          ;store the vector address of timeout command
   1105 021356  013737  002402  002470          mov     cmdpak,LSTCRN           ;store the CRN of the timed out command
   1106 021364  013737  002412  002472          mov     cmdpak+10,LSTCMD        ;store the opcode of timed out command
   1107
   1108 021372  032737  100000  002464          bit     #bit15,cmdrng+2         ;test ownership of ring make sure we own  t
   1109 021400  001363                          bne     GDS0                    ;if we don't own it wait until we do
   1110 021402  012737  000016  002376          mov     #14.,cmdlen             ;load lenght of packet to be send
   1111 021410  112737  000000  002400          movb    #0,cmdlen+2             ;load msg type and credit
   1112 021416  112737  000002  002401          movb    #dup.id,cmdlen+3        ;load DUP connection ID
   1113 021424  005237  002402                  inc     cmdpak                  ;load new CRN
   1114 021430  005037  002404                  clr     cmdpak+2
   1115 021434  005037  002406                  clr     cmdpak+4
   1116 021440  005037  002410                  clr     cmdpak+6
   1117 021444  012737  000001  002412          mov     #op.gds,cmdpak+10       ;load up opcode
   1118 021452  005037  002414                  clr     cmdpak+12               ;no modifiers
   1119
   1120 021456  012777  021516  160600          mov     #RFD0,@vector           ;NEW VECTOR PLACE
   1121 021464  012737  002302  002456          mov     #rsppak,rsprng          ;load response packet area into ring
   1122 021472  012737  002402  002462          mov     #cmdpak,cmdrng          ;load command packet area into ring
   1123 021500  012737  140000  002460          mov     #140000,RSPRNG+2        ;PORT OWNERSHIP BIT.
   1124 021506  012737  100000  002464          mov     #bit15,CMDRNG+2
   1125 021514  000655                          br      POLLWT                  ;GO and wait for interrupt
   1126
   1127
   1128                                 ;+
   1129                                 ; There is only 3 ways out code.
   1130                                 ;If GETDUST resposne and TIMED_OUT cmd response was handled
   1131                                 ; if LSTCRN = 0 and RSPPAK+10 = OP.GDS+OP.END then
   1132                                 ; back to DUP dialog mode.
   1133                                 ;or
   1134                                 ; (TIMED_OUT cmd still hasn't returned but GETDUST has returned)
   1135                                 ; if LSTCRN = 0 and RSPPAK+10 = OP.GDS+OP.END then
   1136                                 ; check if idle or active. if idle then error
   1137                                 ; check for progress in progress indicator if no progress then error
   1138                                 ; load LSTVCT into @vector,LSTCRN into cmdpak, LSTCMD into cmdpak+10
   1139                                 ; set response ring ownership to Port Owned
   1140                                 ; jmp to pollwt.
   1141                                 ;or
   1142                                 ; (TIMED_OUT cmd response recieved before GETDUST response returned)
   1143                                 ; if LSTCRN = 0 and RSPPAK+10 not= OP.GDS+OP.END then
   1144                                 ; clear LSTCRN and
   1145                                 ; jmp to pollwt.
   1146                                 ;+
   1147
   1148 021516                          RFD0:                                   ;INTR TO HERE if GETDUST or TIMED OUT cmd
   1149 021516  106427  000340                  mtps    #340                    ;No interrupts please
   1150 021522  062706  000004                  add     #4,sp                   ;fix stack 4 for intrpt
```

Global subroutines

```
   1151 021526  013701  002402                  mov     cmdpak,r1               ;check command packet CRN
   1152 021532  013700  002302                  mov     rsppak,r0               ;check response packet CRN
   1153 021536  020001                          cmp     r0,r1                   ;Are they the SAME must be GETDUST cmd
   1154 021540  001103                          bne     3$                      ;if not it must be the TIMED_OUT cmd
   1155
   1156 021542  023727  002312  000201          cmp     rsppak+10,#op.gdu+op.end        ;it should be a GETDUST lets make sure
   1157 021550  001412                          beq     1$
   1158 021552                                  printf  #pb11w0                 ;unexpected cmd response in time out loop
   1159 021572  000137  030572                  jmp     unkwn                   ;error handler
   1160
   1161 021576  004737  023416          1$:     jsr     pc,RSPCHK               ;check the response
   1162 021602  005737  002470                  tst     LSTCRN                  ;see if timed out command was already recieved (lstc
rn = 0)
   1163 021606  001004                          bne     2$
   1164 021610  062706  000002                  add     #2,sp                   ;adjust stack for Timed Out cmd's initial call to PO
LLWT
   1165 021614  000137  026530                  jmp     DUPDLG                  ;if Timed out cmd was already received then goto DUP
 dialog mode
   1166
   1167 021620                          2$:                                     ;if Timed out command was not received already (LSTC
RN not= 0)
   1168 021620  132737  000010  002321          bitb    #bit3,rsppak+17         ;if server idle then error
   1169 021626  001010                          bne     1002$                   ;if not check for progress
   1170 021630                                  printf  #pb11                   ;controller idle when it should be active
   1171
   1172 021650  013700  002322          1002$:  mov     rsppak+20,r0            ;check for progress in progress indicator
   1173 021654  013701  002324                  mov     rsppak+22,r1
   1174 021660  020037  002476                  cmp     r0,loprgi               ;see if low word of progress indicator is the same a
s older value
   1175 021664  001007                          bne     1001$                   ;if it is then continue
   1176 021666  020137  002500                  cmp     r1,hiprgi               ;see if high vaule is the same
   1177 021672  001004                          bne     1001$
   1178 021674                                  ERRDF   11,DF15                 ;no progress shown after cmd timeout
   1179
   1180 021704  010037  002476          1001$:  mov     r0,loprgi               ;update progress indicator
   1181 021710  010137  002500                  mov     r1,hiprgi
   1182 021714  013737  002470  002402          mov     LSTCRN,cmdpak           ;move TIMED_OUT cmd CRN into cmd
   1183 021722  013737  002472  002412          mov     LSTCMD,cmdpak+10        ;move TIMED_OUT cmd Opcode into cmd
   1184 021730  013777  002474  160326          mov     LSTVCT,@vector          ;load TIMED_OUT cmd interrupt handler address into v
ector
   1185 021736  012737  140000  002460          mov     #140000,RSPRNG+2        ;Port owned
   1186 021744  000137  021250                  jmp     pollw                   ;wait for TIMED_OUT cmd response
   1187
   1188
   1189
   1190 021750  020037  002470          3$:     cmp     r0,LSTCRN               ;check the crn with the last CRN from the timeout co
mmand
   1191 021754  001412                          beq     4$                      ;
   1192 021756                                  printf  #pb11w1                 ;Unexpected cmd response in time out loop
   1193 021776  000137  030572                  jmp     unkwn                   ;error handler
   1194
   1195                                                                         ;Timed out command recieved but Get Dust Status is s
till in Queue
   1196 022002  013737  002470  002402  4$:     mov     LSTCRN,cmdpak           ;load timed out command values for RSPCHK routine
   1197 022010  013737  002472  002412          mov     LSTCMD,cmdpak+10        ;load timed out command values for RSPCHK routine
   1198 022016  005037  002470                  clr     LSTCRN                  ;if it is the timeout command clear LAST CRN registe
r
   1199 022022  004737  023416                  jsr     pc,RSPCHK               ;go check the command
   1200 022026  012737  140000  002460          mov     #140000,RSPRNG+2        ;PORT OWNERSHIP BIT.
   1201 022034  000137  021250                  jmp     POLLW                   ;go wait for GETDUST intterupt
```

```
                                ;*****************************************************************************
                                ;       HARD INITIALIZE
                                ; This routine hard initialize the disk controller so that DUP commands
                                ; can be issued. This routine is governed by the UQSSP spec.
                                ; Th's format starts by initializating
                                ;       r1 = ptable address
                                ;       r3 = step bit mask
                                ;       r4 = SA address

022040                          HRDINT:
022040  106427  000140                  mtps    #140
022044  013704  002262                  mov     ipreg,r4
022050  012703  004000                  mov     #bit11,r3               ;initialization step reg
022054  005024                  1$:     clr     (r4)+                   ; start initialaztion IP and put SA adrs into r4
022056                                  Break

022060  012700  177777                  mov     # 1,r0                 ;looking for step 1 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>
022064  004737  024350          4$:     jsr     pc,bit15T               ;Look for fatal SAreg error
022070  030314                          bit     r3,(r4)                 ;check sa reg for step 1        bit11 =0 >1
022072  001002                          bne     6$
022074  077005                          sob     r0,4$
022076  000560                          br      timout

022100  013700  002264          6$:     MOV     vector,r0               ;responding in step 1 <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<<<<<<<<
022104  000241                          clc
022106  006200                          asr     r0
022110  006200                          asr     r0
022112  052700  100200                  bis     #<bit15+bit7>,r0        ;= VECTOR/4
022116  013701  002264                  mov     vector,r1
022122  012721  022162                  mov     #sp2int,(r1)+           ;load up interrupt loacation into vector
022126  012711  000140                  mov     #140,(r1)               ;lower the priority
022132  010014                          mov     r0,(r4)                 ; Enable INTERRUPTS,set 1 cmd rng & 1 rsp rng

022134  006303                          asl     r3                      ;next step
022136  012700  177777                  mov     #-1,r0                  ;looking for step 2 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>
022142  106427  000340          8$:     mtps    #340
022146                                  break                           ;waste some time
022150                                  break                           ;waste some time
022152  106427  000140                  mtps    #140
022156  077007                          sob     r0,8$                   ;wait ten seconds
022160  000527                          br      timout                  ;timed out if here

022162  062706  000004          sp2int: add     #4,sp
022166  004737  024350                  jsr     pc,bit15T               ;look for sa error
022172  030314                          bit     r3,(r4)                 ;check step
022174  001541                          beq     wrngstep

022176  012700  002456          12$:    mov     #RSPRNG,R0              ;responding in step 2 <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<<<<<<<<
022202  042700  000001                  bic     #bit0,r0                ;no adapter purge interrupts
022206  013701  002264                  mov     vector,r1
022212  012721  022252                  mov     #sp3int,(r1)+           ;load up interrupt loacation into vector
022216  012711  000140                  mov     #140,(r1)               ;lower the priority
022222  010014                          mov     r0,(r4)                 ;load low ringbase address of tne commun'cat'ons are
a

022224  006303                          asl     r3                      ;next step
```

Global subroutines

```
   1260 022226  012700  177777                mov     #-1,r0                  ;looking for step 3 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>
   1261 022232  106427  000340        13$:     mtps    #340
   1262 022236                                break                           ;waste some time
   1263 022240                                break                           ;waste some time
   1264 022242  106427  000140                mtps    #140
   1265 022246  077007                        sob     r0,13$                  ;wait ten seconds
   1266 022250  000473                        br      timout
   1267 022252                        sp3int:
   1268 022252  062706  000004                add     #4,sp
   1269 022256  004737  024350                jsr     pc,bit15T
   1270 022262  030314                        bit     r3,(r4)
   1271 022264  001505                        beq     wrngstep
   1272
   1273 022266  013701  002264                mov     vector,r1               ;responding in step 3 <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<<<<<<<<<
   1274 022272  012721  022332                mov     #sp4int,(r1)+           ;load up interrupt loacation into vector
   1275 022276  012711  000340                mov     #340,(r1)               ;after step four we want no interrupts until expecte
d.
   1276 022302  005014                        clr     (r4)                    ;load low ringbase address of the communications are
a
   1277
   1278 022304  006303                        asl     r3
   1279 022306  012701  177777                mov     #-1,r1                  ;looking for step 4 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>
   1280 022312  106427  000340        18$:     mtps    #340
   1281 022316                                break                           ;waste some time
   1282 022320                                break                           ;waste some time
   1283 022322  106427  000140                mtps    #140
   1284 022326  077107                        sob     r1,18$                  ;wait ten seconds
   1285 022330  000443                        br      timout
   1286 022332                        sp4int:
   1287 022332  062706  000004                add     #4,sp
   1288 022336  004737  024350                jsr     pc,bit15T               ;check for SA error
   1289 022342  030314                        bit     r3,(r4)                 ;check for step 4
   1290 022344  001455                        beq     wrngstep
   1291
   1292 022346  011401                        mov     (r4),r1                 ;identify the controller number and microcode vers o
n number
   1293 022350  010102                        mov     r1,r2
   1294 022352  006201                        asr     r1                      ;shift version # out
   1295 022354  006201                        asr     r1
   1296 022356  006201                        asr     r1
   1297 022360  006201                        asr     r1
   1298 022362  042701  177400                bic     #177400,r1              ;clear top bits off
   1299 022366  010137  002272                mov     r1,mdlnbr               ;model number storage
   1300 022372  042702  177760                bic     #177760,r2              ;clear model number out
   1301 022376  010237  002274                mov     r2,mcdnbr
   1302 022402  122701  000007                cmpb    #Mrqdx1,r1              ;check for Model #
   1303 022406  001454                        beq     gobit
   1304 022410  122701  000023                cmpb    #Mrqdx3,r1              ;check for Model #
   1305 022414  001451                        beq     gobit
   1306
   1307 022416                                ERRSOFT 6,SFT1                  ; DEVICE FATAL wrong model #,wrong controller
   1308 022426  052737  020000  002270        bis     #bit13,UNTflgs          ;set unknown model number in unit flags
   1309 022434  000137  022540                jmp     gobit                   ;drop unit and go on
   1310
   1311 022440                        timout:
   1312 022440                                ERRDF   5,DF1                   ; DEVICE FATAL controller timeout during hard init
   1313 022450                                Printf  #pb1,r3,(r4)            ; Expected SA step bit xxxxx set, received yyyyyy
   1314 022474  000137  030606                jmp     dropunt                 ;drop unit and go on
   1315
   1316 022500                        wrngstep:
```

Global subroutines

```
1317 022500                                        ERRDF   4,DF2                   ; DEVICE FATAL wrong step bit set after interrupt
1318 022510                                        Printf  #pb1,r3,(r4)            ; Expected SA step bit xxxxx, rece'ved in SA yyyyyy
1319 022534  000137  030606                        jmp     dropunt                 ;drop unit and go on
1320
1321 022540                              GOBIT:
1322 022540  012714  000001                        mov     #1,(r4)                 ;Controller is NOW INITIALIZED
1323 022544  012700  177777                        mov     # 1,r0
1324 022550  000240                      1$:       nop                             ;waste just a l'ttle time so program can terminate
1325 022552  077002                                sob     r0,1$
1326 022554                              GDScmd:
1327 022554                              GETDUST                                   ;Do a Get Dust Status command start things off
     022554  032737  100000  002464     GDS2:     bit     #bit15,cmdrng+2         ;test ownership of ring make sure we own it
     022562  001374                              bne     GDS2                    ;if we don't own it wait until we do
     022564  012737  000016  002376              mov     #14.,cmdlen             ;load lenght of packet to be send
     022572  112737  000000  002400              movb    #0,cmdlen+2             ;load msg type and credit
     022600  112737  000002  002401              movb    #dup.id,cmdlen+3        ,load DUP connection ID
     022606  005237  002402                      inc     cmdpak                  ;load new CRN
     022612  005037  002404                      clr     cmdpak+2
     022616  005037  002406                      clr     cmdpak+4
     022622  005037  002410                      clr     cmdpak+6
     022626  012737  000001  002412              mov     #op.gds,cmdpak+10       ;load up opcode
     022634  005037  002414                      clr     cmdpak+12               ;no modifiers

     022640  012777  022702  157416              mov     #RFD2,@vector           ;New vector place
     022646  012737  002302  002456              mov     #rsppak,rsprng          ;load response packet area into ring
     022654  012737  002402  002462              mov     #cmdpak,cmdrng          ;load command packet area into ring
     022662  012737  140000  002460              mov     #140000,RSPRNG+2        ;Port ownership bit
     022670  012737  100000  002464              mov     #bit15,CMDRNG+2
     022676  004737  021250                      jsr     pc,POLLWT               ;Go to poll and wait routine.

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

     022702                                  RFD2:                             ;Intr to here.
     022702  062706  000006                      add     #6,sp                   ;fix stack for interrupt (4), pollwt subrtn (2)
     022706  012777  025412  157350              mov     #intsrv,@vector         ;Change vector
     022714  004737  023416                      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.
1328 022720  132737  000010  002321              bitb    #bit3,rsppak+17         ;is this server active already
1329 022726  001467                              beq     dnint                   ;branch tc Execute Local Program
1330 022730                                      ERRSOFT 3,SFTO                  ;Soft Error "already active  will do an ABORT cmd
1331 022740                              ABRT                                    ;Doing an ABRT do get into idle state
     022740  032737  100000  002464     ABRT3: bit      #bit15,cmdrng+2         ;test ownership of ring make sure we own 't
     022746  001374                              bne     ABRT3                   ;if we don't own it wait until we do
     022750  012737  000016  002376              mov     #14.,cmdlen             ;load lenght of packet to be send
     022756  112737  000000  002400              movb    #0,cmdlen+2             ;load msg type and cred't
     022764  112737  000002  002401              movb    #dup.id,cmdlen+3        ;load DUP connection ID
     022772  005237  002402                      inc     cmdpak                  ;load new CRN
     022776  005037  002404                      clr     cmdpak+2
     023002  005037  002406                      clr     cmdpak+4
     023006  005037  002410                      clr     cmdpak+6
     023012  012737  000006  002412              mov     #op.abrt,cmdpak+10      ;load up opcode
     023020  005037  002414                      clr     cmdpak+12               ;no modifiers

     023024  012777  023066  157232              mov     #RFD3,@vector           ;New vector place
     023032  012737  002302  002456              mov     #rsppak,rsprng          ;load response packet area 'nto ring
```

```
Global subroutines

        023040  012737  002402  002462          mov     #cmdpak,cmdrng          ;load command packet area into ring
        023046  012737  140000  002460          mov     #140000,RSPRNG+2        ;Port ownership bit.
        023054  012737  100000  002464          mov     #bit15,CMDRNG+2
        023062  004737  021250                  jsr     pc,POLLWT               ;Go to poll and wait routine.

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

        023066                              RFD3:                               ;Intr to here.
        023066  062706  000006                  add     #6,sp                   ;fix stack for interrupt (4), pollwt subrtn (2)
        023072  012777  025412  157164          mov     #intsrv,@vector         ;Change vector
        023100  004737  023416                  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.
   1332 023104  000623                          br      GDScmd                  ;branch back to make sure not busy
   1333 023106                          DNINT:
   1334 023106  000207                          rts     pc
   1335
   1336                                 ;***************************************************************************
   1337                                 ;
   1338                                 ;       Octal number to ASCII Decimal number
   1339                                 ;       r1 = address of ascii decimal data
   1340                                 ;       r0 = octal data word
   1341                                 ;***************************************************************************
   1342 023110                          OCTASC:
   1343 023110  010246                          mov     r2,-(sp)
   1344 023112  010346                          mov     r3,-(sp)
   1345 023114  005002                          clr     r2                      ;clear the decimal table pointer
   1346 023116  005003                  1$:     clr     r3                      ;clear decimal digit
   1347 023120  005203                  2$:     inc     r3                      ;increment decimal digit
   1348 023122  166200  023162                  sub     dectbl(r2),r0           ;subtract a power of ten from accumulator
   1349 023126  002374                          bge     2$                      ;if not negative subtract another
   1350 023130  066200  023162                  add     dectbl(r2),r0           ;adjust accumulator so positive
   1351 023134  005303                          dec     r3                      ;adjust decimal digit
   1352 023136  062703  000060                  add     #60,r3                  ;convert decimal to ascii
   1353 023142  110321                          movb    r3,(r1)+                ;mov ascii digit text into buffer
   1354 023144  005722                          tst     (r2)+                   ;increment table pointer
   1355 023146  005762  023162                  tst     dectbl(r2)              ;check if thats all
   1356 023152  001361                          bne     1$
   1357 023154  012603                          mov     (sp)+,r3
   1358 023156  012602                          mov     (sp)+,r2
   1359 023160  000207                          rts     pc
   1360 023162                          dectbl:
   1361 023162  023420                          .word   10000.
   1362 023164  001750                          .word   1000.
   1363 023166  000144                          .word   100.
   1364 023170  000012                          .word   10.
   1365 023172  000001                          .word   1.
   1366 023174  000000                          .word   0
   1367                                 ;***************************************************************************
   1368                                 ;
   1369                                 ;       ASCII DECIMAL numbers to  Octal numbers
   1370                                 ;       r1 = address of ascii decimal data
   1371                                 ;       r0 = address to store octal data low word, high word
   1372                                 ;***************************************************************************
   1373 023176                          ASCDEC:
   1374 023176  010546                          mov     r5,-(sp)
```

H3

Global subroutines

```
   1375 023200  010446                          mov     r4,-(sp)
   1376 023202  010346                          mov     r3, (sp)
   1377 023204  010246                          mov     r2,-(sp)
   1378 023206  005004                          clr     r4
   1379 023210  005003                          clr     r3
   1380 023212  005002                          clr     r2
   1381 023214  112104                  3$:     movb    (r1)+,r4
   1382 023216  001423                          beq     1$                      ;if digit equals null than all done
   1383                                 ;       cmp     r4,#60                  ;check for a real number value
   1384                                 ;       blt     asklbn                  ;wasn't a real number
   1385                                 ;       cmp     r4,#71
   1386                                 ;       bgt     asklbn                  ;wasn't a real number
   1387
   1388 023220  162704  000060                  sub     #60,r4
   1389 023224  010346                          mov     r3,-(sp)
   1390 023226  010246                          mov     r2,-(sp)                ;save accum
   1391
   1392 023230  012705  000003                  mov     #3,r5                   ;accum * 8
   1393 023234  006302                  4$:     asl     r2
   1394 023236  006103                          rol     r3
   1395 023240  077503                          sob     r5,4$
   1396
   1397 023242  006316                          asl     (sp)                    ;accum*2
   1398 023244  006166  000002                  rol     2(sp)
   1399
   1400 023250  000241                          clc                             ; accum*8 + accum*2
   1401 023252  062602                          add     (sp)+,r2
   1402 023254  005503                          adc     r3
   1403 023256  062603                          add     (sp)+,r3
   1404
   1405 023260  060402                          add     r4,r2                   ;add present digit to accum*10
   1406 023262  005503                          adc     r3
   1407 023264  000753                          br      3$
   1408
   1409 023266  010220                  1$:     mov     r2,(r0)+                ;load lo number
   1410 023270  010310                          mov     r3,(r0)                 ;load hi number
   1411
   1412 023272  012602                          mov     (sp)+,r2                ;restore stack to its orginal
   1413 023274  012603                          mov     (sp)+,r3
   1414 023276  012604                          mov     (sp)+,r4
   1415 023300  012605                          mov     (sp)+,r5
   1416 023302  000207                          rts     pc
   1417
   1418                                 ;*****************************************************************
   1419                                 ;
   1420                                 ; This routine types out the ASCII information passed
   1421                                 ; by the disk controller. This ASCII information is
   1422                                 ; contained in the buffer called DATARE and is offset
   1423                                 ; by 1 word. To fake the DRS macro routine a "$A" is
   1424                                 ; placed in front of the text.
   1425                                 ;*****************************************************************
   1426
   1427 023304                          typDUPbuf:
   1428 023304  012701  002502                  mov     #datare,r1      ;get data area address of ascii info
   1429 023310  063701  002316                  add     rsppak+14,r1    ;add the number of byte transfered
   1430 023314  105021                  1$:     clrb    (r1)+           ;put null characters into data buffer after end of ASCII inf
o
   1431 023316  020127  002626                  cmp     r1,#prgnam      ;
```

Global subroutines

```
1432 023322  001374                          bne     1$              ;we do this to fake out the DRS macro
1433
1434 023324  112737  000045  002502          movb    #45,datare      ;put the "%" del'miter for the DRS macro
1435 023332  112737  000101  002503          movb    #101,datare+1   ;put the "A" for ascii info for the DRS macro
1436 023340                                  printx  #PB13           ;New Line <cr><lf>
1437 023360                                  pr'ntx  #datare         ;pr'nt the message returned from the controller
1438
1439 023400                          clrDUPbuf:
1440 023400  012701  002502                  mov     #datare,r1      ;clear out entire data area
1441 023404  105021                  2$:     clrb    (r1)+           ;
1442 023406  020127  002626                  cmp     r1,#prgnam      ;
1443 023412  001374                          bne     2$              ;
1444 023414  000207                          rts     pc
1445                                 ;*****************************************************************
1446                                 ;
1447                                 ;       THIS ROUTINE IS TO CHECK ON THE RESPONSE PACKET
1448                                 ;       GOODNESS. THE COMMAND REFERENCE NUMBER, THE END CODE
1449                                 ;       AND THE STATUS ARE TESTED.
1450                                 ;*****************************************************************
1451
1452 023416                          RSPCHK:
1453
1454 023416  013701  002402              mov         cmdpak,r1
1455 023422  013700  002302              mov         rsppak,r0
1456 023426  020001                      cmp         r0,r1                   ;compare CRN numbers
1457 023430  001014                      bne         1$
1458 023432  013701  002412              mov         cmdpak+10,r1
1459 023436  062701  000200              add         #200,r1
1460 023442  013700  002312              mov         rsppak+10,r0
1461 023446  020001                      cmp         r0,r1                   ;compare Opcodes
1462 023450  001004                      bne         1$
1463 023452  013701  002314              mov         rsppak+12,r1            ;check the status
1464 023456  001001                      bne         1$
1465 023460  000207                      rts         pc                      ;if all checks then return
1466
1467                                                                         ;if all doesn't check then a bad packet
1468 023462                          1$:     ERRDF   10,df11                 ;Bad response packet
1469 023472                          PRNTpkt:
1470 023472                                  Printb  #PB11crn,cmdpak,rsppak  ;Expected CRN XXXX ,Received CRN YYYY
1471 023522  013701  002312                  mov     rsppak+10,r1            ;check response opcode reply
1472 023526  032701  000200                  bit     #200,r1                 ;see if a end command response was send
1473 023532  001010                          bne     2$
1474 023534                                  printx  #PB11end                ;No end bit in response packet endcode
1475 023554  022701  000201          2$:     cmp     #201,r1
1476 023560  001010                          bne     3$                      ;check if Get Dust Status command
1477 023562                                  printx  #PB11GDS
1478 023602  022701  000202          3$:     cmp     #202,r1
1479 027606  001010                          bne     4$                      ;check if Execute Supplied Program
1480 023610                                  printx  #PB11ESP
1481 023630  022701  000203          4$:     cmp     #203,r1
1482 023634  001010                          bne     5$                      ;check if Execute Local Program
1483 023636                                  printx  #PB11ELP
1484 023656  022701  000204          5$:     cmp     #204,r1
1485 023662  001010                          bne     6$                      ;check if Send Data
1486 023664                                  printx  #PB11SD
1487 023704  022701  000205          6$:     cmp     #205,r1
1488 023710  001022                          bne     7$                      ;check if Receive Data
```

Global subroutines

```
1489 023712                                    printx  #PB11RD
1490 023732                                    Printb  #PBSF0,r3,r5    ;'type xxx, message number xxxxx 's unknow to this program"
1491 023756  022701  000206          7$:      cmp     #206,r1
1492 023762  001010                           bne     8$                      ;check if Abort Program
1493 023764                                    printx  #PB11AP
1494 024004                          8$:      Printb  #PB11op,cmdpak+10,rsppak+10
1495                                                                          ;CMDpkt opcode XXXX,RSPpkt opcode YYYY
1496
1497 024034  013701  002314                    mov     rsppak+12,r1            ;find out what kind of status we have
1498 024040  022701  000000                    cmp     #0.,r1
1499 024044  001010                           bne     10$
1500 024046                                    printx  #pb11s0                 ;status:  successful
1501 024066  022701  000001          10$:     cmp     #1.,r1
1502 024072  001010                           bne     11$
1503 024074                                    printx  #pb11s1                 ;status:  Inval'd Command
1504 024114  022701  000002          11$:     cmp     #2.,r1
1505 024120  001010                           bne     12$
1506 024122                                    printx  #pb11s2                 ;status:  No Region Available
1507 024142  022701  000003          12$:     cmp     #3.,r1
1508 024146  001010                           bne     13$
1509 024150                                    printx  #pb11s3                 ;status:  No Reg'on Suitable
1510 024170  022701  000004          13$:     cmp     #4.,r1
1511 024174  001010                           bne     14$
1512 024176                                    printx  #pb11s4                 ;status:  Program Not Known
1513 024216  022701  000005          14$:     cmp     #5.,r1
1514 024222  001010                           bne     15$
1515 024224                                    printx  #pb11s5                 ;status:  Load Failure
1516 024244  022701  000006          15$:     cmp     #6.,r1
1517 024250  001010                           bne     16$
1518 024252                                    printx  #pb11s6                 ;status:  Standalone
1519 024272  022701  000011          16$:     cmp     #9.,r1
1520 024276  001010                           bne     19$
1521 024300                                    printx  #pb11s9                 ;status:  Host Buffer Access error
1522 024320                          19$:
1523 024320                                    Printb  #PB11sts,rsppak+12      ;Response packet status XXXX
1524 024344  000137  030606                    jmp     dropunt                 ;drop unit and go on
1525
1526
1527                                 ;*************************************************************************
1528                                 ;
1529                                 ;                          BIT FIFTEEN TEST
1530                                 ;*************************************************************************
1531 024350                          BIT15T:
1532 024350  032714  100000                    bit     #bit15,(r4)
1533 024354  001001                           bne     100$
1534 024356  000207                           rts     pc
1535 024360                          100$:    ERRDF   9,df12                  ;Fatal SA error
1536 024370  011401                           mov     (r4),r1
1537 024372  022701  001000                    cmp     #1000,r1
1538 024376  001010                           bne     1$
1539 024400                                    printx  #pb1201                 ;
1540 024420  022701  100001          1$:      cmp     #100001,r1
1541 024424  001010                           bne     2$
1542 024426                                    printx  #pb1202                 ;
1543 024446  022701  100002          2$:      cmp     #100002,r1
1544 024452  001010                           bne     3$
1545 024454                                    printx  #pb1203                 ;
```

Global subroutines

```
1546 024474  022701  100003          3$:     cmp     #100003,r1
1547 024500  001010                          bne     4$
1548 024502                                  printx  #pb1204         ;
1549 024522  022701  100004          4$:     cmp     #100004,r1
1550 024526  001010                          bne     5$
1551 024530                                  printx  #pb1205         ;
1552 024550  022701  100005          5$:     cmp     #100005,r1
1553 024554  001010                          bne     6$
1554 024556                                  printx  #pb1206         ;
1555 024576  022701  100006          6$:     cmp     #100006,r1
1556 024602  001010                          bne     7$
1557 024604                                  printx  #pb1207         ;
1558 024624  022701  100007          7$:     cmp     #100007,r1
1559 024630  001010                          bne     8$
1560 024632                                  printx  #pb1208         ;
1561 024652  022701  100010          8$:     cmp     #100010,r1
1562 024656  001010                          bne     9$
1563 024660                                  printx  #pb1209         ;
1564 024700  022701  100011          9$:     cmp     #100011,r1
1565 024704  001010                          bne     10$
1566 024706                                  printx  #pb1210         ;
1567 024726  022701  100012          10$:    cmp     #100012,r1
1568 024732  001010                          bne     11$
1569 024734                                  printx  #pb1211         ;
1570 024754  022701  100013          11$:    cmp     #100013,r1
1571 024760  001010                          bne     12$
1572 024762                                  printx  #pb1212         ;
1573 025002  022701  100014          12$:    cmp     #100014,r1
1574 025006  001010                          bne     13$
1575 025010                                  printx  #pb1213         ;
1576 025030  022701  100015          13$:    cmp     #100015,r1
1577 025034  001010                          bne     14$
1578 025036                                  printx  #pb1214         ;
1579 025056  022701  100016          14$:    cmp     #100016,r1
1580 025062  001010                          bne     15$
1581 025064                                  printx  #pb1215         ;
1582 025104  022701  100017          15$:    cmp     #100017,r1
1583 025110  001010                          bne     16$
1584 025112                                  printx  #pb1216         ;
1585 025132  022701  100020          16$:    cmp     #100020,r1
1586 025136  001010                          bne     17$
1587 025140                                  printx  #pb1217         ;
1588 025160  022701  100021          17$:    cmp     #100021,r1
1589 025164  001010                          bne     18$
1590 025166                                  printx  #pb1218         ;
1591 025206  022701  100022          18$:    cmp     #100022,r1
1592 025212  001010                          bne     19$
1593 025214                                  printx  #pb1219         ;
1594 025234  022701  100023          19$:    cmp     #100023,r1
1595 025240  001010                          bne     20$
1596 025242                                  printx  #pb1220         ;
1597 025262  022701  100024          20$:    cmp     #100024,r1
1598 025266  001010                          bne     21$
1599 025270                                  printx  #pb1221         ;
1600 025310  022701  100025          21$:    cmp     #100025,r1
1601 025314  001010                          bne     22$
1602 025316                                  printx  #pb1222         ;
```

Global subroutines

```
1603 025336  022701  100026          22$:    cmp     #100026,r1
1604 025342  001010                          bne     23$
1605 025344                                  printx  #pb1223                 ;
1606 025364                          23$:
1607 025364                                  printb  @pb12,r1                ;SA value: xxxxx
1608 025406  000137  030606                  jmp     dropunt                 ;drop unit and go on
1609
1610                                 ;*****************************************************************
1611                                 ;       Unexpected Interrupt Server
1612                                 ;
1613                                 ;*****************************************************************
1614 025412                          intsrv:
1615
1616 025412                                  ERRSF   8,sf100 ;Fatal SA error
1617 025422                                  docln                           ;do clean up and quit
1618 025424  000137  030606                  jmp     dropunt         ;drop test un't and end pass
1619
1620
```

Global subroutines

```
   1622 025430                                  BGNPROT
   1623 025430  177777                          .WORD  1
   1624 025432  177777                          .WORD  1
   1625 025434  177777                          .WORD  1
   1626 025436                                  ENDPROT
   1627
   1628 025436                                   BGNINIT                        ;Sequential example
   1629 025436                                   READEF          #EF.CONTINUE   ;Continue command?
   1630 025444                                   BCOMPLETE       conton         ;Yes, get no P table but st'll init'al'ze
   1631 025446                                   READEF          #EF.NEW        ;New pass
   1632 025454                                   BNCOMPLETE      next           ;if not new then go to next unit number
   1633 025456                                 SETUP:
   1634 025456  012737  177777  002246                 mov      # 1,LOGUNIT     ;Init'alize logical unit nbr
   1635 025464                                 NEXT:
   1636 025464  005237  002246                 inc      LOGUNIT                ;Point to next logical unit
   1637 025470  023737  002246  002012         cmp      LOGUNIT,L$UNIT         ;Have we passed maximum?
   1638 025476  001002                         bne      1$                     ;No
   1639 025500  000137  025704                 jmp      ABORT                  ;Yes, abort the pass
   1640 025504                                 1$:
   1641 025504                                 GPHARD LOGUNIT,PLOC             ;Get the P-table
   1642 025516                                 BNCOMPLETE NEXT                 ;if not available get next unit
   1643
   1644 025520  013700  002252                 mov      ploc,r0
   1645 025524  010037  002254                 mov      r0,ptbl                ;store the Ptable address for unit
   1646 025530  012037  002262                 mov      (r0)+,ipreg            ;store IPreg address into register
   1647 025534  012037  002264                 mov      (r0)+,vector           ;store vector
   1648 025540  012037  002266                 mov      (r0)+,unit             ;store logical drive number
   1649 025544  012037  002270                 mov      (r0)+,untflgs
   1650
   1651 025550  005037  002470         conton: clr      LSTCRN                 ;basic initialization stuff
   1652 025554  005037  002474                 clr      LSTVCT
   1653 025560  005037  002476                 clr      LOPRGI
   1654 025564  005037  002500                 clr      HIPRGI
   1655
   1656 025570  032737  100000  002270         bit      #bit15,untflgs
   1657 025576  001411                         beq      1$                     ;
   1658 025600  032737  040000  002270         bit      #bit14,untflgs
   1659 025606  001005                         bne      1$
   1660 025610                                 dodu     logunit                ;if in auto mode and warning flag isn't acknowledge
drop unit
   1661 025616  000137  025704                 jmp      abort
   1662
   1663 025622  013746  000004         1$:     mov      @#4,-(sp)              ;test to see 'f controller is there
   1664 025626  012737  025642  000004         mov      #$2,@#4
   1665 025634  005077  154422                 clr      @IPreg                 ;get controller into know state
   1666 025640  000410                         br       $3
   1667
   1668 025642                                 $2:      ERRDF   7,DF4          ;NXM trap at controller IP address
   1669 025652                                          dodu    LOGUNIT        ;drop unit
   1670 025660  000701                                  br      next           ;get new unit
   1671
   1672 025662  012637  000004         $3:     mov      (sp)+,@#4              ;move value back into location 4
   1673
   1674 025666  012700  000076                 mov      #76,r0                 ;clean out all packets and 'nterrupt flags
   1675 025672  012701  002276                 mov      #rsp1,r1               ;and the command area
   1676 025676  005021                 $4:     clr      (r1)+
   1677 025700  077002                         sob      r0,$4
   1678
```

Global subroutines

```
1679 025702  000401                          br      end
1680
1681 025704                          ABORT:
1682 025704                                  DOCLN                          ;Do clean-up and abort the pass
1683 025706                           END:
1684 025706                                  ENDINIT                        ;Finished
1685
1686
1687 025710                           BGNAUTO
1688 025710                           DODU LOGUNIT
1689 025716                           ENDAUTO
1690
1691 025720                           BGNCLN
1692 025720  005077  154336                   clr     @IPreg                 ;get controller into know state
1693 025724                                  Break                          ;waste some time
1694 025726                           ENDCLN
1695
1696 025730                           BGNDU
1697 025730                                  printf #drpunt,unit
1698 025754                           ENDDU
1699
```

Global subroutines

```
   1701 025756                                  BGNTST 1
   1702 025756                                ELPcmd:
   1703
   1704 025756  005037  002260                        clr     boot                    ; WARNING   remove boot diskette first
   1705 025762                                GMANIL  bot.dev,BOOT,-1,YES             ; Insert new diskette
   1706                                                                               ; DO you want to continue
   1707 025776  005737  002260                        tst     BOOT                    ;
   1708 026002  001002                                bne     1$                      ; Yes, run format
   1709 026004  000137  030606                        jmp     dropunt                 ; No, drop unit
   1710 026010                                1$:
   1711
   1712 026010  004737  022040                        jsr     pc,hrdint               ; Reinit ctrl in case of unknown state
   1713 026014                                    printb  @pb9,mdlnbr                 ; Print the disk controller model number
   1714 026040                                    printb  @pb10,mcdnbr                ; Print microcode version number in dec.
   1715
   1716 026064  012737  047506  002626                mov     #"FO,PRGnam             ;place "FORMAT" into ascii buffer if in auto mode
   1717 026072  012737  046522  002630                mov     #"RM,PRGnam+2
   1718 026100  012737  052101  002632                mov     #"AT,PRGnam+4
   1719 026106                                EXLCPRG PRGnam                          ;Execute Local program "FORMAT" or what ever they wr
ote
        026106  032737  100000  002464        ELP4:   bit     #bit15,cmdrng+2         ;test ownership of ring make sure we own it
        026114  001374                                bne     ELP4                    ;if we don't own it wait until we do
        026116  012737  000022  002376                mov     #22,cmdlen              ;load lenght of packet to be send
        026124  112737  000000  002400                movb    #0,cmdlen+2             ;load msg type and cred't
        026132  112737  000002  002401                movb    #dup.id,cmdlen+3        ;load DUP connection ID
        026140  005237  002402                        inc     cmdpak                  ;load new CRN
        026144  005037  002404                        clr     cmdpak+2
        026150  005037  002406                        clr     cmdpak+4
        026154  005037  002410                        clr     cmdpak+6
        026160  012737  000003  002412                mov     #op.elp,cmdpak+10       ;load up opcode
        026166  012737  000001  002414                mov     #stdaln,cmdpak+12       ;stand alone modifier
        026174  012700  000006                        mov     #6,r0                   ;6 letters transfer
        026200  012701  002416                        mov     #cmdpak+14,r1           ;starting address to place program name
        026204  012702  002626                        mov     #PRGnam,r2              ;start of Program Name
        026210  112221                        rfdj4:  movb    (r2)+,(r1)+             ;add 2 to bycnt then store
        026212  077002                                sob     r0,rfdj4

        026214  012777  026256  154042                mov     #RFD4,@vector           ;New vector place
        026222  012737  002302  002456                mov     #rsppak,rsprng          ;load response packet area into ring
        026230  012737  002402  002462                mov     #cmdpak,cmdrng          ;load command packet area into ring
        026236  012737  140000  002460                mov     #140000,RSPRNG+2        ;Port ownership bit.
        026244  012737  100000  002464                mov     #bit15,CMDRNG+2
        026252  004737  021250                        jsr     pc,POLLWT               ;Go to poll and wait routine.

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

        026256                                    RFD4:
        026256  062706  000006                        add     #6,sp                   ;Intr to here.
        026262  012777  025412  153774                mov     #intsrv,@vector         ;fix stack for interrupt (4), pollwt subrtn (2)
        026270  004737  023416                        jsr     pc,RSPCHK               ;Change vector
                                                                                       ;Go to routine that will check on
                                                                                       ;the response recvd from the mut.
                                                                                       ;it will check the cmd ref
                                                                                       ;num, the endcode and status.
   1720
   1721 026274  122737  000011  002321                cmpb    #bit3+bit0,rsppak+17    ;is this program a standalone,DUP dialog type
   1722 026302  001406                                beq     1$                      ;
   1723 026304                                        ERRDF   2,DF3                   ;"Device Fatal can't do remote programs"
   1724 026314  000137  030606                        jmp     dropunt                 ;drop unit and go on
```

Global subroutines

```
 1725 026320                                1$:
 1726 026320                                RCDcmd:
 1727 026320                                RECVDAT @datare,#80.
      026320  032737  100000  002464        RCD5:   bit     #bit15,cmdrng+2         ;test ownership of ring make sure we own it
      026326  001374                                bne     RCD5                    ;if we don't own it wait until we do
      026330  012737  000034  002376                mov     #34,cmdlen              ;load lenght of packet to be send
      026336  112737  000000  002400                movb    #0,cmdlen+2             ;load msg type and credit
      026344  112737  000002  002401                movb    #dup.id,cmdlen+3        ;load DUP connection ID
      026352  005237  002402                        inc     cmdpak                  ;load new CRN
      026356  005037  002404                        clr     cmdpak+2
      026362  005037  002406                        clr     cmdpak+4
      026366  005037  002410                        clr     cmdpak+6
      026372  012737  000005  002412                mov     #op.rec,cmdpak+10       ;load up opcode
      026400  005037  002414                        clr     cmdpak+12               ;no modifiers
      026404  012737  000120  002416                mov     #80.,cmdpak+14
      026412  005037  002420                        clr     cmdpak+16
      026416  012737  002502  002422                mov     #datare,cmdpak+20       ;load address of buffer describtor
      026424  005037  002424                        clr     cmdpak+22
      026430  005037  002426                        clr     cmdpak+24
      026434  005037  002430                        clr     cmdpak+26
      026440  005037  002432                        clr     cmdpak+30
      026444  005037  002434                        clr     cmdpak+32

      026450  012777  026512  157606                mov     #RFD5,@vector           ;New vector place
      026456  012737  002302  002456                mov     #rsppak,rsprng          ;load response packet area into ring
      026464  012737  002402  002462                mov     #cmdpak,cmdrng          ;load command packet area into ring
      026472  012737  140000  002460                mov     #140000,RSPRNG+2        ;Port ownership bit.
      026500  012737  100000  002464                mov     #bit15,CMDRNG+2
      026506  004737  021250                        jsr     pc,POLLWT               ;Go to poll and wait routine.

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

      026512                                RFD5:                                   ;Intr to here.
      026512  062706  000006                        add     #6,sp                   ;fix stack for interrupt (4), pollwt subrtn (2)
      026516  012777  025412  153540                mov     #intsrv,@vector         ;Change vector
      026524  004737  023416                        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.
 1728                                       ;+
 1729                                       ;       get
 1730                                       ;       r3 = type
 1731                                       ;       r4 = SA adrs
 1732                                       ;       r5 = sub number
 1733                                       ;-
 1734 026530  113703  002503                DUPDLG: movb    datare+1,r3             ;get dup type info
 1735 026534  006203                                asr     r3
 1736 026536  006203                                asr     r3
 1737 026540  006203                                asr     r3
 1738 026542  006203                                asr     r3
 1739 026544  042703  177760                        bic     #type,r3                ;mask off all but DUP type
 1740 026550  013705  002502                        mov     datare,r5               ;get dup message number info
 1741 026554  042705  170000                        bic     #msgnbr,r5              ;clear out top 4 bits
 1742
 1743
 1744                                       ;+
 1745                                       ; Check for the type.
```

Global subroutines

```
                                        ; If QUESTION type, it will be answered by sending
                                        ; an answer through a Send command which will be followed
                                        ; by a Receive command to await further instructions.
                                        ;
                                        ; If a DEFAULT QUESTION type is given an answer will
                                        ; either be given or a blank send command returned.
                                        ; Either way we will do a Send command followed by a
                                        ; Receive command.
                                        ;
                                        ; If INFORMATIONAL type, check message number and type
                                        ; information according to message number given.
                                        ;
                                        ; if FATAL ERROR type, check message number and print
                                        ; error message accordingly. No other commands will
                                        ; be given following this type of command.
                                        ;
                                        ; If TERMINATION type check the message number and print the
                                        ; correct message. Ussually this implies a succesful
                                        ; end to the formatter. After this command we exit the program
                                        ;
                                        ; If SPECIAL type we are asking for the FCT table to be passed
                                        ; to the RQDX3 controller. We will send the table with a Send
                                        ; command and then to a Receive command to proceed.
                                        ;
026560  022703  000001        qstn:   cmp     #Question,r3            ;test for "question' subtype
026564  001002                        bne     dfqstn                  ;if not branch
                                      
026566  000137  030572                jmp     spcl
                              ;qnbra: jsr     pc,typDUPbuf            ;type out ASCII sent by disk controller
                              ;GMANID  ASK.ANSWER,DATARE,A,177777,0..10.,YES  ;give it an answer
                              ;        jmp     SDTcmd                  ;branch to Send Data command
                                      
                                      
026572  022703  000002        dfqstn: cmp     #DefQuest,r3            ;test for "Default Question" subtype
026576  001402                        beq     dqnbr1
026600  000137  027112                jmp     infrm                   ;if not branch
                                      
026604  004737  023400        dqnbr1: jsr     pc,clrDUPbuf            ;clear out data buffer so DRS macros don t show defa
ult
026610  022705  000001                cmp     #1,r5                   ;check for message number
026614  001026                        bne     dqnbra                  ;check for next message number
                                                                      ;put in message number
026616  013700  002266                mov     unit,r0                 ;get unit number if in auto mode from Hardware P tab
le
026622  012701  002502                mov     #datare,r1              ;store decimal ascii conversion in data area
026626  004737  023110                jsr     pc,OCTASC               ;convert octal to ascii decimal in data area
                                      
026632  012701  002502        4$:     mov     #datare,r1              ;address of ascii decimal data
026636  012700  002266                mov     #unit,r0                ;address to store octal conversion
026642  004737  023176                jsr     pc,ASCDEC               ;convert ascii decimal to octal
026646  022737  000003  002266  2$:     cmp     #3,unit                 ;make sure unit number is less than 4 or between 0 3
026654  002004                        bge     1$
026656  162737  000004  002266        sub     #4,unit                 ;subtract 4 until unit is less than four
026664  000770                        br      2$
026666  000137  026676        1$:     jmp     SDTcmd                  ;branch to Send Data command
                                      
                                                                      ;if unknown use default and continue
                                                                      ;who knows maybe it will be useful some day
026672                        dqnbra:
```

Global subroutines

```
   1803 026672  000137  030572                jmp     spcl
   1804                                ;       jsr     pc,typDUPbuf            ;type out ASCII sent by d'sk controller
   1805                                ;GMANID  ASK.ANSWER,DATARE,A,177777,0.,10.,YES          ;g ve it an answer
   1806 026676                         SDTcmd:
   1807 026676                                 SENDDAT #datare,#10.            ;sent the answer
        026676  032737  100000  002464  SDT6:   bit     #bit15,cmdrng+2         ;test ownership of ring make sure we own it
        026704  001374                         bne     SDT6                    ;if we don't own it wait until we do
        026706  012737  000034  002376         mov     #34,cmdlen              ;load lenght of packet to be send
        026714  112737  000000  002400         movb    #0,cmdlen+2             ;load msg type and credit
        026722  112737  000002  002401         movb    #dup.id,cmdlen+3        ;load DUP connection ID
        026730  005237  002402                 inc     cmdpak                  ;load new CRN
        026734  005037  002404                 clr     cmdpak+2
        026740  005037  002406                 clr     cmdpak+4
        026744  005037  002410                 clr     cmdpak+6
        026750  012737  000004  002412         mov     #op.sen,cmdpak+10       ;load up opcode
        026756  005037  002414                 clr     cmdpak+12               ;no modifiers
        026762  012737  000012  002416         mov     #10.,cmdpak+14
        026770  005037  002420                 clr     cmdpak+16
        026774  012737  002502  002422         mov     #datare,cmdpak+20       ;load address of buffer describtor
        027002  005037  002424                 clr     cmdpak+22
        027006  005037  002426                 clr     cmdpak+24
        027012  005037  002430                 clr     cmdpak+26
        027016  005037  002432                 clr     cmdpak+30
        027022  005037  002434                 clr     cmdpak+32

        027026  012777  027070  153230         mov     #RFD6,@vector           ;New vector place
        027034  012737  002302  002456         mov     #rsppak,rsprng          ;load response packet area into r'ng
        027042  012737  002402  002462         mov     #cmdpak,cmdrng          ;load command packet area into r'ng
        027050  012737  140000  002460         mov     #140000,RSPRNG+2        ;Port ownership bit.
        027056  012737  100000  002464         mov     #b't15,CMDRNG+2
        027064  004737  021250                 jsr     pc,POLLWT               ;Go to poll and wait routine.

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

        027070                          RFD6:                                  ;Intr to here.
        027070  062706  000006                 add     #6,sp                   ;fix stack for interrupt (4), pollwt subrtn (2)
        027074  012777  025412  153162         mov     #intsrv,@vector         ;Change vector
        027102  004737  023416                 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.
   1808 027106  000137  026320                 jmp     RCDcmd                  ;do another receive cmd
   1809
   1810
   1811
   1812 027112  022703  000003         infrm:  cmp     #Inform,r3              ;test for "Informational" subtype
   1813 027116  001040                         bne     term                    ;if not branch
   1814
   1815 027120  022705  000000         inbr0:  cmp     #0,r5                   ;check for message number
   1816 027124  001013                         bne     inbr1                   ;check for next message number
   1817 027126  004737  023400                 jsr     pc,clrDUPbuf            ;clear out DUP buffer so there is no echo on last AS
CII
   1818 027132                                 printf  #sfbegt                 ;format begun
   1819 027152  000420                         br      inbrr
   1820
   1821 027154  022705  000001         inbr1:  cmp     #1,r5                   ;check for message number
   1822 027160  001013                         bne     inbra                   ;check for next message number
   1823 027162  004737  023400                 jsr     pc,clrDUPbuf            ;clear out DUP buffer so there 's no echo on last AS
CII
```

Global subroutines

```
1824 027166                                    printf  #sfdont                ;format complete
1825 027206  000402                            br      inbrr
1826
1827 027210  004737  023304           inbra:  jsr     pc,typDUPbuf            ;type out ASCII sent by disk controller
1828 027214  000137  026320           inbrr:  jmp     RCDcmd                  ;do another receive command
1829
1830
1831
1832 027220  022703  000004           term:   cmp     #terminat,r3            ;test for termination type
1833 027224  001055                            bne     ftler                   ;if not branch
1834
1835 027226  022705  000015           tnbr13: cmp     #13.,r5                 ;test for msg number
1836 027232  001036                            bne     tnbra                   ;branch if not right number
1837 027234                                    printf  #sffcnt                 ;
1838 027254  005077  153002                    clr     @IPreg                  ;can any spurious interrupts
1839 027260                           GMANIL  bot.con,BOOT, 1,YES             ; Do you want to format another?
1840
1841 027274  005737  002260                    tst     BOOT                    ; Yes, execute local program
1842 027300  001007                            bne     1$                      ; No, tell him to insert bootable media
1843
1844 027302                           GMANIL  bot.rep,BOOT,-1,YES             ; Please insert boot media and hit return
1845 027316  000402                            br      2$                      ;
1846 027320  000137  025756           1$:     jmp     ELPcmd                  ;
1847 027324  000137  030606           2$:     jmp     dropunt                 ;
1848
1849 027330  004737  023304           tnbra:  jsr     pc,typDUPbuf            ;type out ASCII sent by disk controller
1850 027334                                    printf  #PF2                    ;print finished local program without procedure erro
r
1851 027354  000137  030614                    jmp     etst                    ;end DUP diaglog but stay in test loop
1852
1853
1854 027360  022703  000005           ftler:  cmp     #ftlerr,r3              ;test for "Fatal Error" subtype
1855 027364  001402                            beq     2$
1856 027366  000137  030572                    jmp     spcl                    ;if not branch
1857
1858
1859 027372                           2$:     ERRHRD  1,HRD0                  ;Hard device  error
1860
1861 027402  022705  000001           fnbr1:  cmp     #1,r5                   ;test for sub number #1
1862 027406  001012                            bne     fnbr2                   ;branch if not sub number #1
1863 027410                           gstsf:
1864 027410                                    printb  #efstat                 ;"GET STATUS failure"
1865 027430  000137  030606                    jmp     dropunt                 ;drop unit and end pass
1866
1867 027434  022705  000002           fnbr2:  cmp     #2.,r5                  ;test for msg number
1868 027440  001012                            bne     fnbr3                   ;branch if not right number
1869 027442                                    printf  #efsndt                 ;
1870 027462  000137  030606                    jmp     dropunt                 ;drop unit and end pass
1871
1872 027466  022705  000003           fnbr3:  cmp     #3.,r5                  ;test for msg number
1873 027472  001012                            bne     fnbr4                   ;branch if not right number
1874 027474                                    printf  #efcmdt                 ;
1875 027514  000137  030606                    jmp     dropunt                 ;drop unit and end pass
1876
1877 027520  022705  000004           fnbr4:  cmp     #4.,r5                  ;test for msg number
1878 027524  001012                            bne     fnbr5                   ;branch if not right number
1879 027526                                    printf  #efrcvt                 ;
1880 027546  000137  030606                    jmp     dropunt                 ;drop unit and end pass
```

Global subroutines

```
1881
1882 027552  022705  000005          fnbr5:  cmp     #5.,r5          ;test for msg number
1883 027556  001012                          bne     fnbr6           ;branch if not right number
1884 027560                                  printf  #efbust         ;
1885 027600  000137  030606                  jmp     dropunt         ;drop unit and end pass
1886
1887 027604  022705  000006          fnbr6:  cmp     #6.,r5          ;test for msg number
1888 027610  001012                          bne     fnbr7           ;branch if not right number
1889 027612                                  printf  #efinit         ;
1890 027632  000137  030606                  jmp     dropunt         ;drop unit and end pass
1891
1892 027636  022705  000007          fnbr7:  cmp     #7.,r5          ;test for msg number
1893 027642  001012                          bne     fnbr8           ;branch if not right number
1894 027644                                  printf  #efnut          ;
1895 027664  000137  030606                  jmp     dropunt         ;drop unit and end pass
1896
1897 027670  022705  000010          fnbr8:  cmp     #8.,r5          ;test for msg number
1898 027674  001012                          bne     fnbr9           ;branch if not right number
1899 027676                                  printf  #efdxft
1900 027716  000137  030606                  jmp     dropunt         ;drop unit and end pass
1901
1902 027722  022705  000011          fnbr9:  cmp     #9.,r5          ;test for msg number
1903 027726  001012                          bne     fnbr10          ;branch if not right number
1904 027730                                  printf  #effcct         ;
1905 027750  000137  030606                  jmp     dropunt         ;drop unit and end pass
1906
1907 027754  022705  000012          fnbr10: cmp     #10.,r5         ;test for msg number
1908 027760  001012                          bne     fnbr11          ;branch if not right number
1909 027762                                  printf  #efsekt         ;
1910 030002  000137  030606                  jmp     dropunt         ;drop unit and end pass
1911
1912 030006  022705  000013          fnbr11: cmp     #11.,r5         ;test for msg number
1913 030012  001012                          bne     fnbr12          ;branch if not right number
1914 030014                                  printf  #efrcct         ;
1915 030034  000137  030606                  jmp     dropunt         ;drop unit and end pass
1916
1917 030040  022705  000014          fnbr12: cmp     #12.,r5         ;test for msg number
1918 030044  001012                          bne     fnbr13          ;branch if not right number
1919 030046                                  printf  #eflbft         ;
1920 030066  000137  030606                  jmp     dropunt         ;drop unit and end pass
1921
1922 030072  022705  000015          fnbr13: cmp     #13.,r5         ;test for msg number
1923 030076  001012                          bne     fnbr14          ;branch if not right number
1924 030100                                  printf  #effcwt         ;
1925 030120  000137  030606                  jmp     dropunt         ;drop unit and end pass
1926
1927 030124  022705  000016          fnbr14: cmp     #14.,r5         ;test for msg number
1928 030130  001012                          bne     fnbr15          ;branch if not right number
1929 030132                                  printf  #efrcrt         ;
1930 030152  000137  030606                  jmp     dropunt         ;drop unit and end pass
1931
1932 030156  022705  000017          fnbr15: cmp     #15.,r5         ;test for msg number
1933 030162  001012                          bne     fnbr16          ;branch if not right number
1934 030164                                  printf  #efrcwt         ;
1935 030204  000137  030606                  jmp     dropunt         ;drop unit and end pass
1936
1937 030210  022705  000020          fnbr16: cmp     #16.,r5         ;test for msg number
```

Global subroutines

```
1938 030214  001012                    bne     fnbr17                  ;branch if not right number
1939 030216                            printf  #efrcft                 ;
1940 030236  000137  030606            jmp     dropunt                 ;drop unit and end pass
1941
1942 030242  022705  000021  fnbr17:   cmp     #17.,r5                 ;test for msg number
1943 030246  001012                    bne     fnbr18                  ;branch if not right number
1944 030250                            printf  #effcrt                 ;
1945 030270  000137  030606            jmp     dropunt                 ;drop unit and end pass
1946
1947 030274  022705  000022  fnbr18:   cmp     #18.,r5                 ;test for msg number
1948 030300  001012                    bne     fnbr19                  ;branch if not right number
1949 030302                            printf  #effcnt                 ;
1950 030322  000137  030606            jmp     dropunt                 ;drop unit and end pass
1951
1952 030326  022705  000023  fnbr19:   cmp     #19.,r5                 ;test for msg number
1953 030332  001012                    bne     fnbr20                  ;branch if not right number
1954 030334                            printf  #effcdt                 ;
1955 030354  000137  030606            jmp     dropunt                 ;drop unit and end pass
1956
1957 030360  022705  000024  fnbr20:   cmp     #20.,r5                 ;test for msg number
1958 030364  001012                    bne     fnbr21                  ;branch if not right number
1959 030366                            printf  #eftmot                 ;
1960 030406  000137  030606            jmp     dropunt                 ;drop unit and end pass
1961
1962 030412  022705  000025  fnbr21:   cmp     #21.,r5                 ;test for msg number
1963 030416  001012                    bne     fnbr22                  ;branch if not right number
1964 030420                            printf  #efillt                 ;
1965 030440  000137  030606            jmp     dropunt                 ;drop unit and end pass
1966
1967 030444  022705  000026  fnbr22:   cmp     #22.,r5                 ;test for msg number
1968 030450  001012                    bne     fnbr23                  ;branch if not right number
1969 030452                            printf  #efwart                 ;
1970 030472  000137  030606            jmp     dropunt                 ;drop unit and end pass
1971
1972 030476  022705  000027  fnbr23:   cmp     #23.,r5                 ;test for msg number
1973 030502  000412                    br      fnbr24                  ;branch if not right number
1974 030504                            printf  #efinpt                 ;
1975 030524  000137  030606            jmp     dropunt                 ;drop unit and end pass
1976
1978 030530  022705  000030  fnbr24:   cmp     #24.,r5                 ;test for msg number
1979 030534  001012                    bne     1$
1980 030536                            printf  #efmedt
1981 030556  000137  030606            jmp     dropunt                 ;drop unit and end pass
1983 030562  004737  023304  1$:       jsr     pc,typDUPbuf            ;type out ASCII sent by disk-controller
1984 030566  000137  030606            jmp     dropunt                 ;drop unit and end pass
1987 030572                    spcl:
1988 030572                    unkwn:  ERRSF   0,SF0                   ; system error unkown response
1989 030602  004737  023472            jsr     pc,PRNTpkt              ;type out packet information
1991 030606                    dropunt:
1992 030606                            DODU    LOGUNIT                 ;drop the unit
1994 030614                    etst:
1995 030614                            docln                           ;take controller offline
1996 030616                    ENDTST
```

Global subroutines

```
 1998 030620                            BGNHRD
 1999
 2000 030622                                 GPRMA ip.adr,0,0,160000,177776,YES   ;Get IP reg addr (170000 177776)
 2001                                                                             ;place in word 2 of the table
 2002                                                                             ;default value is from default
 2003                                                                             ;table.
 2004
 2005 030632                                 GPRMA vec.adr,2,0,0,776,YES          ;Get the vector addr (octal 0-776)
 2006                                                                             ;place in word
 2007                                                                             ;default value is from default
 2008                                                                             ;table.
 2009
 2010
 2011 030642                                 GPRMD drv.nbr,4,D,-1,0,255.,YES      ;Get the logical drive (dECIMAL 0 255)
 2012                                                                             ;place in word
 2013                                                                             ;default value is from default
 2014                                                                             ;table.
 2015
 2016
 2017 030654                            exit hrd
 2018 030656                             ENDHRD
 2019
 2020
 2021 030656                             LASTAD
      030662                            L$LAST::
 2022 030662                            ENDMOD
 2023          000001                    .END
```

Symbol table

```
A      = 000000          CMDLEN  002376          C$RESE= 000033          EFINIT  017775          F$AU   = 000015
ABORT    025704          CMDPAK  002402          C$REVI= 000003          EFINPT  021153          F$AUTO= 000020
ABRT3    022740          CMDRNG  002462          C$RFLA= 000021          EFLBFT  020360          F$BGN  = 000040
ADR    = 000020 G        CONTON  025550          C$RPT  = 000025          EFMEDT  021174          F$CLEA= 000007
ASCDEC   023176          C$AU   = 000052          C$SEFG= 000046          EFNUT   020040          F$DU   = 000016
ASK.DB   005277          C$AUTO= 000061          C$SPRI= 000041          EFRCCT  020271          F$END  = 000041
ASK.LB   005352          C$BRK  = 000022          C$SVEC= 000037          EFRCFT  020630          F$HARD= 000004
ASK.PR   005156          C$BSEG= 000004          C$TOME= 000076          EFRCRT  020561          F$HW   = 000013
ASK.RB   005425          C$BSUB= 000002          DATARE  002502          EFRCVT  017720          F$INIT= 000006
ASK.XB   005224          C$CLCK= 000062          DECTBL  023162          EFRCWT  020604          F$JMP  = 000050
ASMSGT   005040          C$CLEA= 000012          DEFQUE= 000002          EFSEKT  020252          F$MOD  = 000000
ASMSG1   004331          C$CLOS= 000035          DFBAD   016537          EFSNDT  017641          F$MSG  = 000011
ASMSG2   004374          C$CLP1= 000006          DFCON   016637          EFSTAT  017612          F$PROT= 000021
ASMSG3   004417          C$CPBF= 000074          DFDWN   016607          EFTMOT  020751          F$PWR  = 000017
ASMSG4   004501          C$CPME= 000075          DFPTBL  002240 G        EFUNRG  021217          F$RPT  = 000012
ASMSG5   004551          C$CVEC= 000036          DFQSTN  026572          EFWART  021052          F$SEG  = 000003
ASMSG6   004623          C$DCLN= 000044          DFUNT   016476          EF.CON= 000036 G        F$SOFT= 000005
ASMSG7   004647          C$DODU= 000051          DF1     007326          EF.NEW= 000035 G        F$SRV  = 000010
ASMSG8   004706          C$DRPT= 000024          DF11    007612          EF.PWR= 000034 G        F$SUB  = 000002
ASMSG9   004764          C$DU   = 000053          DF12    007647          EF.RES= 000037 G        F$SW   = 000014
ASSEMB= 000010           C$EDIT= 000003          DF13    007703          EF.STA= 000040 G        F$TEST= 000001
AUTO.M   002777          C$ERDF= 000055          DF14    007757          ELPCMD  025756          GDSCMD  022554
B      - 000006          C$ERHR= 000056          DF15    010040          ELP4    026106          GDS0    021350
BIT0   = 000001 G        C$ERRO= 000060          DF16    010130          END     025706          GDS2    022554
BIT00  = 000001 G        C$ERSF= 000054          DF2     007370          ERSEKO= 000003          GOBIT   022540
BIT01  = 000002 G        C$ERSO= 000057          DF3     007437          ERUDON= 000001          GSTSF   027410
BIT02  = 000004 G        C$ESCA= 000010          DF4     007547          ERUINT= 000002          G$CNTO= 000200
BIT03  = 000010 G        C$ESEG= 000005          DIAGMC= 000000          ETST    030614          G$DELM= 000372
BIT04  = 000020 G        C$ESUB= 000003          DNINT   023106          EVL    = 000004 G       G$DISP= 0C0003
BIT05  = 000040 G        C$ETST= 000001          DO.CON  003120          E$END  = 002100          G$EXCP= 000400
BIT06  = 000100 G        C$EXIT= 000032          DQNBRA  026672          E$LOAD= 000035          G$HILI= 000002
BIT07  = 000200 G        C$FREQ= 000101          DQNBR1  026604          FNBR1   027402          G$LOLI= 000001
BIT08  = 000400 G        C$FRME= 000100          DROPUN  030606          FNBR10  027754          G$NO   = 000000
BIT09  = 001000 G        C$GETB= 000026          DRPUNT  016226          FNBR11  030006          G$OFFS= 000400
BIT1   - 000002 G        C$GETW= 000027          DRVTXA  003154          FNBR12  030040          G$OFSI= 000376
BIT10  = 002000 G        C$GMAN= 000043          DRVTXB  003203          FNBR13  030072          G$PRMA= 000001
BIT11  - 004000 G        C$GPHR= 000042          DRVTXC  004235          FNBR14  030124          G$PRMD= 000002
BIT12  - 010000 G        C$GPRI= 000040          DRVTX0  003277          FNBR15  030156          G$PRML= 000000
BIT13  = 020000 G        C$INIT= 000011          DRVTX1  003373          FNBR16  030210          G$RADA= 000140
BIT14  = 040000 G        C$INLP= 000020          DRVTX2  003467          FNBR17  030242          G$RADB= 000000
BIT15  = 100000 G        C$MANI= 000050          DRVTX3  003563          FNBR18  030274          G$RADD= 000040
BIT15T   024350          C$MAP  = 000102          DRVTX4  003657          FNBR19  030326          G$RADL= 000120
BIT2   = 000004 G        C$MEM  = 000031          DRVTX5  003753          FNBR2   027434          G$RADO= 000020
BIT3   = 000010 G        C$MMU  = 000103          DRVTX6  004047          FNBR20  030360          G$XFER= 000004
BIT4   - 000020 G        C$MSG  = 000023          DRVTX7  004142          FNBR21  030412          G$YES  - 000010
BIT5   = 000040 G        C$OPNR= 000034          DRV.NB  002714          FNBR22  030444          HIPRGI  002500
BIT6   = 000100 G        C$OPNW= 000104          DUPDLG  026530          FNBR23  030476          HOE    = 100000 G
BIT7   = 000200 G        C$PNTB= 000014          DUP.ID= 000002          FNBR24  030530          HRDINT  022040
BIT8   = 000400 G        C$PNTF= 000017          EFBUST  017751          FNBR3   027466          HRDO    010505
BIT9   = 001000 G        C$PNTS= 000016          EFCMDT  017667          FNBR4   027520          IBE    = 010000 G
BOE    = 000400 G        C$PNTX= 000015          EFDXFT  020074          FNBR5   027552          IDU    = 000040 G
BOOT     002260          C$PUTB= 000072          EFFCCT  020163          FNBR6   027604          IER    = 020000 G
BOT.CO   006060          C$PUTW= 000073          EFFCDT  020714          FNBR7   027636          INBRA   027210
BOT.DE   005500          C$QIO  = 000377          EFFCNT  020670          FNBR8   027670          INBRR   027214
BOT.RE   005770          C$RDBU= 000007          EFFCRT  020645          FNBR9   027722          INBR0   027120
CINTR    002452          C$REFG= 000047          EFFCWT  020500          FTLER   027360          INBR1   027154
CLRDUP   023400          C$REL  = 000077          EFILLT  021000          FTLERR= 000005          INFORM= 000003
```

Symbol table

```
INFRM   027112          L$EXP4  002064 G        OP.SRX= 000054          PB1210  014727          RFD6    02.070
INTSRV  025412          L$EXP5  002066 G        O$APTS= 000000          PB1211  014771          RINTR   002454
IPREG   002262          L$HARD  030622 G        O$AU  = 000000          PB1212  015025          RSPCHK  023416
IP.ADR  002636          L$HIME  002120 G        O$BGNR= 000000          PB1213  015102          RSPPAK  002302
ISR   = 000100 G        L$HPCP  002016 G        O$BGNS= 000000          PB1214  015146          RSPRNG  002456
IXE   = 004000 G        L$HPTP  002022 G        O$DU  = 000001          PB1215  015217          RSP1    002276
I$AU  = 000041          L$HW    002240 G        O$ERRT= 000000          PB1216  015260          RW$PLL= 140002
I$AUTO= 000041          L$ICP   002104 G        O$GNSW= 000000          PB1217  015354          R$CMD = 140012
I$CLK = 100006          L$INIT  025436 G        O$POIN= 000001          PB1218  015451          R$DAT = 140010
I$CLN = 000041          L$LADP  002026 G        O$SETU= 000001          PB1219  015526          R$FPS = 140006
I$DU  = 000041          L$LAST  030662 G        PARKDR  005043          PB1220  015565          SDTCMD  026676
I$HRD = 000041          L$LOAD  002100 G        PBF0    011657          PB1221  015652          SDT6    026676
I$INIT= 000041          L$LUN   002074 G        PBF1    011757          PB1222  015721          SER.NB  002742
I$MOD = 000041          L$MREV  002050 G        PBF10   012712          PB1223  016014          SETUP   025456
I$MSG = 000041          L$NAME  002000 G        PBF2    012106          PB13    011567          SFBEGT  017002
I$PROT= 000040          L$PRIO  002042 G        PBF3    012162          PB3     010753          SFCYLT  017466
I$PTAB  000041          L$PROT  025430 G        PBF4    012256          PB4     011021          SFDBBT  017250
I$PWR = 000041          L$PRT   002112 G        PBF5    012321          PB5     011073          SFDONT  017023
I$RPT = 000041          L$REPP  002062 G        PBF6    012366          PB6     011164          SFFCNT  017541
I$SEC = 100016          L$REV   002010 G        PBF7    012463          PB7     011266          SFFCUT  017507
I$SEG = 000041          L$SPC   002056 G        PBF8    012562          PB8     011320          SFRBBT  017453
I$SETU= 000041          L$SPCP  002020 G        PBF9    012652          PB9     011354          SFRCBT  017170
I$SRV = 000041          L$SPTP  002024 G        PBSF0   016160          PF2     011572          SFREVT  017047
I$SUB = 000041          L$STA   002030 G        PB0     010642          PLOC    002252          SFR1T   017071
I$TST = 000041          L$TEST  002114 G        PB1     010671          PNT   = 001000 G        SFR2T   017123
I$UDC = 100002          L$TIML  002014 G        PB10    011416          POLLW   021250          SFTRYT  017410
J$JMP = 000167          L$UNIT  002012 G        PB11    011460          POLLWT  021250          SFT0    010530
LOCAL   002250          L10000  002246          PB11AP  013375          PRGNAM  002626          SFT1    010601
LOE   = 040000 G        L10002  025706          PB11CR  012752          PRI   = 002000 G        SFXBBT  017330
LOGUNI  002246          L10003  025716          PB11EL  013274          PRI00 = 000000 G        SF0     010252
LOPRGI  002476          L10004  025726          PB11EN  013130          PRI01 = 000040 G        SF1     010371
LOT   = 000010 G        L10005  025754          PB11ES  013237          PRI02 = 000100 G        SF100   010432
LSTCMD  002472          L10006  030616          PB11GD  013207          PRI03 = 000140 G        SPCL    030572
LSTCRN  002470          L10007  030656          PB11OP  013022          PRI04 = 000200 G        SPECL - 000006
LSTVCT  002474          MAXDRV- 000004          PB11RD  013350          PRI05 = 000240 G        SP2INT  022162
L$ACP   002110 G        MCDNBR  002274          PB11SD  013326          PRI06 = 000300 G        SP3INT  022252
L$APT   002036 G        MDLNBR  002272          PB11ST  013074          PRI07 = 000340 G        SP4INT  022332
L$AUT   002070 G        MOD1    002000 G        PB11S0  013417          PRK.HD  002670          STDALN- 000001
L$AUTO  025710 G        MRQDX1= 000007          PB11S1  013444          PRNTPK  023472          SVCGBL= 000000
L$CCP   002106 G        MRQDX3= 000023          PB11S2  013476          PS0   = 000000          SVCINS= 177777
L$CLEA  025720 G        MSGNBR= 170000          PB11S3  013534          PS7   = 000340          SVCSUB= 177777
L$CO    002032 G        NEXT    025464          PB11S4  013571          PTBL    002254          SVCTAG= 177777
L$DEPO  002011 G        OCTASC  023110          PB11S5  013625          QFDAT   016445          SVCTST= 177777
L$DESC  002126 G        OP.ABR= 000006          PB11S6  013654          QFSER   016726          S$LSYM= 010000
L$DESP  002076 G        OP.DD = 000001          PB11S9  013701          QFUIT   016370          TBQ0    006132
L$DEVP  002060 G        OP.ELP= 000003          PB11W0  013744          QSTN    026560          TBQ1    006217
L$DISP  002124 G        OP.END= 000200          PB11W1  014030          QUESTI= 000001          TBQ10   006461
L$DLY   002116 G        OP.ESP= 000002          PB12    016131          RCDCMD  026320          TBQ11   006504
L$DTP   002040 G        OP.GDS= 000001          PB1201  014121          RCD5    026320          TBQ12   006533
L$DTYP  002034 G        OP.RD = 000003          PB1202  014205          RD.MOD= 000300          TBQ13   006572
L$DU    025730 G        OP.REC= 000005          PB1203  014272          RETRY = 000367          TBQ14   006604
L$DUT   002072 G        OP.RES= 000000          PB1204  014343          RFDJ4   026210          TBQ15   006623
L$DVTY  002160 G        OP.SEN= 000004          PB1205  014404          RFD0    021516          TBQ16   006634
L$EF    002052 G        OP.SI1= 000005          PB1206  014445          RFD2    022702          TBQ17   006661
L$ENVI  002044 G        OP.SO1= 000007          PB1207  014517          RFD3    023066          TBQ18   006700
L$ETP   002102 G        OP.SRD= 000044          PB1208  014572          RFD4    026256          TBQ19   006717
L$EXP1  002046 G        OP.SRP= 000100          PB1209  014626          RFD5    026512          TBQ2    006241
```

L4
.MAIN. MACRO V05.03 Tuesday 10-Jun 86 13:36 Page 21 3

SEQ 0050

Symbol table

```
TBQ20   006752          TERM    027220          T$LSYM= 010000          T$$CLE= 010004          WRNGST  022500
TBQ21   007002          TERMIN= 000004          T$LTNO= 000001          T$$DU = 010005          W$CMD = 140022
TBQ22   007034          TIMOUT  022440          T$NEST= 177777          T$$HAR= 010007          W$DAT = 140020
TBQ23   007047          TNBRA   027330          T$NS0 = 000000          T$$HW = 010000          W$FPL = 140004
TBQ24   007062          TNBR13  027226          T$NS1 = 000004          T$$INI= 010002          X$ALWA= 000000
TBQ25   007075          TYPASC  016271          T$PTHV= ****** GX       T$$PRO= 010001          X$FALS= 000040
TBQ26   007110          TYPDUP  023304          T$PTNU= 000000          T$$TES= 010006          X$OFFS= 000400
TBQ28   007122          TYPE    = 177760        T$SAVL= 177777          T1      025756 G        X$TRUE= 000020
TBQ29   007152          T$ARGC= 000001          T$SEGL= 177777          UAM     = 000200 G      $2      025642
TBQ3    006263          T$CODE= 001004          T$SIZE= ****** GX       UITADR  002256          $3      025662
TBQ30   007203          T$ERRN= 000000          T$SUBN= 000000          UITOTH= 000010          $4      025676
TBQ31   007231          T$EXCP= 000000          T$TAGL= 177777          UNIT    002266          .A.DEF= 000040
TBQ32   007273          T$FLAG= 000041          T$TAGN= 010010          UNKWN   030572          .A.FAT- 000120
TBQ4    006305          T$FREE= ****** GX       T$TEMP= 000000          UNTFLG  002270          .A.INF= 000060
TBQ5    006327          T$GMAN= 000000          T$TEST= 000001          UNT.NB  005114          .A.QUE= 000020
TBQ6    006351          T$HILI- 000377          T$TSTM= 177777          VECTOR  002264          .A.TER= 000100
TBQ7    006373          T$LAST= 000001          T$TSTS= 000001          VEC.AD  002651           A.TYP= 000020
TBQ8    006415          T$LOLI= 000000          T$$AUT= 010003          WARNIN  003020          .B.SPL= 000140
TBQ9    006437

. ABS.  030662    000   (RW,I,GBL,ABS,OVR)
        000000    001   (RW,I,LCL,REL,CON)
Errors detected:  0

*** Assembler statistics


Work  file  reads: 292
Work  file wr'tes: 306
S'ze of work file: 38376 Words  ( 150 Pages)
Size of core pool: 19684 Words  ( 75 Pages)
Operating  system: RSX 11M/PLUS (Under VAX/VMS)

Elapsed t'me: 00:03:31.26
ZRQFA0,ZRQFA0.LST/-SP=SVC35R/ML,ZRQFA0.MAC
```