

Micro Fiche Scan

Name of device(s) tested:

RD31/51/52/53/54,RQDX3,RX33

Test description:

RQDX3 FORMATTER

MAINDEC Number or Package Identifier (after SEP 1977):

CZRQCC0

Fiche Document Part Number:

AH-U110C-MC

Fiche preparation date unknown, using copyright year:

1986

Image resolution:

1-bit black&white, compressed for minimal file size

COPYRIGHT (C) 1985-86 by d|i|g|i|t|a|l

5-
b
A
1

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36

.REM *C

IDENTIFICATION

PRODUCT CODE: AC U109C MC
PRODUCT NAME: CZRQCCO RQDX3 FORMATTER
PRODUCT DATE: JUNE 6, 1986
MAINTAINER: DIAGNOSTIC ENGINEERING
AUTHOR: Richard Dietz

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.

NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANIES.

COPYRIGHT (C) 1986 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	

38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59

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

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

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

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

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

2.0 HOW TO RUN IT?

2.1 HARDWARE REQUIREMENTS

An RQDX3 disk controller and one or more Winchester or RX33 drives configured into a Q bus PDP 11 system.

2.2 SOFTWARE REQUIREMENTS

This diagnostic was written using DRS the Diagnostic Supervisor. The diagnostic is expected to be run under XXDP diagnostic operating system. 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 and the answers to the initial DRS questions. The Default value for the IP address is 172150. This is standard configuration address for the first MSCP controller on a system. Any other MSCP controllers on the system will have to be in the floating address space of the IO page. The default vector address is 154 any other value between 0 774 could be used but is not suggested. If you want the default answers then just hit the "return" key on the keyboard. The Formatter will run an auto sizer to determine the proper drive characteristic table to give to the controller. This auto sizer will figure out how many cylinders on the drive and through a small look up table we decide which table to down line load to the RQDX3 controller. The user will have to enter a drive number and a serial number. After this a warning message will appear asking if the user wants to proceed. The default is no so the user must type "Y" in order to format his drives.

Typical Diagnostic Script:

```
boot up XXDP
.RUN ZRQC??
ZRQCC0.BIN
```

```
DRSXM-A0
ZRQC-C-0
RQDX3 Disk Park\Format Utility
Unit is RD51,RD52,RD53,RD54,RX33,RD31
Restart Address is 141656
DR>START
```

Please type yes to "Change HW?"

```
Change HW ? Y
# Units ? 1
```

```
IP Address 172150 ? <rtm>
Vector Address 154 ? <rtm>
Just Park heads N ? <rtm>
Logical Drive (0-255) 0 ? <rtm>
Drive Serial Number(1 32000) 12345 ? <rtm>
```

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

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

2.3.2 UIT TABLES

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

NOTE this is only for the RQDX3 disk controller and NOT for the RQDX1/2.

Unit Information Tables listed:

Enter UIT:
 UIT Drive Name

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

2.4 PROGRAM MESSAGES AND FORMAT COMPLETION

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

Completion Report:

xxx	Revectorred LBNs
xxx	Primary revectorred LBNs

```
232      xxx      Secondary/tertiary revectoring LBNs
233      xxx      Bad Blocks in the RCT area due to data errors
234      xxx      Bad Blocks in the DBN area due to data errors
235      xxx      Bad Blocks in the XBN area due to data errors
236      xxx      Blocks retired on check pass
237      FCT was not used
238      Format Completed
239
240      RQDX Drive xxxx finished
241      PLEASE wait .... Parking drive heads
242
243      pass aborted for this unit
244      ZRQC EOP 1
245      0 Cumulative errors
246
247      Note that every time the disk formats successfully the program
248      drops the UNIT. This is purposely done so one doesn't reformat
249      it twice.
250
251
252      RX33 diskette formatting is a little varied in that several extra
253      questions will be asked. These questions were installed mainly to
254      protect the person trying to format a diskette on the same drive as
255      their boot media. If the drive doing the formatting is not the boot
256      drive then please ignore the warnings.
257
258      WARNING Remove boot diskette if in drive.
259      Insert a diskette to be formatted & press <RETURN>.
260
261      Format Complete
262      FCT was not used
263      Format completed
264
265      Do you want to format another diskette?
266
267      If boot drive, reinsert boot diskette & press <RETURN>.
268
269      RQDX Drive xxxx finished
270      pass aborted for this unit
271      ZRQC EOP 1
272      0 Cumulative errors
273
274
275      2.5      EXECUTION TIME
276      The execution time for this diagnostic varies greatly according
277      to the size of the drive being formatted. If an error in the
278      drive configuration or state such as a write protect switch
279      being on, an error will occur right after all the questions have
280      been answered. If there are no errors the formatter will take
281      between 5 minutes to 60 minutes depending on the drive being formatted.
282      A RD51 takes between 10 minutes to format depending on the way
283      questions are answered. A RD52 takes between 10 & 25 minutes to format
284      and a RD53 a very long time to format. The program checks continuously
285      to make sure the controller is still working. If no progress is
286      indicated by the progress indicator a timeout error will occur. If
287      the disk controller goes off line for some unapparent reason the
288      formatter will know. Either way if one checks the light on the
```

289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345

Winchester to see if it is lite or check the READY light of the drive for a flickering light, this will tell the user that the formatter is working. When the formatter completes a "Format complete" message will appear on the terminal.

3. ERRORS

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

error #	Comment	Problem
0,SF0	;unkown response	
	Not a DUP standard local program or Data Error in local program execution.	
1,HRD0	;Fatal DUP type returned	
	Error with Format program check detailed error message more then likely this will be a drive error or drive configuration error. If the detailed message has a GET STATUS error. This means that the drive you asked to format had the wrong status. Example offline,write 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 #,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	

346 Something in system interrupting or late interrupt. This
347 could be the system clock or an interrupt from an IO port.
348 If the interrupt is at address 4,10 probable a software error
349 Try again.
350
351 9,DF12 ;Fatal SA error
352 Controller crashed. check detailed error message either dead
353 controller or configuration error.
354
355 10,DF11 ;Bad response packet
356 Inappropriate command or soft controller error check
357 detail message for more info.
358
359 11,DF13 ;no progress shown after cmd timeout
360 The controller didn't indicate progress which means that it's
361 working very slow or is stuck. Leave the program running for a
362 couple minutes. If this message repeats then the drive is likely
363 broken. If you just get 1 message it is possible the controller
364 took too long to revector a block. This is probable a drive error
365 or a drive with many revector blocks.
366
367 12,DF14 ;no interrupt after get dust status command controller dead
368 The controller got lost. The program running in the controller
369 got out of synch with the host program. This could mean several
370 things. Check for a loose controller board loose cables. Try running
371 again after rebooting the system. If you still get the error check
372 the controller.
373

4. PROGRAM DESIGN AND FLOW

374
375
376 The program is kind of simple. There is only 1 command ring and
377 1 response ring. For every command send there is expected 1 response.
378 If the command sent times out a "Get DUST Status" command is sent to
379 check on the controllers progress. This usually happens when the actual
380 format is being done. The rest of the commands pass information
381 back and forth from the user to the controller and back with out ever
382 timing out. This program is written according to UQSSP and DUP specs.
383 This specs can be acquired from NEWTON::ARCH\$FILES:. At the start of the
384 program the INIT sequence brings the controller into the higher protocol
385 state of running DUP commands. Once initialized the controller executed
386 a GET DUST STATUS command to make sure the controller is in an Idle
387 state.
388

389
390 If idle which it should be the program asks for a program name to run.
391 The EXECUTE LOCAL PROGRAM command is executed which should start the
392 program into the DUP dialog loop. This dialog is described in the DUP
393 spec. Here several SEND DATA and RECEIVE DATA commands are executed to
394 ask questions and supply information on the success and completion of
395 the local FORMAT program running in the RQDX3.
396

397 A pass will occur when the formatter has completed formatting
398 all the logical units.
399

5.0 GLOSSARY

400
401
402

403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452

ZRQCc0 follows the module name format described in the
XXDP Programmer's Guide.

RQ Identifies the hardware and thus the module.

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

c 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 B contains an autosizing routine which will
size the drive instead of having the user pick the drive table.
This will keep people out of the systems and lower the changes
of loose cables etc. Also added a AUTO mode which allows no manual
interventions. Set up the default p-table to format drive 0 3.
Since floppies are always the last drive in the system this is
gaurenteed to format all the drive in the system and error when 't
gets to the floppy.

Revision C contains several changes. First RX33,RD31,RD54 support was
added. The RX33 boot device questions where added. The autosizer was
fixed to also size for floppies. The Autosizer errors are now reported
to the host along with what drives are located on what units and there
drive size or floppy type. The default question in manual mode was
changed so that if an FCT (factory control table) is not present
"Bad Block Information" it will not continue on. This was changed for
all drives except the RD51 which doesn't have a FCT table. Also there
was a small change to the autosizer which affects version C1 hardware
etched RQDX3 boards specially the ones without the LUN ECO. The autosizer
now run in the beg'ning and the end. A head parking feature was added
so that RD31 and RD32 heads would be parked in the inner most cylinder
upon completion of the program. The autosizer utility was updated to
displays a little more information.

)x

K1

.MAIN. MACRO V05.03 Tuesday 10-Jun 86 13:21 Page 5

SEQ 001C

```
454
455      .MCALL SVC
456 000000 SVC
457 000000 .ENABLE ABS,AMA
458      000052 .-52
459 000052 010000      .word      bit12      ;extended monitor in XXDP
460      002000      .=2000
461 002000 BGNMOD MOD1
462 002000 POINTER BGNDU,BGNCLN,BGNPROT,BGNSETUP
463 002000 HEADER ZRQC,C,0,600,0
464 002122 DISPATCH 1
465 002126 DESCRIPT <RQDX3 Format\Park Disk Utility>
466 002166 DEVTYPE <RD51,RD52,RD53,RD31,RD54,RX33 *** Answer "Y" to "Change HW (L) ?" ***>
467
```

LI

469 002274
470 002276 172150
471 002300 000154
472 002302 000000
473 002304 030071
474 002306 100000
475 002310
476

BGNHW DFPTBL
 .WORD 172150
 .WORD 154
 .WORD 000000
 .WORD 012345.
 .word 100000
ENDHW

;IP address
;Vector address
;unit zero as default drive
;serial number
;auto sizer-"yes", warning="no" or don't continue

478 002310

EQUALS

; BIT DIFINITIONS

100000	BIT15==	100000
040000	BIT14==	40000
020000	BIT13==	20000
010000	BIT12==	10000
004000	BIT11==	4000
002000	BIT10==	2000
001000	BIT09==	1000
000400	BIT08==	400
000200	BIT07==	200
000100	BIT06==	100
000040	BIT05==	40
000020	BIT04==	20
000010	BIT03==	10
000004	BIT02==	4
000002	BIT01==	2
000001	BIT00==	1

001000	BIT9==	BIT09
000400	BIT8==	BIT08
000200	BIT7==	BIT07
000100	BIT6==	BIT06
000040	BIT5==	BIT05
000020	BIT4==	BIT04
000010	BIT3==	BIT03
000004	BIT2==	BIT02
000002	BIT1==	BIT01
000001	BIT0==	BIT00

; EVENT FLAG DEFINITIONS

; EF32:EF17 RESERVED FOR SUPERVISOR TO PROGRAM COMMUNICATION

000040	EF.START==	32.	; BIT POSITION IN SECOND STATUS WORD
000037	EF.RESTART==	31.	; (100000) START COMMAND WAS ISSUED
000036	EF.CONTINUE==	30.	; (040000) RESTART COMMAND WAS ISSUED
000035	EF.NEW==	29.	; (020000) CONTINUE COMMAND WAS ISSUED
000034	EF.PWR==	28.	; (010000) A NEW PASS HAS BEEN STARTED
			; (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
479          .sbttl Literals
480
481          ;+
482          ; Mask values to mask out specified flags
483          ; -
484      000010      UITothr = 10          ;UIT other
485                                     ;if UIT doesn't exist
486
487          ;+
488          ; M'sc.
489          ; -
490      000004      MaxDrv  = 4          ;Maximum Number of drives
491      000002      DUP.id  = bit1       ;DUP connection ID
492      000007      Mrqdx1  = 7.        ;model number for RQDX1
493      000023      Mrqdx3  = 19.       ;model number for RQDX3
494      000001      stdaln  = bit0       ;stand alone modifier
495      000367      retry   = 367        ;Number of retries UDC
496
497          ;+
498          ; Opcodes for DUP commands
499          ; -
500      000001      op.gds   = 1
501      000006      op.abrt  = 6
502      000004      op.sen   = 4
503      000005      op.rec   = 5
504      000003      op.elp   = 3
505      000002      op.esp   = 2
506      000200      op.end   = 200
507
508          ;+
509          ; Message type masks
510          ; -
511      000001      Question = 1
512      000002      DefQuest = 2
513      000003      inform   = 3
514      000004      terminat = 4
515      000005      ftlerr   = 5
516      000006      spec1    = 6
517
518      177760      type      = 177760
519      170000      msgnbr    = 170000
520
521          ;+
522          ; Auto sizer literals
523
524          ; Interrupt Service Routines and Priority Levels

```

terals

```

523
524      100002      ;$udc      -      100002      ; Pointer to UDC interrupt handler
525      100006      ;$clk      =      100006      ; Pointer to Clock interrupt handler
526      100016      ;$sec      -      100016      ; Pointer to Sector Done Interrupt hand. r
527      000000      ps0        -      0          ; Allow Any Interrupts
528      000340      ps7        -      340        ; Inhibit Interrupts
529
530      ; CSRs
531
532      140002      rw$pll      -      140002
533      140004      w$fp1      -      140004
534      140006      r$fps      =      140006
535      140010      r$dat      =      140010
536      140012      r$cmd      -      140012
537      140020      w$dat      -      140020
538      140022      w$cmd      -      140022
539
540      ; RECEIVE DATA ASCII reply message types:
541
542      000020      .a.typ      =      20          ; ASCII Message Type Multiplier
543      000020      .a.que      -      1*.a.typ    ; Question
544      000040      .a.def      =      2*.a.typ    ; Default question
545      000060      .a.inf      =      3*.a.typ    ; Information
546      000100      .a.ter      =      4*.a.typ    ; Termination
547      000120      .a.fat      =      5*.a.typ    ; Fatal error
548
549      ; RECEIVE DATA binary message types.
550
551      000140      .b.spl      =      6*.a.typ    ; Special
552
553      ; Status Codes returned by SIZER (Success is zero)
554
555      000001      erudon      =      1          ; UDC Never Done
556      000002      eruint      =      2          ; UDC Never Interrupted
557      000003      ersek0      =      3          ; Couldn't Restore to Cyl 0
558
559      ; UDC Commands
560
561      000000      op.res      =      0          ; Reset 9224
562      000001      op.dd       =      1          ; Deselect Drive
563      000003      op.rd       =      3          ; Restore Drive
564      000005      op.sil      =      5          ; Step In One Cylinder
565      000007      op.sol      =      7          ; Step Out One Cylinder
566      000044      op.srd      =      44         ; Select Winchester Drive
567      000054      op.srx      =      54         ; Select Floppy Drive
568      000100      op.srp      =      100        ; Set Register Pointer
569      000300      rd.mode     =      300        ; RD Mode
570
571

```

Macro Definitions

```

573      .sbt1 Macro Definitions
574
575
576      ;+
577      ;      Execute a GET DUST STATUS command and then check the response.
578      ;
579
580
581      000000
582      000001
583      A=0
584      B=1
585      .MACRO GETDUST
586      B=B+1
587      gdstp \B
588      .ENDM
589
590      .MACRO GDSTMP B
591      .list
592      GDS'B: bit    #bit15,cmdrng+2      ;test ownership of ring make sure we own it
593             bne    GDS'B               ;if we don't own it wait until we do
594             mov    #14.,cmdlen         ;load length of packet to be send
595             movb   #0,cmdlen+2         ;load msg type and credit
596             movb   #dup.id,cmdlen+3    ;load DUP connection ID
597             inc    cmdpak              ;load new CRN
598             clr    cmdpak+2
599             clr    cmdpak+4
600             clr    cmdpak+6
601             mov    #op.gds,cmdpak+10    ;load up opcode
602             clr    cmdpak+12           ;no modifiers
603
604             mov    #RFD'B,@vector      ;New vector place
605             mov    #rspak,rsprng       ;load response packet area into ring
606             mov    #cmdpak,cmdrng      ;load command packet area into ring
607             mov    #140000,RSPRNG+2    ;Port ownership bit.
608             mov    #bit15,CMDRNG+2
609             jsr    pc,POLLWT           ;Go to poll and wait routine.
610
611      ;*****
612
613      RFD'B:
614             add    #6,sp               ;Intr to here.
615             mov    #intsrv,@vector     ;fix stack for interrupt (4), pollwt subrtn (2)
616             jsr    pc,RSPCHK           ;Change vector
617
618             ;Go to routine that will check on
619             ;the response recvd from the mut.
620             ;it will check the cmd ref
621             ;num, the endcode and status.
622
623      .nlist
624      .ENDM

```

Macro Definitions

```

621
622
623      ;+
624      ;      Execute an ABORT command and then checks the response.
625      ; -
626
627
628      .MACRO  ABRT
629      B-B+1
630      abrttmp \B
631      .ENDM
632
633      .MACRO  ABRTTMP B
634      .list
635      ABRT'B: bit    #bit15,cmdrng+2      ;test ownership of ring make sure we own it
636              bne    ABRT'B              ;if we don't own it wait until we do
637              mov     #14.,cmdlen         ;load length of packet to be send
638              movb    #0,cmdlen+2         ;load msg type and credit
639              movb    #dup.id,cmdlen+3    ;load DUP connect'on ID
640              inc     cmdpak              ;load new CRN
641              clr     cmdpak+2
642              clr     cmdpak+4
643              clr     cmdpak+6
644              mov     #op.abrt,cmdpak+10  ;load up opcode
645              clr     cmdpak+12           ;no modifiers
646
647              mov     #RFD'B,@vector      ;New vector place
648              mov     #rsppak,rsprng       ;load response packet area into ring
649              mov     #cmdpak,cmdrng       ;load command packet area into ring
650              mov     #140000,RSPRNG+2    ;Port ownership bit.
651              mov     #bit15,CMDRNG+2
652              jsr     pc,POLLWT           ;Go to poll and wait routine.
653
654      ;*****
655
656      RFD'B:
657              add     #6,sp               ;_ntr to here.
658              mov     #intsrv,@vector     ;fix stack for interrupt (4), pollwt subrtn (2)
659              jsr     pc,RSPCHK           ;Change vector
660                                           ;Go to routine that will check on
661                                           ;the response recvd from the mut.
662                                           ;it will check the cmd ref
663                                           ;num, the endcode and status.
664      .nl'st
        .ENDM

```

Macro Definitions

```

666
667
668      ;+
669      ;      Execute a Send data cmd in dup and then check the response for the proper info
670      ;
671
672      .MACRO SENDDAT SPLACE,SBYTCN      ;Execute a Send Data command
673      B-B+1      ;increment the CRN number
674      sendtmp \B,SPLACE,Sbytcn      ;call variable A,B as if it where a number (\)
675      .ENDM
676
677
678      .MACRO SENDTMP B,Splace,Sbytcnt
679      .list
680      SDT'B: bit #bit15,cmdrng+2      ;test ownership of ring make sure we own it
681      bne SDT'B      ;if we don't own it wait until we do
682      mov #34,cmdlen      ;load lenght of packet to be send
683      movb #0,cmdlen+2      ;load msg type and credit
684      movb #dup.id,cmdlen+3      ;load DUP connection ID
685      inc cmdpak      ;load new CRN
686      clr cmdpak+2
687      clr cmdpak+4
688      clr cmdpak+6
689      mov #op.sen,cmdpak+10      ;load up opcode
690      clr cmdpak+12      ;no modifiers
691      mov Sbytcnt,cmdpak+14
692      clr cmdpak+16
693      mov Splace,cmdpak+20      ;load address of buffer descriptor
694      clr cmdpak+22
695      clr cmdpak+24
696      clr cmdpak+26
697      clr cmdpak+30
698      clr cmdpak+32
699
700      mov #RFD'B,@vector      ;New vector place
701      mov #rsppak,rsprng      ;load response packet area into ring
702      mov #cmdpak,cmdrng      ;load command packet area into ring
703      mov #140000,RSPRNG+2      ;Port ownership bit.
704      mov #bit15,CMDRNG+2
705      jsr pc,POLLWT      ;Go to poll and wait routine.
706
707      ;*****
708
709      RFD'B:
710      add #6,sp      ;Intr to here.
711      mov #intsrv,@vector      ;fix stack for interrupt (4), pollwt subrtn (2)
712      jsr pc,RSPCHK      ;Change vector
713      ;Go to routine that will check on
714      ;the response recvd from the mut.
715      ;it will check the cmd ref
716      ;num, the endcode and status.
717      .nlist
718      .ENDM

```

Macro Definitions

```

719
720
721      ;*
722      ;      Execute a Receive Data command and the check the response.
723      ;
724
725      .MACRO RECVDAT Rplace,Rbytcnt      ;Execute a Send Data command
726      B-B+1                             ;increment the CRN number
727      recvtmp \B,Rplace,Rbytcnt        ;call variable A,B as if it where a number (\)
728      .ENDM
729
730      .MACRO RECVTMP B,RPlace,Rbytcnt
731      .list
732      RCD'B: bit      #b't15,cmdrng+2      ;test ownership of ring make sure we own it
733      bne      RCD'B      ;if we don't own it wait until we do
734      mov      #34,cmdlen      ;load lenght of packet to be send
735      movb     #0,cmdlen+2      ;load msg type and credit
736      movb     #dup.id,cmdlen+3      ;load DUP connection ID
737      inc      cmdpak          ;load new CRN
738      clr      cmdpak+2
739      clr      cmdpak+4
740      clr      cmdpak+6
741      mov      #op.rec,cmdpak+10      ;load up opcode
742      clr      cmdpak+12      ;no modifiers
743      mov      Rbytcnt,cmdpak+14
744      clr      cmdpak+16
745      mov      Rplace,cmdpak+20      ;load address of buffer descr'btor
746      clr      cmdpak+22
747      clr      cmdpak+24
748      clr      cmdpak+26
749      clr      cmdpak+30
750      clr      cmdpak+32
751
752      mov      #RFD'B,@vector      ;New vector place
753      mov      #rspak,rsprng      ;load response packet area into ring
754      mov      #cmdpak,cmdrng      ;load command packet area into ring
755      mov      #140000,RSPRNG+2      ;Port ownership b't.
756      mov      #bit15,CMDRNG+2
757      jsr      pc,POLLWT      ;Go to poll and wait routine.
758
759      ;*****
760
761      RFD'B:
762      add      #6,sp      ;Intr to here.
763      mov      #intsrv,@vector      ;fix stack for interrupt (4), pollwt subrtn (2)
764      jsr      pc,RSPCHK      ;Change vector
765      ;Go to routine that will check on
766      ;the response recvd from the mut.
767      ;it will check the cmd ref
768      ;num, the endcode and status.
769      .nlist
770      .ENDM

```

G2

Macro Definitions

```

772
773
774      ;*
775      ;   Execute a Execute Local Program command and the check the response.
776      ;
777
778
779      .MACRO EXLCPRG Enamadr          ;Execute a Send Data command
780      B-B+1                          ;increment the CRN number
781      elptmp \B,Enamadr             ;call variable A,B as 'f it where a number (\)
782      .ENDM
783
784      .MACRO ELPTMP B,Enamadr
785      .list
786      ELP'B: bit    #bit15,cmdrng+2    ;test ownership of ring make sure we own it
787              bne    ELP'B            ;if we don't own it wait until we do
788              mov     #22,cmdlen       ;load lenght of packet to be send
789              movb    #0,cmdlen+2      ;load msg type and credit
790              movb    #dup.id,cmdlen+3 ;load DUP connection ID
791              inc     cmdpak           ;load new CRN
792              clr     cmdpak+2
793              clr     cmdpak+4
794              clr     cmdpak+6
795              mov     #op.elp,cmdpak+10 ;load up opcode
796              mov     #stdaln,cmdpak+12 ;stand alone modifier
797              mov     #6,r0            ;6 letters transfer
798              mov     #cmdpak+14,r1     ;starting address to place program name
799              mov     #Enamadr,r2      ;start of Program Name
800      rfdj'B: movb    (r2)+,(r1)+      ;add 2 to bycnt then store
801              sob     r0,rfdj'B
802
803              mov     #RFD'B,@vector   ;New vector place
804              mov     #rsppak,rsprng   ;load response packet area into ring
805              mov     #cmdpak,cmdrng   ;load command packet area into ring
806              mov     #140000,RSPRNG+2 ;Port ownership bit.
807              mov     #bit15,CMDRNG+2
808              jsr     pc,POLLWT        ;Go to poll and wait routine.
809
810      ;*****
811
812      RFD'B:      ;Intr to here.
813              add     #6,sp            ;fix stack for interrupt (4). pollwt subrtn (2)
814              mov     #intsrv,@vector  ;Change vector
815              jsr     pc,RSPCHK        ;Go to routine that will check on
816                                          ;the response recvd from the mut.
817                                          ;it will check the cmd ref
818                                          ;num, the endcode and status.
819
820      .nlist
821      .ENDM

```

Macro Definitions

```

823
824
825      ;*      Execute a Eexecute Supplied Program command and the check the response.
826      ;
827
828
829
830      .MACRO  EXCSUPPRG      ;Execute a Supplied program command
831      B-B+1      ;increment the CRN number
832      esptmp \B      ;call variable A,B as if it where a number (\)
833      .ENDM
834
835      .MACRO  ESPTMP  B
836      .list
837      ESP'B:  bit    #b't15,cmdrng+2      ;test ownership of ring make sure we own it
838              bne    ESP'B      ;if we don't own it wait until we do
839              mov     #50,cmdlen      ;load lenght of packet to be send
840              movb    #0,cmdlen+2      ;load msg type and credit value
841              movb    #dup.'d,cmdlen+3  ;load DUP connection ID
842              clr     CMDpak+2
843              clr     CMDpak+4
844              clr     CMDpak+6
845              mov     #op.esp,CMDpak+10  ;load up opcode
846              mov     #0,CMDpak+12      ;no stand alone modifier
847              mov     #<autoend autosz>,cmdpak+14 ;load length of prg into buffer
848              clr     cmdpak+16
849              mov     #autosz,cmdpak+20  ;starting address of downline load prg
850              clr     CMDpak+22
851              clr     CMDpak+24
852              clr     CMDpak+26
853              clr     CMDpak+30
854              clr     CMDpak+32
855              clr     CMDpak+34      ;overlay buffer descriptor
856              clr     CMDpak+36
857              clr     CMDpak+40
858              clr     CMDpak+42
859              clr     CMDpak+44
860              clr     CMDpak+46
861              mov     #RFD'B,@vector    ;New vector place
862              mov     #rsppak,rsprng     ;load response packet area into ring
863              mov     #cmdpak,cmdrng     ;load command packet area into ring
864              mov     #140000,RSPRNG+2  ;Port ownership bit.
865              mov     #bit15,CMDRNG+2
866              jsr     pc,POLLWT          ;Go to poll and wa't routine.
867      ;*****
868      RFD'B:      ;Intr to here.
869              add     #6,sp              ;fix stack for interrupt (4), pollwt subrtn (2)
870              mov     #intsrv,@vector    ;Channe vector
871              jsr     pc,RSPCHK          ;Go to routine that will check on
872              ;the response recvd from the mut.
873              ;it will check the cmd ref
874              ;num, the endcode and status.
875      .nlist
      .ENDM

```

Word & Buffer definitions

```

877
878
879 002310 000000
880 002312 000000
881 002314 000000
882 002316 000000
883 002320 000000
884 002322 000000
885
886
887
888
889
890
891
892 002324 000000
893 002326 000000
894 002330 000000
895 002332 000123
896 002334 177777
897 002336 000000
898
899 002340 000000
900 002342 000000
901 002344 000000
902
903 002346
904 002352
905 002446
906 002452
907
908 002522 000000
909 002524 000000
910 002526 002352
911 002530 140000
912 002532 002452
913 002534 100000
914 002536 177777
915
916 002540 000000
917 002542 000000
918 002544 000000
919 002546 000000
920 002550 000000
921
922
923 002552
924
925 002676
926 002704
927 002705
928 002720
929 002733
930 002746
931
932

      .sbt1 Word & Buffer definitions
LOGUNIT: .WORD          ;logunit number
LOCAL:   .WORD          ;
PLOC:    .WORD          ;p table address
ptbl:    .WORD          ;p table address
UITadr:  .word
BOOT:    .word          ;bootable media

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

IPreg:   .WORD 0          ;Address of the SA and IP registers
Vector:  .word 0
Unit:    .word 0          ;unit number
          .word 123
sernbr:  .word 177777     ;serial number
UNITflgs: .word 0        ;flags, bit15 =auto mode
          ;bit13 =unknown model number, bit12 =park heads only
          ;model number of the controller as returned in step 4
          ;micorcode number of the controller as returned in step 4
          ;th's is a pointer to the correct UIT table

mdlnbr:  .word 0
mcdnbr:  .word 0
UIN:     .word 0

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

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

LSTCRN:  .word 0          ;storage for unreturned command CRN
LSTCMD:  .word 0          ;storage for unreturned command opcode
LSTVCT:  .word 0          ;storage for unreturned command interrupt vector address
LOPRGI:  .word 0          ;Low word of the progress indicator
HIPRGI:  .word 0          ;High word of progress indicator

      ;data area
      .nlist bin
      DATARE: .asciz /*A12345678901234567890123456789012345678901234567890123456789012345678901234567890/
      .even
      PRGnam: .ascii /FORMAT/ ;address of local format program name
      .byte 0 ;null for asciz
      XBN:    .ASCIZ /0123456789/
      DBN:    .ASCIZ /0123456789/
      LBN:    .ASCIZ /0123456789/
      RBN:    .ASCIZ /0123456789/
      .even
      .list bin

```

Word & Buffer definitions

934
 935
 936
 937
 938
 939
 940
 941 002776 002776
 942 002776 177777
 943 003000 003000
 944
 945
 946
 947
 948
 949
 950 003000
 951
 952 003000 000071
 953 003002 000000
 954 003004 000127
 955 003006 000000
 956 003010 052360
 957 003012 000000
 958 003014 000220
 959 003016 000000
 960 003020 000022
 961 003022 000004
 962 003024 000462
 963 003026 000156
 964 003030 000462
 965 003032 000000
 966 003034 000001
 967 003036 000044
 968 003040 000004
 969 003042 040063
 970 003044 022544
 971 003046 000002
 972 003050 000002
 973 003052 000001
 974 003054 000020
 975 003056 000020
 976 003060 000005
 977 003062 000020
 978 003064 000015
 979 003066 000001
 980 003070 000001
 981 003072 000001
 982 003074 000002
 983 003076 000151
 984 003100 000463
 985 003102 000463
 986 000104
 987 003104
 988
 989
 990

.sbt1 DISK UNIT INFORMATION TABLES

;*
 ; The following tables are made up of disk drive parameters which will be
 ; feed to the FORMAT controller local program which will then use the
 ; information to format the drives.

;-2776
 .word 1 ;back door for custom table build
 ;-3000

;*
 ; Unit Information table RD51 Seagate
 ;

UIT0:

;/ *Top of Unit Information table (UIT)
 ;/XBN size (lo wrd) XBN size = 3*(1+sectors_per track)/
 ;/XBN size (hi wrd)/
 ;/DBN size (lo wrd)/
 ;/DBN size (hi wrd)/
 ;/LBN size (lo wrd)/
 ;/LBN size (hi wrd)/
 ;/RBN size (lo wrd)/
 ;/RBN size (hi wrd)/
 ;/Sectors per track/
 ;/Surfaces per unit/
 ;/Cylinders per unit/
 ;/Write precomp cylinder/
 ;/Reduce write current cylinder /
 ;/Seek Rate/
 ;/Use CRC or ECC/
 ;/RCT Size/
 ;/Number of RCT copies/
 ;/Media (lo wrd)/
 ;/Media (hi wrd)/
 ;/Sector Interleave (n-to 1)/
 ;/Surface to Surface Skew/
 ;/Cylinder to Cylinder Skew/
 ;/Gap size 0/
 ;/Gap size 1/
 ;/Gap size 2/
 ;/Gap size 3/
 ;/Sync size/
 ;/MSCP cylinders per Unit/
 ;/MSCP Groups per Cylinder/
 ;/MSCP Tracks per Group/
 ;/Max allowed bad spots per surface/
 ;/Bad spot tolerance (bytes)/
 ;/auto recal cylinder
 ;/auto recal cylinder

UITsiz = .-UIT0
 . = 3000 + UITsiz

;*
 ;

DISK UNIT INFORMATION TABLES

```

991      ;      Unit Information table   RD52 Quantum drive
992      ;
993
994
995 003104      UIT1:
996
997 003104 000066      .word 54.      ;/*Top of Unit Information table (UIT)
998 003106 000000      .word 0      ;/XBN size (lo wrd) XBN size = 3*(1+sectors_per_track)/
999 003110 000122      .word 82.      ;/XBN size (hi wrd)/
1000 003112 000000      .word 0      ;/DBN size (lo wrd)/
1001 003114 166140      .word 60512. ;/DBN size (hi wrd)/
1002 003116 000000      .word 0      ;/LBN size (lo wrd)/
1003 003120 000250      .word 168.    ;/LBN size (hi wrd)/
1004 003122 000000      .word 0      ;/RBN size (lo wrd)/
1005 003124 000021      .word 17.     ;/RBN size (hi wrd)/
1006 003126 000010      .word 8.      ;/Sectors per track/
1007 003130 001000      .word 512.    ;/Surfaces per unit/
1008 003132 000400      .word 256.    ;/Cylinders per unit/
1009 003134 001000      .word 512.    ;/Write precomp cylinder/
1010 003136 000000      .word 0      ;/Reduce write current cylinder /
1011 003140 000001      .word 1      ;/Seek Rate/
1012 003142 000004      .word 4      ;/Use CRC or ECC/
1013 003144 000010      .word 8.      ;/RCT Size/
1014 003146 040064      .word 180100000000110100 ;/Number of RCT copies/
1015 003150 022544      .word 180010010101100100 ;tH4034;/Media (lo wrd)/
1016 003152 000001      .word 1      ;tH2564;/Media (hi wrd)/
1017 003154 000002      .word 2      ;/Sector Interleave (n-to 1)/
1018 003156 000015      .word 13.     ;/Surface to Surface Skew/
1019 003160 000020      .word 16.     ;/Cylinder to Cylinder Skew/
1020 003162 000020      .word 16.     ;/Gap size 0/
1021 003164 000005      .word 5.      ;/Gap size 1/
1022 003166 000050      .word 40.     ;/Gap size 2/
1023 003170 000015      .word 13.     ;/Gap size 3/
1024 003172 000001      .word 1      ;/Sync size/
1025 003174 000001      .word 1      ;/MSCP cylinders per Unit/
1026 003176 000001      .word 1      ;/MSCP Groups per Cylinder/
1027 003200 000012      .word 10.     ;/MSCP Tracks per Group/
1028 003202 000151      .word 105.    ;/Max allowed bad spots per surface/
1029 003204 001000      .word 512.    ;/Bad spot tolerance (bytes)/
1030 003206 001000      .word 512.    ;/auto recal cylinder
1031
1032      003210      . =3000+UITsiz+UITsiz
1033
1034
1035      ;+      Unit Information table   RD52 Atasi
1036      ;
1037      ;-
1038
1039
1040 003210      UIT2:
1041
1042 003210 000066      .word 54.      ;/*Top of Unit Information table (UIT)
1043 003212 000000      .word 0      ;/XBN size (lo wrd) XBN size = 3*(1+sectors_per_track)/
1044 003214 000101      .word 65.      ;/XBN size (hi wrd)/
1045 003216 000000      .word 0      ;/DBN size (lo wrd)/
1046 003220 166140      .word 60512. ;/DBN size (hi wrd)/
1047 003222 000000      .word 0      ;/LBN size (lo wrd)/

```

L2

DISK UNIT INFORMATION TABLES

1048	003224	000250	.word	168.	;/RBN size (lo wrd)/
1049	003226	000000	.word	0	;/RBN size (hi wrd)/
1050	003230	000021	.word	17.	;/Sectors per track/
1051	003232	000007	.word	7.	;/Surfaces per unit/
1052	003234	001205	.word	645.	;/Cylinders per unit/
1053	003236	000500	.word	320.	;/Write precomp cylinder/
1054	003240	001205	.word	645.	;/Reduce write current cylinder /
1055	003242	000000	.word	0	;/Seek Rate/
1056	003244	000001	.word	1	;/Use CRC or ECC/
1057	003246	000004	.word	4	;/RCT Size/
1058	003250	000010	.word	8.	;/Number of RCT copies/
1059	003252	040064	.word	+B0100000000110100	;/H4034;/Media (lo wrd)/
1060	003254	022544	.word	+B0010010101100100	;/H2564;/Media (hi wrd)/
1061	003256	000001	.word	1	;/Sector Interleave (n to-1)/
1062	003260	000002	.word	2	;/Surface to Surface Skew/
1063	003262	000007	.word	7.	;/Cylinder to Cylinder Skew/
1064	003264	000020	.word	16.	;/Gap size 0/
1065	003266	000020	.word	16.	;/Gap size 1/
1066	003270	000005	.word	5.	;/Gap size 2/
1067	003272	000050	.word	40.	;/Gap size 3/
1068	003274	000015	.word	13.	;/Sync size/
1069	003276	000001	.word	1	;/MSCP cylinders per Unit/
1070	003300	000001	.word	1	;/MSCP Groups per Cylinder/
1071	003302	000001	.word	1	;/MSCP Tracks per Group/
1072	003304	000024	.word	20.	;/Max allowed bad spots per surface/
1073	003306	000151	.word	105.	;/Bad spot tolerance (bytes)/
1074	003310	001206	.word	646.	;/auto recal cylinder
1075	003312	001206	.word	646.	;/auto recal cylinder

1076
1077 003314 .-3000*UITsiz+UITsiz+UITsiz

1078
1079 ;+
1080 ; Unit Information table RD53 Micropol's
1081 ;-

1082
1083
1084 003314 UIT3:
1085 ; /*Top of Unit Information table (UIT)
1086 003314 000066 .word 54. ;/XBN size (lo wrd) XBN size = 3*(1+sectors_per track)/
1087 003316 000000 .word 0 ;/XBN size (hi wrd)/
1088 003320 000122 .word 82. ;/DBN size (lo wrd)/
1089 003322 000000 .word 0 ;/DBN size (hi wrd)/
1090 003324 016730 .word 7640. ;/LBN size (lo wrd)/
1091 003326 000002 .word 2. ;/LBN size (hi wrd)/
1092 003330 000430 .word 280. ;/RBN size (lo wrd)/
1093 003332 000000 .word 0 ;/RBN size (hi wrd)/
1094 003334 000021 .word 17. ;/Sectors per track/
1095 003336 000010 .word 8. ;/Surfaces per unit/
1096 003340 002000 .word 1024. ;/Cylinders per unit/
1097 003342 002000 .word 1024. ;/Write precomp cylinder/
1098 003344 002000 .word 1024. ;/Reduce write current cylinder /
1099 003346 000000 .word 0 ;/Seek Rate/
1100 003350 000001 .word 1 ;/Use CRC or ECC/
1101 003352 000005 .word 5 ;/RCT Size/
1102 003354 000010 .word 8. ;/Number of RCT copies/
1103 003356 040065 .word +B0100000000110101 ;/H4035;/Media (lo wrd)/
1104 003360 022544 .word +B0010010101100100 ;/H2564;/Media (hi wrd)/

DISK UNIT INFORMATION TABLES

1105	003362	000001	.word	1	;/Sector Interleave (n-to-1)/
1106	003364	000002	.word	2	;/Surface to Surface Skew/
1107	003366	000010	.word	8.	;/Cylinder to Cylinder Skew/
1108	003370	000020	.word	16.	;/Gap size 0/
1109	003372	000020	.word	16.	;/Gap size 1/
1110	003374	000005	.word	5.	;/Gap size 2/
1111	003376	000050	.word	40.	;/Gap size 3/
1112	003400	000015	.word	13.	;/Sync size/
1113	003402	000001	.word	1	;/MSCP cylinders per Unit/
1114	003404	000001	.word	1	;/MSCP Groups per Cylinder/
1115	003406	000001	.word	1	;/MSCP Tracks per Group/
1116	003410	000040	.word	32.	;/Max allowed bad spots per surface/
1117	003412	000156	.word	110.	;/Bad spot tolerance (bytes)/
1118	003414	002000	.word	1024.	;/auto recal cylinder
1119	003416	002000	.word	1024.	;/auto recal cylinder

1120
1121 003420 .-3000+UITsiz+UITsiz+UITsiz+UITsiz

1122
1123
1124
1125 ;+ Unit Information table RD31 Seagate
1126 ;
1127

1128
1129 003420 UIT4:
1130 ; /*Top of Unit Information table (UIT)
1131 003420 000066 .word 54. ;/XBN size (lo wrd) XBN size = 3*(1+sectors_per_track)/
1132 003422 000000 .word 0 ;/XBN size (hi wrd)/
1133 003424 000016 .word 14. ;/DBN size (lo wrd)/
1134 003426 000000 .word 0 ;/DBN size (hi wrd)/
1135 003430 121160 .word 41584. ;/LBN size (lo wrd)/
1136 003432 000000 .word 0 ;/LBN size (hi wrd)/
1137 003434 000144 .word 100. ;/RBN size (lo wrd)/
1138 003436 000000 .word 0 ;/RBN size (hi wrd)/
1139 003440 000021 .word 17. ;/Sectors per track/
1140 003442 000004 .word 4. ;/Surfaces per unit/
1141 003444 001147 .word 615. ;/Cylinders per unit/
1142 003446 000400 .word 256. ;/Write precomp cylinder/
1143 003450 001147 .word 615. ;/Reduce write current cylinder /
1144 003452 000000 .word 0 ;/Seek Rate/
1145 003454 000001 .word 1 ;/Use CRC or ECC/
1146 003456 000003 .word 3 ;/RCT Size/
1147 003460 000010 .word 8. ;/Number of RCT copies/
1148 003462 040037 .word †B01000000000011111 †H401F;/Media (lo wrd)/
1149 003464 022544 .word †B0010010101100100 †H2564;/Media (hi wrd)/
1150 003466 000001 .word 1 ;/Sector Interleave (n-to 1)/
1151 003470 000002 .word 2 ;/Surface to Surface Skew/
1152 003472 000004 .word 4. ;/Cylinder to Cylinder Skew/
1153 003474 000020 .word 16. ;/Gap size 0/
1154 003476 000020 .word 16. ;/Gap size 1/
1155 003500 000005 .word 5. ;/Gap size 2/
1156 003502 000050 .word 40. ;/Gap size 3/
1157 003504 000015 .word 13. ;/Sync size/
1158 003506 000001 .word 1 ;/MSCP cylinders per Unit/
1159 003510 000001 .word 1 ;/MSCP Groups per Cylinder/
1160 003512 000001 .word 1 ;/MSCP Tracks per Group/
1161 003514 000010 .word 8. ;/Max allowed bad spots per surface/

N2

.MAIN. MACRO V05.03 Tuesday 10-Jun 86 13:21 Page 15 4

SEQ 002r

DISK UNIT INFORMATION TABLES

```

1162 003516 000151      .word 105.      ;/Bad spot tolerance (bytes)/
1163 003520 001147      .word 615.      ;/auto recal cylinder
1164 003522 001150      .word 616.      ;/auto recal cylinder
1165
1166      003524      .=3000+UITsiz+UITsiz+UITsiz+UITsiz+UITsiz
1167
1168
1169      ;+
1170      ;      Unit Information table RD54 Maxtor Drive
1171      ;
1172
1173
1174 003524      UIT5:
1175
1176 003524 000066      .word 54.      ;/*Top of Unit Information table (UIT)
1177 003526 000000      .word 0      ;/XBN size (lo wrd) XBN size = 3*(1+sectors_per_track)/
1178 003530 000000      .word 0      ;/XBN size (hi wrd)/
1179 003532 000000      .word 201.     ;/DBN size (lo wrd)/
1180 003534 137730      .word 0      ;/DBN size (hi wrd)/
1181 003536 000004      .word 137730  ;/LBN size (lo wrd)/
1182 003540 001141      .word 4      ;/LBN size (hi wrd)/
1183 003542 000000      .word 609.     ;/RBN size (lo wrd)/
1184 003544 000021      .word 0      ;/RBN size (hi wrd)/
1185 003546 000017      .word 17.      ;/Sectors per track/
1186 003550 002311      .word 15.      ;/Surfaces per unit/
1187 003552 002311      .word 1225.    ;/Cylinders per unit/
1188 003554 002311      .word 1225.    ;/Write precomp cylinder/
1189 003556 000000      .word 1225.    ;/Reduce write current cylinder /
1190 003560 000001      .word 0      ;/Seek Rate/
1191 003562 000007      .word 1      ;/Use CRC or ECC/
1192 003564 000010      .word 7      ;/RCT Size/
1193 003566 040066      .word 8.      ;/Number of RCT copies/
1194 003570 022544      .word tB0100000000110110 ;tH4036;/Media (lo wrd)/
1195 003572 000001      .word tB0010010101100100 ;tH2564;/Media (hi wrd)/
1196 003574 000002      .word 1      ;/Sector Interleave (n-to 1)/
1197 003576 000010      .word 2      ;/Surface to Surface Skew/
1198 003600 000020      .word 8.      ;/Cylinder to Cylinder Skew/
1199 003602 000020      .word 16.     ;/Gap size 0/
1200 003604 000005      .word 16.     ;/Gap size 1/
1201 003606 000050      .word 5.      ;/Gap size 2/
1202 003610 000015      .word 40.     ;/Gap size 3/
1203 003612 000001      .word 13.     ;/Sync size/
1204 003614 000001      .word 1      ;/MSCP cylinders per Unit/
1205 003616 000001      .word 1      ;/MSCP Groups per Cylinder/
1206 003620 000040      .word 1      ;/MSCP Tracks per Group/
1207 003622 000151      .word 32.     ;/Max allowed bad spots per surface/
1208 003624 002311      .word 105.     ;/Bad spot tolerance (bytes)/
1209 003626 002312      .word 1225.    ;/auto recal cylinder
1210      ;/auto recal cylinder possible on this vendor's
1211      ;/drive mmm
1212      003630      .=3000+UITsiz+UITsiz+UITsiz+UITsiz+UITsiz+UITsiz
1213
1214
1215      ;+
1216      ;      Unit Information table
1217      ;
1218

```

DISK UNIT INFORMATION TABLES

```

1219
1220 003630
1221
1222 003630 000066
1223 003632 000000
1224 003634 000057
1225 003636 000000
1226 003640 016677
1227 003642 000002
1228 003644 000524
1229 003646 000000
1230 003650 000021
1231 003652 000010
1232 003654 002000
1233 003656 002000
1234 003660 002000
1235 003662 000000
1236 003664 000001
1237 003666 000005
1238 003670 000003
1239 003672 040065
1240 003674 022544
1241 003676 000001
1242 003700 000002
1243 003702 000010
1244 003704 000020
1245 003706 000020
1246 003710 000005
1247 003712 000050
1248 003714 000015
1249 003716 000001
1250 003720 000001
1251 003722 000001
1252 003724 000040
1253 003726 000156
1254 003730 002000
1255 003732 002000
1256
1257 003734
1258
1259
1260
1261
1262
1263
1264
1265 003734
1266
1267 003734 000066
1268 003736 000000
1269 003740 000057
1270 003742 000000
1271 003744 016677
1272 003746 000002
1273 003750 000524
1274 003752 000000
1275 003754 000021

UIT6:
.word 54.
.word 0
.word 47.
.word 0
.word 016677
.word 2
.word 340.
.word 0
.word 17.
.word 8.
.word 1024.
.word 1024.
.word 1024.
.word 0
.word 1
.word 5
.word 3
.word +B0100000000110101 ;+H4035;/Media (lo wrd)/
.word +B0010010101100100 ;+H2564;/Media (hi wrd)/
.word 1
.word 2
.word 8.
.word 16.
.word 16.
.word 5.
.word 40.
.word 13.
.word 1
.word 1
.word 1
.word 32.
.word 110.
.word 1024.
.word 1024.

; /*Top of Unit Information table (UIT)
;/XBN size (lo wrd) XBN size = 3*(1+sectors per track)
;/XBN size (hi wrd)/
;/DBN size (lo wrd)/
;/DBN size (hi wrd)/
;/LBN size (lo wrd)/
;/LBN size (hi wrd)/
;/RBN size (lo wrd)/
;/RBN size (hi wrd)/
;/Sectors per track/
;/Surfaces per unit/
;/Cylinders per unit/
;/Write precomp cylinder/
;/Reduce write current cylinder /
;/Seek Rate/
;/Use CRC or ECC/
;/RCT Size/
;/Number of RCT copies/
;/Sector Interleave (n-to-1)/
;/Surface to Surface Skew/
;/Cylinder to Cylinder Skew/
;/Gap size 0/
;/Gap size 1/
;/Gap size 2/
;/Gap size 3/
;/Sync size/
;/MSCP cylinders per Unit/
;/MSCP Groups per Cylinder/
;/MSCP Tracks per Group/
;/Max allowed bad spots per surface/
;/Bad spot tolerance (bytes)/
;/auto recal cylinder
;/auto recal cylinder

.=3000*UITsiz+UITsiz+UITsiz+UITsiz+UITsiz+UITsiz+UITsiz

;+
; Unit Information table
;-

UIT7:
.word 54.
.word 0
.word 47.
.word 0
.word 016677
.word 2
.word 340.
.word 0
.word 17.

; /*Top of Unit Information table (UIT)
;/XBN size (lo wrd) XBN size = 3*(1+sectors_per_track)/
;/XBN size (hi wrd)/
;/DBN size (lo wrd)/
;/DBN size (hi wrd)/
;/LBN size (lo wrd)/
;/LBN size (hi wrd)/
;/RBN size (lo wrd)/
;/RBN size (hi wrd)/
;/Sectors per track/

```

DISK UNIT INFORMATION TABLES

1276	003756	000010	.word	8.	;/Surfaces per unit/
1277	003760	002000	.word	1024.	;/Cylinders per unit/
1278	003762	002000	.word	1024.	;/Write precomp cylinder/
1279	003764	002000	.word	1024.	;/Reduce write current cylinder /
1280	003766	000000	.word	0	;/Seek Rate/
1281	003770	000001	.word	1	;/Use CRC or ECC/
1282	003772	000005	.word	5	;/RCT Size/
1283	003774	000003	.word	3	;/Number of RCT copies/
1284	003776	040065	.word	†B0100000000110101	;/Media (lo wrd)/
1285	004000	022544	.word	†B0010010101100100	;/Media (hi wrd)/
1286	004002	000001	.word	1	;/Sector Interleave (n-to-1)/
1287	004004	000002	.word	2	;/Surface to Surface Skew/
1288	004006	000010	.word	8.	;/Cylinder to Cylinder Skew/
1289	004010	000020	.word	16.	;/Gap size 0/
1290	004012	000020	.word	16.	;/Gap size 1/
1291	004014	000005	.word	5.	;/Gap size 2/
1292	004016	000050	.word	40.	;/Gap size 3/
1293	004022	000015	.word	13.	;/Sync size/
1294	004022	000001	.word	1	;/MSCP cylinders per Unit/
1295	004024	000001	.word	1	;/MSCP Groups per Cylinder/
1296	004026	000001	.word	1	;/MSCP Tracks per Group/
1297	004030	000040	.word	32.	;/Max allowed bad spots per surface/
1298	004032	000156	.word	110.	;/Bad spot tolerance (bytes)/
1299	004034	002000	.word	1024.	;/auto recal cylinder
1300	004036	002000	.word	1024.	;/auto recal cylinder

1301
1302 004040 .-3000*UITsiz*UITsiz*UITsiz*UITsiz*UITsiz*UITsiz*UITsiz*UITsiz

1303
1304
1305
1306 ;+ DEFAULT Unit Information table
1307 ;
1308 ;

1309
1310 004040 UITdf:

1311					;/Top of Unit Information table (UIT)
1312	004040	000066	.word	54.	;/XBN size (lo wrd) XBN size - 3*(1+sectors per track)/
1313	004042	000000	.word	0	;/XBN size (hi wrd)/
1314	004044	000311	.word	201.	;/DBN size (lo wrd)/
1315	004046	000000	.word	0	;/DBN size (hi wrd)/
1316	004050	137710	.word	137710	;/LBN size (lo wrd)/
1317	004052	000004	.word	4	;/LBN size (hi wrd)/
1318	004054	001161	.word	625.	;/RBN size (lo wrd)/
1319	004056	000000	.word	0	;/RBN size (hi wrd)/
1320	004060	000021	.word	17.	;/Sectors per track/
1321	004062	000017	.word	15.	;/Surfaces per unit/
1322	004064	002311	.word	1225.	;/Cylinders per unit/
1323	004066	002311	.word	1225.	;/Write precomp cylinder/
1324	004070	002311	.word	1225.	;/Reduce write current cylinder /
1325	004072	000000	.word	0	;/Seek Rate/
1326	004074	000001	.word	1	;/Use CRC or ECC/
1327	004076	000007	.word	7	;/RCT Size/
1328	004100	000010	.word	8.	;/Number of RCT copies/
1329	004102	040066	.word	†B0100000000110110	;/Media (lo wrd)/
1330	004104	022544	.word	†B0010010101100100	;/Media (hi wrd)/
1331	004106	000001	.word	1	;/Sector Interleave (n-to-1)/
1332	004110	000002	.word	2	;/Surface to Surface Skew/

DISK UNIT INFORMATION TABLES

1333	004112	000015	.word	13.	;/Cylinder to Cylinder Skew/
1334	004114	000020	.word	16.	;/Gap size 0/
1335	004116	000020	.word	16.	;/Gap size 1/
1336	004120	000005	.word	5.	;/Gap size 2/
1337	004122	000050	.word	40.	;/Gap size 3/
1338	004124	000015	.word	13.	;/Sync size/
1339	004126	000001	.word	1	;/MSCP cylinders per Unit/
1340	004130	000001	.word	1	;/MSCP Groups per Cylinder/
1341	004132	000001	.word	1	;/MSCP Tracks per Group/
1342	004134	000012	.word	10.	;/Max allowed bad spots per surface/
1343	004136	000151	.word	105.	;/Bad spot tolerance (bytes)/
1344	004140	002000	.word	1024.	;/auto recal cylinder
1345	004142	002000	.word	1024.	;/auto recal cylinder
1346					

DISK PARAMETER QUESTIONS

```

1348                                     .sbttl DISK PARAMETER QUESTIONS
1349     .nlist bin
1350
1351     ;*
1352     ; P table Questions
1353     ;*
1354
1355 004144 IP.adr: .ASCIZ /IP Address/
1356 004157 vec.adr: .ASCIZ /Vector Address/
1357 004176 prk.hds: .ASCIZ /Just park the heads/
1358 004222 drv.nbr: .ASCIZ /Logical Drive (0 255)/
1359 004250 ser.nbr: .ASCIZ /Drive Serial Number(1 32000)/
1360 004305 auto.md: .ASCIZ /Auto Format Mode/
1361 004326 warning: .ASCIZ /***** WARNING all the data on this drive will be DESTROYED ****/
1362 004425 .byte 0
1363
1364 004426 do.cont: .ASCIZ /Proceed to format the drive/
1365
1366 004462 DrvTxa: .asciz /*N*AUT# Drive Name*N/
1367 004511 DrvTxb: .asciz /*A_-----*N/
1368 004605 DrvTx0: .asciz /*A 0 RD51 *N/
1369 004701 DrvTx1: .asciz /*A 1 RD52 part # 30 21721 02 (1 light on front panel) *N/
1370 004775 DrvTx2: .asciz /*A 2 RD52 part # 30-23227-02 (2 lights on front panel) *N/
1371 005071 DrvTx3: .asciz /*A 3 RD53 *N/
1372 005165 DrvTx4: .asciz /*A 4 RD31 *N/
1373 005261 DrvTx5: .asciz /*A 5 RD54 *N/
1374 005355 DrvTx6: .asciz /*A 6 *N/
1375 005450 DrvTx7: .asciz /*A 7 *N/
1376 005543 DrvTxc: .asciz /*A 10 *N/
1377 005637 ASMSGr: .ASCIZ /*A Unrecognized Drive *N/
1378
1379 005733 ASMSG1: .ASCII /*N*AAUTOSIZER FOUND:/
1380 005757 .ASCIZ /*N*Aut Cyls UIT# Drive Name*N/
1381 006021 ASMSG7: .ASCIZ /*A *D1*A Nonexistent*N/
1382 006066 ASMSG8: .ASCIZ /*A *D1*A RX50 Floppy (UNFORMATABLE)*N/
1383 006152 ASMSG9: .ASCIZ /*A *D1*A RX33 Floppy (FORMATABLE)*N/
1384 006234 ASMSG2: .ASCIZ /*A *D1*A *D4*A /
1385 006257 ASMSG3: .ASCIZ /*N*AAUTOSIZER RETURNED FAILURE STATUS CODE *D1*A:/
1386 006341 ASMSG4: .ASCIZ /*N*A CONTROLLER CHIP NEVER WENT DONE/
1387 006411 ASMSG5: .ASCIZ /*N*A CONTROLLER CHIP NEVER INTERRUPTED/
1388 006463 ASMSG6: .ASCIZ /*N*A SEEK FAILED/
1389 006507 ASMSGT: .ASCIZ /*N/
1390 006512 parkdrv: .ASCIZ /*N*PLEASE wa't .... parking d'sk heads./
1391
1392 006563 Unt.nbr: .ASCIZ /Enter Unit Identifier Table (UIT)/
1393 006625 ask.prg: .ASCIZ /What local program do you want to run/
1394 006673 ask.xbn: .ASCIZ /Enter XBN size in decimal (upto 10 digits)/
1395 006746 ask.dbn: .ASCIZ /Enter DBN size in decimal (upto 10 digits)/
1396 007021 ask.lbn: .ASCIZ /Enter LBN size in decimal (upto 10 digits)/
1397 007074 ask.rbn: .ASCIZ /Enter RBN size in decimal (upto 10 digits)/
1398
1399
1400 007147 bot.dev: .ASCII <15><12>/WARNING - If FLOPPY remove boot diskette if in drive to be formatted and/
1401 007261 .ASCII <15><12>/ insert a diskette to be formatted./
1402 007351 .ASCII <15><12>/ If WINCHESTER check if wrt protect switch (off) & ready switch (on)./
1403 007471 .ASCIZ <15><12>/WARNING - All data on drive will be DESTROYED, do you want to continue?/
1404 007603 bot.rep: .ASCIZ /If boot drive, reinsert boot diskette & press <RETURN>./

```

DISK PARAMETER QUESTIONS

```

1405 007673 bot.con: .ASCIZ <15><12>/Do you want to format another diskette?/
1406
1407 ; Top of Unit Information table (UIT)
1408
1409 007745 TBQ0: .ASCIZ /XBN size (lo wrd) XBN size 3*(1+sectors per_track)/
1410 010032 TBQ1: .ASCIZ /XBN size (hi wrd)/
1411 010054 TBQ2: .ASCIZ /DBN size (lo wrd)/
1412 010076 TBQ3: .ASCIZ /DBN size (hi wrd)/
1413 010120 TBQ4: .ASCIZ /LBN size (lo wrd)/
1414 010142 TBQ5: .ASCIZ /LBN size (hi wrd)/
1415 010164 TBQ6: .ASCIZ /RBN size (lo wrd)/
1416 010206 TBQ7: .ASCIZ /RBN size (hi wrd)/
1417 010230 TBQ8: .ASCIZ /Sectors per track/
1418 010252 TBQ9: .ASCIZ /Surfaces per unit/
1419 010274 TBQ10: .ASCIZ /Cylinders per unit/
1420 010317 TBQ11: .ASCIZ /Write precomp cylinder/
1421 010346 TBQ12: .ASCIZ /Reduce write current cylinder /
1422 010405 TBQ13: .ASCIZ /Seek Rate/
1423 010417 TBQ14: .ASCIZ /Use CRC or ECC/
1424 010436 TBQ15: .ASCIZ /RCT Size/
1425 010447 TBQ16: .ASCIZ /Number of RCT copies/
1426 010474 TBQ17: .ASCIZ /Media (lo wrd)/
1427 010513 TBQ18: .ASCIZ /Media (hi wrd)/
1428 010532 TBQ19: .ASCIZ /Sector Interleave (n-to 1)/
1429 010565 TBQ20: .ASCIZ /Surface to Surface Skew/
1430 010615 TBQ21: .ASCIZ /Cylinder to Cylinder Skew/
1431 010647 TBQ22: .ASCIZ /Gap size 0/
1432 010662 TBQ23: .ASCIZ /Gap size 1/
1433 010675 TBQ24: .ASCIZ /Gap size 2/
1434 010710 TBQ25: .ASCIZ /Gap size 3/
1435 010723 TBQ26: .ASCIZ /Sync size/
1436 010735 TBQ28: .ASCIZ /MSCP cylinders per Unit/
1437 010765 TBQ29: .ASCIZ /MSCP Groups per Cylinder/
1438 011016 TBQ30: .ASCIZ /MSCP Tracks per Group/
1439 011044 TBQ31: .ASCIZ /Max allowed bad spots per surface/
1440 011106 TBQ32: .ASCIZ /Bad spot tolerance (bytes)/
1441
1442 011141 DF1: .ASCIZ /Controller Initialization Timeout/
1443 011203 DF2: .ASCIZ /Controller never advanced to next step/
1444 011252 DF3: .ASCIZ /Controller can not execute local programs or non STD DUP dialog program/
1445 011362 DF4: .ASCIZ /NXM Trap at controllers IP address/
1446 ;DF10: .ASCIZ /No Interrupt occurred after SA polled/
1447 011425 DF11: .ASCIZ /Bad Response Packet returned/
1448 011462 DF12: .ASCIZ /Fatal SA error ctrl offline/
1449 011516 DF13: .ASCIZ /No progress shown after a cmd had timed out/
1450 011572 DF14: .ASCIZ /GET DUST CMD time_out after another CMD time_out/
1451 011653 DF15: .ASCIZ /*N*AFatal error was reported when running local program/
1452 011743 DF16: .ASCIZ /*N*AA Special was reported when running local program don't know how to handle it/
1453 012065 SF0: .ASCIZ /DUP protocol Error, unexpected message/
1454 012134 SF1: .ASCIZ /*N*ASYSTEM is NOT in manual mode/
1455 012175 SF100: .ASCIZ /Unexpected or delayed Controller Interrupt/
1456 012250 HRD0: .ASCIZ /Fatal Format error/
1457 012273 SFT0: .ASCIZ /Controller in an unexpected ACTIVE state/
1458 012344 SFT1: .ASCIZ /Wrong Model Number on controller/
1459 012405 PB0: .ASCIZ /*N*AModel # listed #06/
1460 012434 PB1: .ASCIZ /*N*AEExpected SA step bit #06*A,Received in SA #06/
1461 012516 PB3: .ASCIZ /*N*AAasking for Format Parameter table/

```

DISK PARAMETER QUESTIONS

```

1462 012564 PB4: .ASCIZ /%N%AReceived valid Format Parameter table/
1463 012636 PB5: .ASCIZ /%N%AOn UNIT %06%A, %06 Bad Blks were found during Format/
1464 012727 PB6: .ASCIZ /%N%AOn UNIT %06%A, %06 Bad Blks were found during Verify pass %06/
1465 013031 PB7: .ASCIZ /%N%ADUP Message Type: %06/
1466 013063 PB8: .ASCIZ /%N%ADUP message number: %06/
1467 013117 PB9: .ASCIZ /%N%AMSCP Controller model #: %D3/
1468 013161 PB10: .ASCIZ /%N%A Microcode version #: %D3/
1469 013223 PB11: .ASCIZ /%N%AController is IDLE when it should be ACTIVE running format program/
1470 013332 PB13: .ASCIZ /%N/
1471 013335 PF2: .ASCIZ /%N%N%AFinished local program without procedure error/
1472 013422 PBF0: .ASCIZ /%N%AFormat Parameter table entry at byte %06%A is out of range/
1473 013522 PBF1: .ASCIZ /%N%AFormat Parameter table entry at byte %06%A is incompatible with entry at byte %06/
1474 013651 PBF2: .ASCIZ /%N%AUNIT %06%A does not exist on controller/
1475 013725 PBF3: .ASCIZ /%N%AUNIT %06%A does exist but doesn't respond on controller/
1476 014021 PBF4: .ASCIZ /%N%AUNIT %06%A is write protected /
1477 014064 PBF5: .ASCIZ /%N%AWrite Fault detected on UNIT %06/
1478 014131 PBF6: .ASCIZ /%N%AAtempt to step hd %03%A at cyl %03%A failed on UNIT %06/
1479 014226 PBF7: .ASCIZ /%N%AAtempt to format hd %03%A at cyl %03%A failed on UNIT %06/
1480 014325 PBF8: .ASCIZ /%N%ATo many Bad Blocks total Bad Blocks %06/
1481 014415 PBF9: .ASCIZ /%N%ADisk Controller model : %D3/
1482 014455 PBF10: .ASCIZ /%N%A Microcode version : %D3/
1483 014515 PB11crn: .ASCIZ /%N%AEExpected CRN %06%A,Received CRN %06/
1484 014565 PB11op: .ASCIZ /%N%ACMDpkt Opcode %06%A,RSPpkt Opcode %06/
1485 014637 PB11sts: .ASCIZ /%N%AResponse pkt status %06/
1486 014673 PB11end: .ASCIZ /%N%ANo end bit(200) in response packet endcode/
1487 014752 PB11GDS: .ASCIZ /%N%AGet Dust Status cmd/
1488 015002 PB11ESP: .ASCIZ /%N%AEExecute Supplied Prg cmd/
1489 015037 PB11ELP: .ASCIZ /%N%AEExecute Local Prg cmd/
1490 015071 PB11SD: .ASCIZ /%N%ASend Data cmd/
1491 015113 PB11RD: .ASCIZ /%N%AReceive Data cmd/
1492 015140 PB11AP: .ASCIZ /%N%AAbort Prg cmd/
1493 015162 pb11s0: .ASCIZ /%N%Asts: successful/
1494 015207 pb11s1: .ASCIZ /%N%Asts: Invalid Command/
1495 015241 pb11s2: .ASCIZ /%N%Asts: No Region Available/
1496 015277 pb11s3: .ASCIZ /%N%Asts: No Region Suitable/
1497 015334 pb11s4: .ASCIZ /%N%Asts: Program Not Known/
1498 015370 pb11s5: .ASCIZ /%N%Asts: Load Failure/
1499 015417 pb11s6: .ASCIZ /%N%Asts: Standalone/
1500 015444 pb11s9: .ASCIZ /%N%Asts: Host Buffer Access error/
1501 015507 pb11w0: .ASCIZ /%N%AUknown command OPCode received in timeout loop/
1502 015573 pb11w1: .ASCIZ /%N%AUknown command CRN received in command timeout loop/
1503 015664 pb1201: .ASCIZ /%N%ASA er: Envelope\packet Read (parity or timeout)/
1504 015750 pb1202: .ASCIZ /%N%ASA er: Envelope\packet Write (parity or timeout)/
1505 016035 pb1203: .ASCIZ /%N%ASA er: Controller ROM and RAM parity/
1506 016106 pb1204: .ASCIZ /%N%ASA er: Controller RAM parity/
1507 016147 pb1205: .ASCIZ /%N%ASA er: Controller ROM parity/
1508 016210 pb1206: .ASCIZ /%N%ASA er: Queue Read (parity or timeout)/
1509 016262 pb1207: .ASCIZ /%N%ASA er: Queue Write (parity or timeout)/
1510 016335 pb1208: .ASCIZ /%N%ASA er: Interrupt Master/
1511 016371 pb1209: .ASCIZ /%N%ASA er: Host Access Timeout (higher level protocol dependent)/
1512 016472 pb1210: .ASCIZ /%N%ASA er: Credit Limit Exceeded /
1513 016534 pb1211: .ASCIZ /%N%ASA er: Bus Master Error/
1514 016570 pb1212: .ASCIZ /%N%ASA er: Diagnostic Controller Fatal error/
1515 016645 pb1213: .ASCIZ /%N%ASA er: Instruction Loop Timeout/
1516 016711 pb1214: .ASCIZ /%N%ASA er: Invalid Connection Identifier/
1517 016762 pb1215: .ASCIZ /%N%ASA er: Interrupt Write Error/
1518 017023 pb1216: .ASCIZ /%N%ASA er: MAINTENANCE READ\WRITE Invalid Region Identifier/

```

DISK PARAMETER QUESTIONS

```

1519 017117 pb1217: .ASCIZ /%N%ASA er: MAINTENANCE WRITE Load to non loadable controller/
1520 017214 pb1218: .ASCIZ /%N%ASA er: Controller RA:1 error (non parity)/
1521 017271 pb1219: .ASCIZ /%N%ASA er: INIT sequence error/
1522 017330 pb1220: .ASCIZ /%N%ASA er: High level protocol incompatibility error/
1523 017415 pb1221: .ASCIZ /%N%ASA er: Purge/poll hardware failure/
1524 017464 pb1222: .ASCIZ /%N%ASA er: Mapping Register read error (parity or timeout)/
1525 017557 pb1223: .ASCIZ /%N%ASA er: Attempt to set port data transfer mapping when option not present/
1526 017674 PB12: .ASCIZ /%N%ASA Value (oct) %06/
1527
1528 017723 PBsf0: .ASCIZ /%N%ADUP type %06%A message number %06/
1529 017771 DRPunt: .ASCIZ /%N%N%ARQDX DRIVE %06%A finished./
1530 020032 TYPASC: .ASCIZ /%N%APLEASE TYPE ANSWER to controller question or just <return>/
1531
1532 ;mmm
1533 ;

```

FORMAT Messages

```

1535          .sbt11 FORMAT Messages
1536
1537          ; queries
1538
1539 020131 qfuit: ;.byte 2...b.spl ; Unit Info Table? (spl #2)
1540 020131          .asciz 'N*AEntering UIT*02*A: on drive number *D3*N'
1541 020206 qfdat: ;.byte 0...a.que ; Date? (que #0)
1542 020206          .asciz 'Enter date <MM DD-YYYY>:'
1543 020237 dfunt: ;.byte 1...a.def ; Unit? (def #1)
1544 020237          .asciz 'Enter unit number to format <0>:'
1545 020300 dfbad: ;.byte 4...a.def ; Use Bad? (def #4)
1546 020300          .asciz 'Use existing bad block information <N>:'
1547 020350 dfdwn: ;.byte 5...a.def ; Downline? (def #5)
1548 020350          .asciz 'Use down line load <Y>:'
1549 020400 dfcon: ;.byte 6...a.def ; Continue? (def #6)
1550 020400          .asciz 'Continue if bad block information is inaccessible <N>:'
1551 020467 qfser: ;.byte 7...a.que ; Serial #? (que #7)
1552 020467          .asciz 'Enter non zero serial number <8 10 d'gits>:'
1553 020543 ASK.ANSWER:
1554 020543          .asciz 'ans>'
1555
1556          ; Informational Messages
1557
1558 020550 sfbegt: ;.byte 0...a.inf ; Begin (inf #0)
1559 020550          .asciz 'N*AFFormat Begun'
1560 020571 sfdont: ;.byte 1...a.inf ; Complete (inf #1)
1561 020571          .asciz 'N*AFFormat complete'
1562 020615 sfrevt: ;.byte 2...a.inf ; # of Revectorized LBNS (inf #2)
1563 020615          .asciz 'N* Revectorized LBNS'
1564 020637 sfr1t: ;.byte 3...a.inf ; # of primary ... (inf #3)
1565 020637          .asciz 'N* Primary revectorized LBNS'
1566 020671 sfr2t: ;.byte 4...a.inf ; # of secondary ... (inf #4)
1567 020671          .asciz 'N* Secondary/tertiary revectorized LBNS'
1568 020736 sfrcbt: ;.byte 5...a.inf ; # of Bad RCT blocks ... (inf #5)
1569 020736          .asciz 'N* Bad blocks in the RCT area due to data errors'
1570 021016 sfdbbt: ;.byte 7...a.inf ; # of Bad DBNs ... (inf #7)
1571 021016          .asciz 'N* Bad blocks in the DBN area due to data errors'
1572 021076 sfxbbt: ;.byte 9...a.inf ; # of Bad XBNs ... (inf #9)
1573 021076          .asciz 'N* Bad blocks in the XBN area due to data errors'
1574 021156 sftryt: ;.byte 11...a.inf ; # of Retries (inf #11)
1575 021156          .asciz 'N* Blocks retried on the check pass'
1576 021221 sfrbbt: ;.byte 14...a.inf ; # of Bad RBNS ... (inf #14)
1577 021221          .asciz 'N* Bad RBNS'
1578 021234 sfcylt: ;.byte 15...a.inf ; Formatting Cyl (inf #15)
1579 021234          .asciz 'Formatting Cyl *'

```

FORMAT Messages

```

1581
1582      ; Successful Termination Messages
1583
1584      ;.byte      12...a.te      ; Reformat Worked (ter #12)
1585 021255  sffcut: .asciz 'N%AFCT used successfully'
1586      ;.byte      13...a.ter      ; Reconstruct Worked (ter #13)
1587 021307  sffcnt: .asciz 'N%AFCT wa. not used'
1588 021333      .asciz 'N%AFFormat completed'
1589      ; Error messages
1590
1591 021360  efstat: ;.byte 1...a.fat      ; Status Error (fat #1)
1592 021360      .asciz 'N%AGET STATUS failure'
1593
1594 021407  efsndt: ;.byte 2...a.fat      ; Send Error (fat #2)
1595 021407      .asciz 'N%AQ PORT send error'
1596
1597 021435  efcmdt: ;.byte 3...a.fat      ; Command Error (fat #3)
1598 021435      .asciz 'N%AUnsuccessful command'
1599
1600 021466  efrcvt: ;.byte 4...a.fat      ; Receive Error (fat #4)
1601 021466      .asciz 'N%AQ PORT receive error'
1602
1603 021517  efbust: ;.byte 5...a.fat      ; Bus Error (fat #5)
1604 021517      .asciz 'N%AQ-Bus I/O error'
1605
1606 021543  efinit: ;.byte 6...a.fat      ; Format Init Error (fat #6)
1607 021543      .asciz 'N%AFormatter 'n'tial'ization error'
1608
1609 021606  efnut:  ;.byte 7...a.fat      ; Unit nonexistent error (fat #7)
1610 021606      .asciz 'N%ANonexistent unit number'
1611
1612 021642  efdxft: ;.byte 8...a.fat      ; DBN/XBN Format error (fat #8)
1613 021642      .asciz 'N%ADB N/XBN format error (drive FORMAT command failed)'
1614
1615 021731  effcct: ;.byte 9...a.fat      ; FCT copies error (fat #9)
1616 021731      .asciz 'N%AFCT does not have enough good copies of each block'
1617
1618 022020  efsekt: ;.byte 10...a.fat      ; Seek error (fat #10)
1619 022020      .asciz 'N%ASEEK error'
1620
1621 022037  efrcct: ;.byte 11...a.fat      ; RCT copies error (fat #11)
1622 022037      .asciz 'N%ARCT does not have enough good copies of each block'
1623
1624 022126  eflbft: ;.byte 12...a.fat      ; LBN format error (fat #12)
1625 022126      .asciz 'N%ALBN format error (drive FORMAT command failed)'
1626
1627 022211  effcwt: ;.byte 13...a.fat      ; FCT write error (fat #13)
1628 022211      .asciz 'N%AFCT write error (check write protect switch)'
1629
1630 022272  efrcrt: ;.byte 14...a.fat      ; RCT read error (fat #14)
1631 022272      .asciz 'N%ARCT read error'
1632
1633 022315  efrcwt: ;.byte 15...a.fat      ; RCT write error (fat #15)
1634 022315      .asciz 'N%ARCT write error'
1635
1636 022341  efrcft: ;.byte 16...a.fat      ; RCT full error (fat #16)
1637 022341      .asciz 'N%ARCT full'
  
```

FORMAT Messages

```

1638
1639 022356 effcnt: ;.byte 17...a.fat ; FCT read error (fat #17)
1640 022356 .asciz 'N%AFCT read error'
1641
1642 022401 effcnt: ;.byte 18...a.fat ; FCT nonexistent error (fat #18)
1643 022401 .asciz 'N%AFCT nonexistent'
1644
1645 022425 effcdt: ;.byte 19...a.fat ; FCT downline load error (fat #19)
1646 022425 .asciz 'N%AFCT Down line load error'
1647
1648 022462 eftmot: ;.byte 20...a.fat ; Drive t'meout error (fat #20)
1649 022462 .asciz 'N%ADrive in't timeout'
1650
1651 022511 efillt: ;.byte 21...a.fat ; Illegal response error (fat #21)
1652 022511 .asciz 'N%AIlegal response to start up quest'on'
1653
1654 022563 efwart: ;.byte 22...a.fat ; Head error (fat #22)
1655 022563 .asciz 'N%AWARNING possible head addressing problem run diagnostics
1656
1657 022664 efinpt: ;.byte 23...a.fat ; Input error (fat #23)
1658 022664 .asciz 'N%AINPUT Error '
1659
1660 022705 efmedt: ;.byte 24...a.fat ; Media error (fat #24)
1661 022705 .asciz 'N%AMedia degraded'
1662
1663 022730 efunrg: ;.byte 1...a.fat ; Status Error (fat #1)
1664 022730 .asciz 'N%AUncogonized drive'
1665
1666 .list bin
1667 .even

```

Global subroutines

[illegible]

Global subroutines

```

1726 023022 106427 000340 mtps #340 ;don't want interrupts while setting up for cmd
1727 023026 004737 031606 jsr pc,BIT15T ;test SA make sure not a fatal error
1728 023032 013700 002462 mov cmdpak+10,r0 ;get opcode
1729 023036 022700 000001 cmp #op.gds,r0 ; if the command issued was a GETDUST STATUS and time
out big trouble
1730 023042 001006 bne GDS0 ;if not go do a GET DUST to find out what the situat
ion is
1731 023044 ERRDF 12,df14 ;type no interrupt after get dust status command cont
roller dead
1732 023054 000137 037420 jmp dropunt ;drop unit and go on
1733 ;GETDUST ;save timed out command information
1734
1735
1736 023060 017737 157242 002544 GDS0: mov @vector,LSTVCT ;store the vector address of timeout command
1737 023066 013737 002452 002540 mov cmdpak,LSTCRN ;store the CRN of the timed out command
1738 023074 013737 002462 002542 mov cmdpak+10,LSTCMD ;store the opcode of timed out command
1739
1740 023102 032737 100000 002534 bit #b't15,cmdrng+2 ;test ownership of ring make sure we own it
1741 023110 001363 bne GDS0 ;if we don't own it wait until we do
1742 023112 012737 000016 002446 mov #14.,cmdlen ;load lenght of packet to be send
1743 023120 112737 000000 002450 movb #0,cmdlen+2 ;load msg type and credit
1744 023126 112737 000002 002451 movb #dup.id,cmdlen+3 ;load DUP connection ID
1745 023134 005237 002452 inc cmdpak ;load new CRN
1746 023140 005037 002454 clr cmdpak+2
1747 023144 005037 002456 clr cmdpak+4
1748 023150 005037 002460 clr cmdpak+6
1749 023154 012737 000001 002462 mov #op.gds,cmdpak+10 ;load up opcode
1750 023162 005037 002464 clr cmdpak+12 ;no modifiers
1751
1752 023166 012777 023226 157132 mov #RFD0,@vector ;NEW VECTOR PLACE
1753 023174 012737 002352 002526 mov #rsppak,rsprng ;load response packet area into ring
1754 023202 012737 002452 002532 mov #cmdpak,cmdrng ;load command packet area into ring
1755 023210 012737 140000 002530 mov #140000,RSPRNG+2 ;PORT OWNERSHIP BIT.
1756 023216 012737 100000 002534 mov #bit15,CMDRNG+2
1757 023224 000655 br POLLWT ;GO and wait for interrupt
1758
1759
1760 ;+
1761 ; There is only 3 ways out code.
1762 ;If GETDUST resposne and TIMED_OUT cmd response was handled
1763 ; if LSTCRN = 0 and RSPPAK+10 = OP.GDS+OP.END then
1764 ; back to DUP dialog mode.
1765 ;or
1766 ; (TIMED_OUT cmd still hasn't returned but GETDUST has returned)
1767 ; if LSTCRN = # and RSPPAK+10 = OP.GDS+OP.END then
1768 ; check if idle or active. if idle then error
1769 ; check for progress in progress indicator if no progress then error
1770 ; load LSTVCT into @vector,LSTCRN into cmdpak, LSTCMD into cmdpak+10
1771 ; set response ring ownership to Port Owned
1772 ; jmp to pollwt.
1773 ;or
1774 ; (TIMED_OUT cmd response recieved before GETDUST response returned)
1775 ; if LSTCRN = # and RSPPAK+10 not= OP.GDS+OP.END then
1776 ; clear LSTCRN and
1777 ; jmp to pollwt.
1778 ;+
1779
1780 023226 RFD0: ;INTR TO HERE if GETDUST or TIMED_OUT cmd
1781 023226 106427 000340 mtps #340 ;No interrupts please
1782 023232 062706 000004 add #4,sp ;fix stack 4 for intrpt

```

Global subroutines

```

1783 023236 013701 002452      mov    cmdpak,r1      ;check command packet CRN
1784 023242 013700 002352      mov    rsppak,r0      ;check response packet CRN
1785 023246 020001              cmp    r0,r1      ;Are they the SAME must be GETDUST cmd
1786 023250 001103              bne    3$      ;if not it must be the TIMED_OUT cmd
1787
1788 023252 023727 002362 000201  cmp    rsppak+10,#op.gds+op.end      ;it should be a GETDUST lets make sure
1789 023260 001412              beq    1$
1790 023262              printf  #pb11w0      ;unexpected cmd response in time out loop
1791 023302 000137 037404              jmp    unkwn      ;error handler
1792
1793 023306 004737 030654          1$:  jsr    pc,RSPCHK      ;check the response
1794 023312 005737 002540          tst    LSTCRN      ;see if timed out command was already recieved (lstc
rn = 0)
1795 023316 001004              bne    2$
1796 023320 062706 000002          add    #2,sp      ;adjust stack for Timed Out cmd's initial call to P0
LLWT
1797 023324 000137 034100          jmp    DUPDLG      ;if Timed out cmd was already received then goto DUP
dialog mode
1798
1799 023330          2$:              ;if Timed out command was not received already (LSTC
RN not= 0)
1800 023330 132737 000010 002371  bitb    #bit3,rsppak+17      ;if server idle then error
1801 023336 001010              bne    1002$      ;if not check for progress
1802 023340              printf  #pb11      ;controller idle when it should be active
1803
1804 023360 013700 002372          1002$:  mov    rsppak+20,r0      ;check for progress in progress indicator
1805 023364 013701 002374          mov    rsppak+22,r1
1806 023370 020037 002546          cmp    r0,loprgi      ;see if low word of progress indicator is the same a
s older value
1807 023374 001007              bne    1001$      ;if it is then continue
1808 023376 020137 002550          cmp    r1,hiprgi      ;see if high vaule is the same
1809 023402 001004              bne    1001$
1810 023404              ERRDF  11,DF13      ;no progress shown after cmd timeout
1811
1812 023414 010037 002546          1001$:  mov    r0,loprgi      ;update progress indicator
1813 023420 010137 002550          mov    r1,hiprgi
1814 023424 013737 002540 002452  mov    LSTCRN,cmdpak      ;move TIMED_OUT cmd CRN into cmd
1815 023432 013737 002542 002462  mov    LSTCMD,cmdpak+10      ;move TIMED_OUT cmd Opcode into cmd
1816 023440 013777 002544 156660  mov    LSTVCT,@vector      ;load TIMED_OUT cmd interrupt handler address into v
ector
1817 023446 012737 140000 002530  mov    #140000,RSPRNG+2      ;Port owned
1818 023454 000137 022760          jmp    pollw      ;wait for TIMED OUT cmd response
1819
1820
1821
1822 023460 020037 002540          3$:  cmp    r0,LSTCRN      ;check the crn with the last CRN from the timeout co
mmand
1823 023464 001412              beq    4$
1824 023466              printf  #pb11w1      ;Unexpected cmd response in time out loop
1825 023506 000137 037404              jmp    unkwn      ;error handler
1826
1827              ;Timed out command recieved but Get Dust Status is s
till in Queue
1828 023512 013737 002540 002452  4$:  mov    LSTCRN,cmdpak      ;load timed out command values for RSPCHK routine
1829 023520 013737 002542 002462  mov    LSTCMD,cmdpak+10      ;load timed out command values for RSPCHK routine
1830 023526 005037 002540          clr    LSTCRN      ;if it is the timeout command clear LAST CRN registe
r
1831 023532 004737 030654          jsr    pc,RSPCHK      ;go check the command
1832 023536 012737 140000 002530  mov    #140000,RSPRNG+2      ;PORT OWNERSHIP BIT.
1833 023544 000137 022760          jmp    POLLW      ;go wait for GETDUST interupt

```

[illegible]

[illegible]

Global subroutines

```

1949 024210          ERRDF 4,DF2          ; DEVICE FATAL wrong step bit set after interrupt
1950 024220          Printf #pb1,r3,(r4) ; Expected SA step bit xxxxx, received in SA yyyyyy
1951 024244 000137 037420 jmp dropunt ;drop unit and go on
1952
1953 024250          GOBIT:
1954 024250 012714 000001 mov #1,(r4)          ;Controller is NOW INITIALIZED
1955 024254 012700 177777 mov #1,r0
1956 024260 000240 1$: nop          ;waste just a little time so program can terminate
1957 024262 077002 sob r0,1$
1958 024264          GDScmd:
1959 024264 GETDUST          ;Do a Get Dust Status command start things off
024264 032737 100000 002534 GDS2: bit #bit15,cmdrng+2 ;test ownership of ring make sure we own it
024272 001374 bne GDS2          ;if we don't own it wait until we do
024274 012737 000016 002446 mov #14.,cmdlen ;load length of packet to be send
024302 112737 000000 002450 movb #0,cmdlen+2 ;load msg type and credit
024310 112737 000002 002451 movb #dup.id,cmdlen+3 ;load DUP connection ID
024316 005237 002452 inc cmdpak ;load new CRN
024322 005037 002454 clr cmdpak+2
024326 005037 002456 clr cmdpak+4
024332 005037 002460 clr cmdpak+6
024336 012737 000001 002462 mov #op.gds,cmdpak+10 ;load up opcode
024344 005037 002464 clr cmdpak+12 ;no modifiers

024350 012777 024412 155750 mov #RFD2,@vector ;New vector place
024356 012737 002352 002526 mov #rsppak,rsprng ;load response packet area into ring
024364 012737 002452 002532 mov #cmdpak,cmdrng ;load command packet area into ring
024372 012737 140000 002530 mov #140000,RSPRNG+2 ;Port ownership bit.
024400 012737 100000 002534 mov #bit15,CMDRNG+2
024406 004737 022760 jsr pc,POLLWT ;Go to poll and wait routine.

;*****

024412          RFD2:
024412 062706 000006 add #6,sp ;Intr to here.
024416 012777 032650 155702 mov #intsrv,@vector ;fix stack for interrupt (4), pollwt subrtn (2)
024424 004737 030654 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.
;is this server active already
;branch to Execute Local Program
;Soft Error "already active" will do an ABORT cmd
;Doing an ABRT do get into 'dle state
;test ownership of ring make sure we own it
;if we don't own it wait until we do
;load length of packet to be send
;load msg type and credit
;load DUP connection ID
;load new CRN

1960 024430 132737 000010 002371 bitb #bit3,rsppak+17
1961 024436 001467 beq dnint
1962 024440 ERRSOFT 3,SFT0
1963 024450          ABRT
024450 032737 100000 002534 ABRT3: bit #bit15,cmdrng+2
024456 001374 bne ABRT3
024460 012737 000016 002446 mov #14.,cmdlen
024466 112737 000000 002450 movb #0,cmdlen+2
024474 112737 000002 002451 movb #dup.id,cmdlen+3
024502 005237 002452 inc cmdpak
024506 005037 002454 clr cmdpak+2
024512 005037 002456 clr cmdpak+4
024516 005037 002460 clr cmdpak+6
024522 012737 000006 002462 mov #op.abrt,cmdpak+10 ;load up opcode
024530 005037 002464 clr cmdpak+12 ;no modifiers

024534 012777 024576 155564 mov #RFD3,@vector ;New vector place
024542 012737 002352 002526 mov #rsppak,rsprng ;load response packet area into ring

```

Global subroutines

024550	012737	002452	002532	mov	#cmdpak,cmdrng	;load command packet area into ring
024556	012737	140000	002530	mov	#140000,RSPRNG+2	;Port ownership bit.
024564	012737	100000	002534	mov	#bit15,CMDRNG+2	
024572	004737	022760		jsr	pc,POLLWT	;Go to poll and wait routine.

024576				RFD3:		;Intr to here.
024576	062706	000006		add	#6,sp	;fix stack for interrupt (4), pollwt subrtn (2)
024602	012777	032650	155516	mov	#intsrvc,@vector	;Change vector
024610	004737	030654		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.
						;branch back to make sure not busy

1964	024614	000623		br	GDScmd
1965	024616				
1966	024616	000207			
1967					

DNINT: rts pc

Global subroutines

```

1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981 024620
1982 024620 032737 100000 002534
024620 001374
024626 012737 000050 002446
024630 112737 000000 002450
024636 112737 000002 002451
024644 005037 002454
024652 005037 002456
024656 005037 002460
024662 012737 000002 002462
024666 012737 000000 002464
024674 012737 001204 002466
024702 005037 002470
024710 012737 025270 002472
024714 005037 002474
024722 005037 002476
024726 005037 002500
024732 005037 002502
024736 005037 002504
024742 005037 002506
024746 005037 002510
024752 005037 002512
024756 005037 002514
024762 005037 002516
024766 005037 002520
024772 012777 025040 155322
024776 012737 002352 002526
025004 012737 002452 002532
025012 012737 140000 002530
025020 012737 100000 002534
025026 004737 022760
025034
025040
025040 062706 000006
025044 012777 032650 155254
025052 004737 030654
1983 025056
025056 032737 100000 002534
025064 001374
025066 012737 000034 002446
025074 112737 000000 002450
025102 112737 000002 002451
025110 005237 002452

;*****
;      AUTOSizer
;      This routine runs the Execute Supplied program called AUTOSZ
;      This program is downline loaded into the controller to determine
;      which drive is out in the controller. First you must tell which drive
;      you want to format. After listing the drive number the program will load
;      the program and figure which DEC drive it is and which UIT to load into
;      the disk controller for the format program.
;*****
AUTOSizer:
  excSUPprg      ;downline load the program autosz
  ESP4: bit      #bit15,cmdrng+2      ;test ownership of ring make sure we own it
        bne      ESP4                ;if we don't own it wait until we do
        mov      #50,cmdlen           ;load length of packet to be send
        movb     #0,cmdlen+2          ;load msg type and credit value
        movb     #dup.id,cmdlen+3     ;load DUP connection ID
        clr      CMDpak+2
        clr      CMDpak+4
        clr      CMDpak+6
        mov      #op.esp,CMDpak+10    ;load up opcode
        mov      #0,CMDpak+12         ;no stand alone modifier
        mov      #<autoend-autosz>,cmdpak+14 ;load length of prg into buffer
        clr      cmdpak+16
        mov      #autosz,cmdpak+20    ;starting address of downline load prg
        clr      CMDpak+22
        clr      CMDpak+24
        clr      CMDpak+26
        clr      CMDpak+30
        clr      CMDpak+32
        clr      CMDpak+34           ;overlay buffer descriptor
        clr      CMDpak+36
        clr      CMDpak+40
        clr      CMDpak+42
        clr      CMDpak+44
        clr      CMDpak+46
        mov      #RFD4,@vector        ;New vector place
        mov      #rspak,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.
;*****
;      RFD4:
;      add      #6,sp
;      mov      #intsrv,@vector
;      jsr      pc,RSPCHK
;      Intr to here.
;      fix stack for interrupt (4), pollwt subrtn (2)
;      Change vector
;      Go to routine that will check on
;      the response recvd from the mut.
;      get results of auto size
;      test ownership of ring make sure we own it
;      if we don't own it wait until we do
;      load length of packet to be send
;      load msg type and credit
;      load DUP connection ID
;      load new CRN
  Recvdata      #msg,#msglen
  RCD5: bit      #bit15,cmdrng+2
        bne      RCD5
        mov      #34,cmdlen
        movb     #0,cmdlen+2
        movb     #dup.id,cmdlen+3
        inc      cmdpak

```

G4

.MAIN. MACRO V05.03 Tuesday 10 Jun 86 13:21 Page 21 1

SEQ 0045

Global subroutines

```

025114 005037 002454      clr      cmdpak+2
025120 005037 002456      clr      cmdpak+4
025124 005037 002460      clr      cmdpak+6
025130 012737 000005 002462  mov      #op.rec,cmdpak+10      ;load up opcode
025136 005037 002464      clr      cmdpak+12      ;no modifiers
025142 012737 000014 002466  mov      #msglen,cmdpak+14
025150 005037 002470      clr      cmdpak+16
025154 012737 026460 002472  mov      #msg,cmdpak+20      ;load address of buffer descriptor
025162 005037 002474      clr      cmdpak+22
025166 005037 002476      clr      cmdpak+24
025172 005037 002500      clr      cmdpak+26
025176 005037 002502      clr      cmdpak+30
025202 005037 002504      clr      cmdpak+32

025206 012777 025250 155112  mov      #RFD5,@vector      ;New vector place
025214 012737 002352 002526  mov      #rsppak,rsprng      ;load response packet area into ring
025222 012737 002452 002532  mov      #cmdpak,cmdrng      ;load command packet area into ring
025230 012737 140000 002530  mov      #140000,RSPRNG+2      ;Port ownership bit.
025236 012737 100000 002534  mov      #b't15.CMDRNG+2
025244 004737 022760      jsr      pc,PULLWT      ;Go to poll and wait routine.

```

;*****

```

025250                                RFD5:      ;Intr to here.
025250 062706 000006 155044      add      #6,sp      ;fix stack for interrupt (4). pollwt subrtn (2)
025254 012777 032650      mov      #intsrv,@vector      ;Change vector
025262 004737 030654      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.
1984 025266 000207      rts      pc      ;return

```

Global subroutines

```

1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036 025270
2037
2038 025270 001204
2039 025272 000000
2040 025274 000000
2041 025276 000000
2042 025300 101 125 124

```

```

.sbt1 AUTOSZ
;*****
;          AUTOSZ
;      This is the actual down line loaded code which is placed in
;      the RAM inside the RQDX3 controller. This code figures out the
;      cylinder size of the drive. From the cylinder size we can determine
;      which drive it is. If the drive is a winchester we will step the drive
;      into the inner most cylinder. The inner most cylinder for most drives
;      is the parking cylinder.
;
;+
; AUTOSZ Determine Drive Type and Size
; Input:      None.
; Output:
;      A Special Type Message:
;
;      +-----+
;      | Special Msg #10 (decimal) | +00
;      +-----+
;      | Status | +02
;      +-----+
;      | Innermost Cylinder for Unit 0 | +04
;      +-----+
;      | Innermost Cylinder for Unit 1 | +06
;      +-----+
;      | Innermost Cylinder for Unit 2 | +10
;      +-----+
;      | Innermost Cylinder for Unit 3 | +12
;      +-----+
;
;      where, status = 0 for success,
;                    1 for UDC never went done,
;                    2 for UDC never interrupted,
;                    3 for Seek Failed
;
;      cylinder = 3 for RX33 Floppy
;                2 for RX50 Floppy
;                0 to 2048 for Winnie,
;                -1 for Non existent unit
;
; Note: The Unit Numbers will correspond to the numbers that the Host
;       would use (i.e., not necessarily the DRVSEL numbers). Thus,
;       Winnies will always precede Floppies and "null devices".
;*****
AUTOSZ:
.dsable AMA
.word <AUTOend-AUTOSZ> ;Byte count low TEST HEADER
.word 0 ;byte count high
.word 0 ;overlay low
.word 0 ;overlay high
.ascii /AUTOSZ/ ;6 character asciz name

```

AUTOSZ

```

025303      117      123      132
2043
2044 025306 000001      .even
2045 025310      000      .word      1      ;version number
2046 025311      177      .byte      0      ;flags
2047 025312 000240      .byte      177      ;timeout
2048
2049 025314      AUTO::      nop      ;start down line loaded test
2050 025314 000240
2051
2052      ; Executable Code Starts Here
2053
2054 025316 106427 000340      mtps      #ps7      ; Set up our own interrupts handlers
2055 025322 005037 140004      clr      @#w$fp1      ; clear the leds
2056 025326 013746 100002      mov      @#i$udc, (sp)      ; Save the MSCP handlers - UDC
2057 025332 013746 100006      mov      @#i$clk, (sp)      ; ... Clock
2058 025336 013746 100016      mov      @#i$sec, (sp)      ; ... Sector
2059
2060      ; Taken from RQDX3.MAC m$'n'it code:
2061
2062 025342 112737 000000 140022      movb      #op.res,@#w$cmd      ; reset the smc9224 chip
2063 025350 112737 000111 140022      movb      #op.srp+11,@#w$cmd      ; enable interrupts
2064 025356 112737 000040 140020      movb      #40,@#w$dat      ;
2065 025364 005067 001064      clr      s$$bug      ; assume the bug 's not present
2066 025370 032737 020000 140006      b't      #20000,@#r$fps      ; 's the ECO wire there?
2067 025376 001415      beq      sizset      ; definitely not
2068 025400 112737 000001 140022      movb      #op.dd,@#w$cmd      ; deselect all drives
2069 025406 012700 001000      mov      #1000,r0      ; wait for a bit
2070 025412      sizwt:      ; ...
2071 025412 005300      dec      r0      ; ...
2072 025414 001376      bne      sizwt      ; ...
2073
2074 025416 032737 020000 140006      b't      #20000,@#r$fps      ; is the ECO wire there?
2075 025424 001002      bne      s'zset      ; nope
2076 025426 005267 001022      inc      s$$bug      ; say it is
2077
2078 025432      sizset:      ; Set up handlers
2079 025432 010700      mov      pc,r0      ; ...
2080 025434 062700 000670      add      #<s$$udc-.>,r0      ; Use our own udc handler
2081 025440 010037 100002      mov      r0,@#i$udc      ; ...
2082 025444 010700      mov      pc,r0      ; ...
2083 025446 062700 000716      add      #<s$$rti .>,r0      ;
2084 025452 010037 100006      mov      r0,@#i$clk      ; Make clock interrupt rti
2085 025456 010037 100016      mov      r0,@#i$sec      ; Make sector interrupt rti
2086 025462 106427 000000      mtps      #ps0      ; Make it good
2087
2088      ; Go Size the Drives
2089
2090 025466 010146      mov      r1, (sp)      ; Save Registers
2091 025470 010246      mov      r2, -(sp)      ; Save Registers
2092 025472 010346      mov      r3, (sp)      ;
2093 025474 010702      mov      pc,r2      ; Point to Unit Descriptor Table
2094 025476 062702 000766      add      #<msgdat+2> .,r2      ; ...
2095 025502 010200      mov      r2,r0      ;
2096 025504 012703 000004      mov      #4.,r3      ; Initialize all Unit Descriptors
2097 025510      siznon:      ; ...
2098 025510 012720 177777      mov      #-1.,(r0)+      ; ... to "Non-Existant Unit"

```

J4

.MAIN. MACRO V05.03 Tuesday 10-Jun 86 13:21 Page 22 2

SEQ 0048

AUTOSZ

```

2099 025514 077303          sob      r3,siznon          ; ...
2100                                     ;
2101 025516 012703 000002      mov      #2.,r3          ; Set Drive Count to logical unit 0
2102                                     ;
2103 025522                sizlop::          ; ** Loop Until We Get All of Them **
2104                                     ; **Check if it is a Winnie**
2105 025522 012737 000010 140002      mov      #bit3,@#rw$pll      ; Set up Pl1ctl Csr
2106 025530 012737 000104 140022      mov      #op.srp+4,@#w$cmd      ; Set up UDC registers
2107 025536 005037 140020          clr      @#w$dat          ; ... Head 0
2108 025542 005037 140020          clr      @#w$dat          ; ... Cylinder 0
2109 025546 012737 000110 140022      mov      #op.srp+8.,@#w$cmd      ; ...
2110 025554 012737 000300 140020      mov      #rd.mode,@#w$dat      ; ... Set mode for winnie
2111 025562 010300          mov      r3,r0          ; Select the Drive
2112 025564 062700 000044          add      #op.srd,r0          ; ... op.sd.rd=44
2113 025570 004767 000572          jsr      pc,doudc          ; Do UDC command
2114 025574 005700          tst      r0          ; Okay?
2115 025576 001402          beq      sizfps          ; Nope, something 's screwed up
2116 025600 000167 000416          jmp      sizend
2117 025604                sizfps:          ;
2118 025604 032737 140000 140006      bit      #bit14+bit15,@#r$fps      ; Winnie?
2119 025612 001121          bne      s'zwin          ; Yes, go set cylinder count
2120                                     ;
2121 025614                sizflp:          ; ** Check if it is a Floppy **
2122 025614 012737 000011 140002      mov      #bit0+bit3,@#rw$pll      ; Set Pl1ctl CSR
2123 025622 112737 000107 140022      movb     #op.srp+7,@#w$cmd      ; Set up UDC registers
2124 025630 112737 000367 140020      movb     #retry,@#w$dat          ; ... retry = 367
2125 025636 010300          mov      r3,r0          ; Select the Drive
2126 025640 062700 000054          add      #op.srx,r0          ; ... op.sd.rx=54
2127 025644 004767 000516          jsr      pc,doudc          ; Do UDC command
2128 025650 005700          tst      r0          ; Okay?
2129 025652 001123          bne      sizend          ; Nope, something is screwed up
2130 025654 005004          clr      r4          ; Step counter
2131                                     ;
2132 025656                steprx:          ; ** Step In & Out Until Track 0 Found **
2133 025656 020427 000240          cmp      r4,#160.          ; How many times have we step?
2134 025662 002034          bge      sizrx          ; Enough?
2135 025664 112737 000111 140022      movb     #op.srp+9.,@#w$cmd      ; Set up UDC registers
2136 025672 132737 000020 140010      bitb     #bit4,@#r$dat          ; At track 0?
2137 025700 001025          bne      sizrx          ; Yes, then go check Floppy type
2138 025702 020427 000120          cmp      r4,#80.          ; Is step counter >= 80 ?
2139 025706 002412          blt      stepout          ;
2140 025710 020427 000202          cmp      r4,#130.          ; Is step counter <= 130 ?
2141 025714 003007          bgt      stepout          ;
2142 025716 012700 000005          mov      #op.sil,r0          ; Step in one track
2143 025722 004767 000440          jsr      pc,doudc          ; Do UDC command
2144 025726 005700          tst      r0          ; Okay?
2145 025730 001134          bne      sizend          ; Nope, something is screwed up
2146 025732 000406          br       stepmore          ;
2147 025734                stepout:          ;
2148 025734 012700 000007          mov      #op.sol,r0          ; Step out one track
2149 025740 004767 000422          jsr      pc,doudc          ; Do UDC command
2150 025744 005700          tst      r0          ; Okay?
2151 025746 001125          bne      sizend          ; Nope, something is screwed up
2152 025750                stepmore:          ;
2153 025750 005204          inc      r4          ; Increment step counter
2154 025752 000741          br       steprx          ; ** Bottom of find track 0 loop **
2155

```

AUTOSZ

```

2156 025754      sizrx:      movb    #op.srp+9.,@#w$cmd      ; ** Check Floppy type RX50/RX33 **
2157 025754 112737 000111 140022      bitb    #bit4,@#r$dat      ; Set up UDC registers
2158 025762 132737 000020 140010      beq     sizdrv      ; At track 0?
2159 025770 001506      movb    #op.srp+4,@#w$cmd      ; Set up UDC registers
2160 025772 112737 000104 140022      movb    #1,@#w$dat      ; .. Head =1
2161 026000 112737 000001 140020      mov     r3,r0      ; Select the Drive
2162 026006 010300      add     #op.srx,r0      ; ... op.sd.rx=54
2163 026010 062700 000054      jsr     pc,doudc      ; Do UDC command
2164 026014 004767 000346      tst     r0      ; Okay?
2165 026020 005700      bne     sizend      ; Nope, something is screwed up
2166 026022 001077      movb    #op.srp+9.,@#w$cmd      ; Set up UDC registers
2167 026024 112737 000111 140022      bitb    #bit4,@#r$dat      ; At track 0?
2168 026032 132737 000020 140010      bne     sizrx3      ; No, it's an RX50
2169 026040 001003      mov     #2,(r2)      ; Mark it as an RX50
2170 026042 012712 000002      br      sizrd      ;
2171 026046 000455      ;
2172 026050      sizrx3:      mov     #3,(r2)      ; Yes, mark it as an RX33
2173 026050 012712 000003      br      sizrd      ; Go do next drive
2174 026054 000452      ;
2175 026056      sizwin:      clr     (r2)      ; It's a Winnie Set Count to 0
2176 026056 005012      ;
2177      ;
2178 026060 012700 000007      mov     #op.s01,r0      ; Step out one track
2179 026064 004767 000276      jsr     pc,doudc      ; Do UDC command
2180 026070 005700      tst     r0      ; Okay?
2181 026072 001053      bne     sizend      ; Nope, something is screwed up
2182      ;
2183 026074 012700 000003      mov     #ersek0,r0      ; Assume that seek to 0 failed
2184 026100 112737 000111 140022      movb    #op.srp+9.,@#w$cmd      ; At Cylinder 0?
2185 026106 132737 000020 140010      bitb    #bit4,@#r$dat      ; ...
2186 026114 001442      beq     sizend      ; Nope, something's wrong
2187      ;
2188 026116      sizin:      inc     (r2)      ; ** Step In Unt'l Track 0 Found **
2189 026116 005212      mov     #op.si1,r0      ; Up Cylinder Count
2190 026120 012700 000005      jsr     pc,doudc      ; Step In One Cylinder
2191 026124 004767 000236      tst     r0      ; Do UDC Command
2192 026130 005700      bne     sizend      ; Okay?
2193 026132 001033      ; Nope, something is screwed up
2194      ;
2195 026134 112737 000111 140022      movb    #op.srp+9.,@#w$cmd      ; At Cylinder 0?
2196 026142 132737 000020 140010      bitb    #bit4,@#r$dat      ; If so, skip to bump up
2197 026150 001003      bne     parkit      ; ... descriptors
2198      ;
2199 026152 021227 004000      cmp     (r2),#2048.      ; SMC Cylinder Limit Reached?
2200 026156 002757      blt     sizin      ; ** Bottom of Step In Loop **
2201      ;
2202 026160      parkit:      mov     (r2),r1      ; step in, to inner most cylinder
2203 026160 011201      dec     r1      ; get total number of cylinders
2204 026162 005301      mov     #op.si1,r0      ; we want one less then recalibrate cylinder
2205 026164 012700 000005      jsr     pc,doudc      ; Step In One Cylinder
2206 026170 004767 000172      tst     r0      ; Do UDC Command
2207 026174 005700      bne     sizend      ; Okay?
2208 026176 001011      sob     r1,1$      ; Nope, something is screwed up
2209 026200 077107      ;
2210 026202      sizrd:      add     #untdsz,r2      ; ** This was a Winnie **
2211      ;
2212 026202 062702 000002      ; Bump Pointer to Next Unit Descriptor

```

AUTOSZ

```

2213
2214 026206          sizdrv:          ; ** Check Next Drive **
2215 026206 005203      inc            r3          ; Up Drive Count
2216 026210 020327 000005  cmp          r3,#5.      ; All 4 Drives Checked?
2217 026214 003002      bgt            sizend       ; ...
2218 026216 000167 177300  jmp          sizlop      ; ** Bottom of Loop **
2219
2220 026222          sizend:          ; ** Send Status and Table **
2221 026222 010067 000234  mov          r0,msgdat    ; Save status
2222 026226 012700 000001  mov          #op.dd,r0    ; Deselect Drive
2223 026232 004767 000130  jsr          pc,doudc     ; ...
2224 026236 012603      mov          (sp)+,r3        ; Pop
2225 026240 012602      mov          (sp)+,r2        ; ...
2226 026242 012601      mov          (sp)+,r1        ; ...
2227 026244 106427 000340  mtps         #ps7         ; Put the MSCP Handlers Back
2228 026250 012637 100016  mov          (sp)+,@#i$sec  ; ...
2229 026254 012637 100006  mov          (sp)+,@#i$clk  ; ...
2230 026260 012637 100002  mov          (sp)+,@#i$udc  ; ...
2231 026264 106427 000000  mtps         #ps0         ; ...
2232
2233 026270          sizexi::          ; ** Okay, talk to the Host **
2234
2235          ;PutData,msg,msglen  Send Response to Host
2236
2237 026270 010700      mov          pc,r0            ;figure the relative address
2238 026272 062700 000166  add          #msg-.,r0      ;... of the buffer
2239 026276 012746 000014  mov          #msglen,(sp)   ;load lenght in bytes of the buffer
2240 026302 010046      mov          r0,-(sp)         ;load relative address of the buffer
2241 026304 013746 000146  mov          @#146,(sp)     ;load location of routine in microcode
2242 026310 004736      jsr          pc,@(sp)+        ;call Put Data routine in Ucode
2243 026312 022626      cmp          (sp)+,(sp)+      ;fix stack
2244
2245          ; Term'nate Supplied Program
2246
2247 026314 013700 000142  mov          @#142,r0        ;load location of routine in microcode
2248 026320 004710      jsr          pc,(r0)          ;call Terminate routine in Ucode
2249 026322 000207      rts          pc              ; ...

```

AUTOSZ

```

2251      ;+
2252      ; UDC Interrupt Handler
2253      ;
2254      ; Taken from RQDX3.MAC m$udc code:
2255      ; -
2256
2257      s$$udc::      ; UDC Handler
2258      026324      005767      000124      tst      s$$bug      ; 's the ECO wire there?
2259      026330      001404      beq      s$$udi      ; nope
2260      026332      032737      020000      140006      bit      #20000, @#r$fps      ; 's the 9224 interrupt line set?
2261      026340      001011      bne      s$$rti      ; if not, must be a bogus interrupt
2262
2263      s$$udi::      ; ...
2264      026342      113746      140012      movb     @#r$cmd, (sp)      ; get interrupt status
2265      026346      142716      000035      bicb     #35, (sp)      ; clear bits of no interest
2266      026352      122726      000240      cmpb     #240, (sp)+      ; val'd status?
2267      026356      001002      bne      s$$rti      ; no, it's a bogus interrupt
2268      026360      005267      000072      inc      s$$flag      ; set the flag
2269
2270      ;+
2271      ; Return from Interrupt
2272      ;
2273
2274      s$$rti::      ;;; ...
2275      026364      000002      rti      ;;; just quit
2276
2277      ;+
2278      ; DOUDC - Do a UDC Command
2279      ;
2280      ; This routine sends a commands and waits an interrupt or
2281      ; until timer expires.
2282      ;
2283      ; Input:      r0      - command
2284      ; Output:     r0      - 0 for success, non zero for failure
2285      ; -
2286
2287      msec = 30.*132.      ; Max Step Rate + some *
2288      ; loop for 7.5 MHz T11 clock
2289
2290      doudc::      ; ** Do a UDC command **
2291      026366      010146      mov     r1, -(sp)      ; save r1
2292      026370      005067      000062      clr      s$$flag      ; Clear udc flag (interrupt pending)
2293      026374      010037      1+0022      mov     r0, @#w$cmd      ; Send the command
2294      026400      012700      004000      mov     #2048., r0      ; Set the rom timer (max cylinders)
2295
2296      mswait:      ; ** Wait **
2297      026404      012701      007570      mov     #msec, r1      ; set one millisecond counter
2298      026410      ms'n:      ; ** Top of Inner Loop **
2299      026410      005767      000042      tst      s$$flag      ; 3.60 udc interrupted
2300      026414      001005      bne      msend      ; 1.60 out if udc interrupted
2301      026416      077104      sob      r1, ms'n      ; 2.40 Total: 7.60 @7.5MHz=>
2302      ; 8.5457 @6.67MHz
2303      026420      077007      sob      r0, mswait      ; ** Bottom of Outer Loop **
2304      026422      012700      000002      mov     #eruint, r0      ; Never Interrupted
2305      026426      000410      br       douret      ; ...
2306
2307      msend:      ; ** Interrupt Happened **

```

N4

.MAIN. MACRO V05.03 Tuesday 10 Jun 86 13:21 Page 23 1

SEQ 0052

AUTOSZ

2308	026430	012700	000001	mov	#erudon,r0	; Assume Never Done
2309	026434	013701	140012	mov	@#r\$cmd,r1	; Get the return status
2310	026440	032701	000040	bit	#bit5,r1	; All done yet?
2311	026444	001401		beq	douret	; If so, pop out of this
2312						
2313	026446	005000		clr	r0	; Assume everything's ok
2314						
2315	026450			douret:		; ** Return **
2316	026450	012601		mov	(sp)+,r1	
2317	026452	000207		rts	pc	; Back to caller

SIZER supplied Program Data

2319				.sbt1 SIZER Supplied Program Data	
2320					
2321				; .psect cldata	
2322					
2323				; Special Stuff	
2324					
2325	026454			s\$\$bug: .blkw 1	; ECO Wire
2326	026456			s\$\$flag: .blkw 1	; UDC flag
2327					
2328				; Packet Area	
2329					
2330	026460	012	140	msg:: .byte 10...b.spl	; Final Message
2331	026462			msgdat: .blkw 5.	; Status and Unit Descriptor Table
2332		000014		msglen = .msg	; Message Length (Byte Count)
2333		000002		untdsz = 2.	; Unit Descriptor Length
2334					
2335				.enable AMA	
2336	026474			AUTOend:	

SIZER Supplied Program Data

```

2338
2339
2340
2341
2342
2343
2344
2345
2346
2347 026474
2348 026474 123727 026461 000140
2349 026502 001401
2350 026504 000207
2351
2352 026506 123727 026460 000012
2353 026514 001401
2354 026516 000207
2355 026520
2356 026520 005737 026462
2357 026524 001457
2358
2359
2360
2361 026526
2362 026552 023727 026462 000001
2363 026560 001010
2364 026562
2365 026602 023727 026462 000002
2366 026610 001010
2367 026612
2368 026632 023727 026462 000003
2369 026640 001010
2370 026642
2371 026662
2372 026662 000207
2373
2374
2375 026664
2376 026664
2377 026704 012701 026464
2378 026710 005002
2379 026712 022711 177777
2380 026716 001013
2381 026720
2382 026742 000137 027522
2383 026746 022711 000002
2384 026752 001013
2385 026754
2386 026776 000137 027522
2387 027002 022711 000003
2388 027006 001013
2389 027010
2390 027032 000137 027522
2391 027036
2392 027036
2393
2394 027062

;*****
;               AUTodisplay
;   This routine will display the results of the autosizers
;   findings. It will say weather the autosizer errored or not and
;   what drives it found.
;*****
AUTodis:
    cmpb    msg+1, #.b.spl      ;check 'f Special Message
    beq     1$                  ;'f not then no 'nfo to print
    rts     pc                  ;so just return

1$:    cmpb    msg, #10.        ;
    beq     2$                  ;check message number
    rts     pc                  ;return if msg number doesn t match

2$:    tst     msg+2            ;test complet'on status of Autosizer
    beq     24$                 ;if zero no error report the findings
                                   ;if not zero then there 's an error

; Autos'zer Failure Code
    printb  #ASMSG3, msg+2      ; Print Autosizer Failure Code
    cmp     msg+2, #1           ; Is it a UDC never done error ?
    bne     11$                 ; No, check for next code
    printb  #ASMSG4             ; Yes, Tell error type
11$:    cmp     msg+2, #2       ; Is 't a UDC never interrupted error ?
    bne     12$                 ; No, check for next code
    printb  #ASMSG5             ; Yes, Tell error type
12$:    cmp     msg+2, #3       ; Is it a seek error ?
    bne     13$                 ; No, go reinitialize ctrl
    printb  #ASMSG6             ; Yes, Tell error type
13$:    rts     pc              ;return

; Autosizer Findings
24$:    printb  #ASMSG1          ; print Autosizer findings
    mov     #msg+4, r1          ; first cylinder entry
    clr     r2                  ; Start with unit number zero
26$:    cmp     #-1., (r1)      ; Is unit Non-existent ?
    bne     61$                 ; No, check for RX50
    printb  #ASMSG7, R2         ; Yes, tell 't is non-existent
    jmp     20$                 ;
61$:    cmp     #2., (r1)      ; Is unit an RX50 ?
    bne     62$                 ; No, check for RX33
    printb  #ASMSG8, R2         ; Yes, tell it is an RX50
    jmp     20$                 ;
62$:    cmp     #3., (r1)      ; Is unit an RX33 ?
    bne     63$                 ; No, then it is a Winchester
    printb  #ASMSG9, R2         ; Yes, tell 't is RX33
    jmp     20$                 ;
63$:    printb  #ASMSG2, r2, (r1) ; It is a WINCHESTER
                                   ; Tell it is a Winchester with so many cylinder

71$:

```

SIZER Supplied Program Data

```

2395 027062 023711 003102      cmp      UIT0+UITsiz-2,(r1)      ;if cylinder # equals UIT table # this is the correc
t  UIT table
2396 027066 001403      beq      710$
2397 027070 023711 003100      cmp      UIT0+UITsiz 4,(r1)      ;if cylinder # equals UIT table # this is the correc
t  JIT table
2398 027074 001012      bne      72$
2399 027076      710$:      printb  #DrvTx0      ;1      rd51
2400 027116 000137 027522      jmp
2401
2402 027122 023711 003206      72$:      cmp      UIT1+UITsiz-2,(r1)      ;if cylinder # equals UIT table # this is the correc
t  UIT table
2403 027126 001403      beq      720$
2404 027130 023711 003204      cmp      UIT1+UITsiz-4,(r1)      ;if cylinder # equals UIT table # this is the correc
t  UIT table
2405 027134 001011      bne      73$
2406 027136      720$:      printb  #DrvTx1      ;1      rd52
2407 027156 000561      br
2408
2409 027160 023711 003312      73$:      cmp      UIT2+UITsiz-2,(r1)      ;if cylinder # equals UIT table # th's is the correc
t  UIT table
2410 027164 001403      beq      730$
2411 027166 023711 003310      cmp      UIT2+UITsiz 4,(r1)      ;if cylinder # equals UIT table # this is the correc
t  UIT table
2412 027172 001011      bne      74$
2413 027174      730$:      printb  #DrvTx2      ;1      rd52
2414 027214 000542      br
2415
2416 027216 023711 003416      74$:      cmp      UIT3+UITsiz 2,(r1)      ;if cylinder # equals UIT table # this is the correc
t  UIT table
2417 027222 001403      beq      740$
2418 027224 023711 003414      cmp      UIT3+UITsiz 4,(r1)      ;if cylinder # equals UIT table # this is the correc
t  UIT table
2419 027230 001011      bne      75$
2420 027232      740$:      printb  #DrvTx3      ;1      rd53
2421 027252 000523      br
2422
2423 027254 023711 003522      75$:      cmp      UIT4+UITsiz-2,(r1)      ;if cylinder # equals UIT table # this is the correc
t  UIT table
2424 027260 001403      beq      750$
2425 027262 023711 003520      cmp      UIT4+UITsiz 4,(r1)      ;if cylinder # equals UIT table # th's is the correc
t  UIT table
2426 027266 001011      bne      76$
2427 027270      750$:      printb  #DrvTx4      ;1      rd54
2428 027310 000504      br
2429
2430 027312 023711 003626      76$:      cmp      UIT5+UITsiz-2,(r1)      ;if cylinder # equals UIT table # this is the correc
t  UIT table
2431 027316 001403      beq      760$
2432 027320 023711 003624      cmp      UIT5+UITsiz-4,(r1)      ;if cylinder # equals UIT table # this is the correc
t  UIT table
2433 027324 001011      bne      77$
2434 027326      760$:      printb  #DrvTx5      ;1      rd31
2435 027346 000465      br
2436
2437 027350 023711 003732      77$:      cmp      UIT6+UITsiz 2,(r1)      ;if cylinder # equals UIT table # th's is the correc
t  UIT table
2438 027354 001403      beq      770$
2439 027356 023711 003730      cmp      UIT6+UITsiz-4,(r1)      ;if cylinder # equals UIT table # this is the correc
t  UIT table
2440 027362 001011      bne      78$
2441 027364      770$:      printb  #DrvTx6      ;1      rd
2442 027404 000446      br
2443
2444 027406 023711 004036      78$:      cmp      UIT7+UITsiz-2,(r1)      ;if cylinder # equals UIT table # this is the correc
t  UIT table
2445 027412 001403      beq      780$

```

```

2447 027420 001011
2448 027422 780$: bne 79$
2449 027442 000427 printb #DrvTx7 ;1 rd
2450 br 20$
2451 027444 023711 004142 79$: cmp UITdf+UITsiz-2,(r1) ;if cylinder # equals UIT table # this is the correc
t UIT table
MAIN. MACRO V05.03 Tuesday 10 Jun-86 13:21 Page 25-2
SEQ 0056

SIZER Supplied Program Data

2452 027450 001403 beq 790$
2453 027452 023711 004140 cmp UITdf+UITsiz-4,(r1) ;if cylinder # equals UIT table # this is the correc
t UIT table
2454 027456 001011 bne 80$
2455 027460 790$: printb #DrvTxc ;1 custom rd
2456 027500 000410 br 20$
2457
2458 027502 80$: printb #ASMSGr ;"Unrecognized Drive"
2459
2460 027522 005721 20$: tst (r1)+ ; Point to next unit descriptor
2461 027524 005202 inc r2 ; Set for next unit
2462 027526 020227 000004 cmp r2,#MaxDrv ; Last unit?
2463 027532 001402 beq 27$ ; Yes, exit routine
2464 027534 000137 026712 jmp 26$ ; No, do next unit
2465 027540 000207 27$: rts pc ;
2466
2467 ;*****
2468 ;
2469 ; This routine builds the UIT table or get the UIT table
2470 ; depending who the questions are answered to the manual questions.
2471 ; If the unit is a listed or regconizable drive we will use a prebuilt
2472 ; UIT table. If not we will have to ask all the questions to build
2473 ; a table.
2474 ;
2475 ;*****
2476 027542 BLDUIT:
2477 027542 032737 100000 002336 bit #bit15,untflgs
2478 027550 001402 beq manbld
2479 027552 000137 030060 jmp autobld
2480
2481 027556 manbld: printf #DrvTxa ;print out UIT tables and the'r related drives
2482 027576 printf #DrvTxb ;UIN Drive
2483 027616 printf #DrvTx0 ;0 rd51
2484 027636 printf #DrvTx1 ;1 rd52
2485 027656 printf #DrvTx2 ;2 etc
2486 027676 printf #DrvTx3 ;3 etc
2487 027716 printf #DrvTx4 ;4
2488 027736 printf #DrvTx5
2489 027756 printf #DrvTx6
2490 027776 printf #DrvTx7
2491 030016 printf #DrvTxc
2492
2493 030036 GMANID unt.nbr,UIN,0,17,0,10,no ;GET Unit identifier number (0 7)
2494 ;PLACE IN bits 0-3.
2495 ;no defaults person must know what Unit Identificati
on number.
2496 030056 000515 br uitloc ;get correct table address into UITadrs
2497
2498 030060 autobld:
2499 030060 013700 002330 mov unit,r0 ;get unit number
2500 030064 006300 asl r0 ;get the byte offset of tbl
2501 030066 012737 000000 002344 1$: mov #0,uin ;pick UIT number 0
2502 030074 023760 003102 026464 cmp UIT0+UITsiz 2,msg+4(r0) ;if cylinder # equals UIT table # this is the correc
t UIT table
2503 030102 001503 beq 2$
2504 030104 012737 000001 002344 mov #1,uin ;pick UIT number 1
2505 030112 023760 003206 026464 cmp UIT1+UITsiz-2,msg+4(r0) ;if cylinder # equals UIT table # this is the correc
t UIT table
2506 030120 001474 beq 2$
2507 030122 012737 000002 002344 mov #2,uin ;pick UIT number 2
2508 030130 023760 003312 026464 cmp UIT2+UITsiz-2,msg+4(r0) ;if cylinder # equals UIT table # this is the correc
t UIT table

```

SIZER Supplied Program Data

```

2509 030136 001465      beq      2$
2510 030140 012737 000003 002344      mov      #3,uin      ;pick UIT number 3
2511 030146 023760 003416 026464      cmp      UIT3+UITsiz 2,msg+4(r0) ;if cylinder # equals UIT table # this is the correc
t UIT table
2512 030154 001456      beq      2$
2513 030156 012737 000004 002344      mov      #4,uin      ;pick UIT number 4
2514 030164 023760 003522 026464      cmp      UIT4+UITsiz-2,msg+4(r0) ;if cylinder # equals UIT table # this is the correc
t UIT table
2515 030172 001447      beq      2$
2516 030174 023760 003520 026464      cmp      UIT4+UITsiz-4,msg+4(r0) ;if cylinder # equals UIT table # this is the correc
t UIT table
2517 030202 001443      beq      2$      ;automatic recal feature of this drive
2518 030204 012737 000005 002344      mov      #5,uin      ;pick UIT number 5
2519 030212 023760 003626 026464      cmp      UIT5+UITsiz 2,msg+4(r0) ;if cylinder # equals UIT table # this is the correc
t UIT table
2520 030220 001434      beq      2$
2521 030222 023760 003624 026464      cmp      UIT5+UITsiz 4,msg+4(r0) ;if cylinder # equals UIT table # this is the correc
t UIT table
2522 030230 001430      beq      2$      ;automatic recal feature of this drive
2523 030232 012737 000006 002344      mov      #6,uin      ;pick UIT number 6
2524 030240 023760 003732 026464      cmp      UIT6+UITsiz 2,msg+4(r0) ;if cylinder # equals UIT table # this is the correc
t UIT table
2525 030246 001421      beq      2$
2526 030250 012737 000007 002344      mov      #7,uin      ;pick UIT number 7
2527 030256 023760 004036 026464      cmp      UIT7+UITsiz 2,msg+4(r0) ;if cylinder # equals UIT table # this is the correc
t UIT table
2528 030264 001412      beq      2$
2529 030266      printb      #efunrg      ;"No UIT table suitable for this drive"
2530 030306 000137 037420      jmp      dropunt      ;drop unit and end pass
2531 030312      2$:
2532 030312      uitloc:
2533 030312 012703 003000      mov      #UIT0,r3      ;r3 contains base address of UIT tables
2534 030316 013702 002344      mov      UIN,r2      ;get the correct UIT table address into UITadr regis
ter
2535 030322 001403      beq      11$      ;if UIN=0 then set table to UIT0
2536 030324 062703 000104 10$:      add      #UITsiz,r3      ;else multiply UIT size by the UIN number and add to
base address
2537 030330 077203      sob      r2,10$
2538 030332 010337 002320 11$:      mov      r3,UITadr      ;store the proper address of the UIT table
2539 030336 000137 030344      jmp      cont      ;all done
2540
2541 030342      tblbld:      ;We must build a UNIT INFORMATION TABLE
2542 030342 000240      nop      ;try IRQCBI for custom built tables available thru S
DC.
2543 030344 000207      cont:      rts      pc      ;go back
2544      ;*****
2545      ;
2546      ;      Octal number to ASCII Decimal number
2547      ;      r1 = address of ascii decimal data
2548      ;      r0 = octal data word
2549      ;*****
2550 030346      OCTASC:
2551 030346 010246      mov      r2,-(sp)
2552 030350 010346      mov      r3,(sp)
2553 030352 005002      clr      r2      ;clear the decimal table pointer
2554 030354 005003 1$:      clr      r3      ;clear decimal digit
2555 030356 005203 2$:      inc      r3      ;increment decimal digit
2556 030360 166200 030420      sub      dectbl(r2),r0      ;subtract a power of ten from accumulator
2557 030364 002374      bge      2$      ;if not negative subtract another
2558 030366 066200 030420      add      dectbl(r2),r0      ;adjust accumulator so positive
2559 030372 005303      dec      r3      ;adjust decimal digit
2560 030374 062703 000060      add      #60,r3      ;convert decimal to ascii
2561 030400 110321      movb     r3,(r1)+      ;mov ascii digit text into buffer
2562 030402 005722      tst      (r2)+      ;increment table pointer
2563 030404 005762 030420      tst      dectbl(r2)      ;check if thats all
2564 030410 001361      bne      1$

```

SIZER Supplied Program Data

```

2566 030414 012602      mov      (sp)+,r2
2567 030416 000207      rts      pc
2568 030420      dectbl:  .word    10000.
2569 030420 023420      .word    1000.
2570 030422 001750      .word    100.
2571 030424 000144      .word    10.
2572 030426 000012      .word    1.
2573 030430 000001      .word    0
2574 030432 000000      ;*****
2575      ;
2576      ;       ASCII DECIMAL numbers to Octal numbers
2577      ;       r1 - address of ascii decimal data
2578      ;       r0 = address to store octal data low word, high word
2579      ;*****
2580      ASCDEC:
2581 030434      mov      r5, (sp)
2582 030434 010546      mov      r4, (sp)
2583 030436 010446      mov      r3, (sp)
2584 030440 010346      mov      r2, -(sp)
2585 030442 010246      clr      r4
2586 030444 005004      clr      r3
2587 030446 005003      clr      r2
2588 030450 005002      3$:      movb   (r1)+,r4
2589 030452 112104      beq      1$
2590 030454 001423      ;if digit equals null than all done
2591      ;             cmp      r4,#60      ;check for a real number value
2592      ;             blt      asklbn      ;wasn't a real number
2593      ;             cmp      r4,#71
2594      ;             bgt      asklbn      ;wasn't a real number
2595      ;
2596 030456 162704 000060      sub      #60,r4
2597 030462 010346      mov      r3, (sp)
2598 030464 010246      mov      r2, (sp)      ;save accum
2599      ;
2600 030466 012705 000003      mov      #3,r5      ;accum * 8
2601 030472 006302      4$:      asl      r2
2602 030474 006103      rol      r3
2603 030476 077503      sob      r5,4$
2604      ;
2605 030500 006316      asl      (sp)      ;accum*2
2606 030502 006166 000002      rol      2(sp)
2607      ;
2608 030506 000241      clc
2609 030510 062602      add      (sp)+,r2
2610 030512 005503      adc      r3
2611 030514 062603      add      (sp)+,r3
2612      ;
2613 030516 060402      add      r4,r2      ;add present digit to accum*10
2614 030520 005503      adc      r3
2615 030522 000753      br      3$
2616      ;
2617 030524 010220      1$:      mov      r2,(r0)+      ;load lo number
2618 030526 010310      mov      r3,(r0)      ;load hi number
2619      ;
2620 030530 012602      mov      (sp)+,r2
2621 030532 012603      mov      (sp)+,r3
2622 030534 012604      mov      (sp)+,r4
;restore stack to its original

```

SIZER Supplied Program Data

```

2623 030536 012605      mov      (sp)+,r5
2624 030540 000207      rts      pc
2625
2626 ;*****
2627 ;
2628 ; This routine types out the ASCII information passed
2629 ; by the disk controller. This ASCII information is
2630 ; contained in the buffer called DATARE and is offset
2631 ; by 1 word. To fake the DRS macro routine a "A" is
2632 ; placed in front of the text.
2633 ;*****
2634
2635 030542      typDUPbuf:
2636 030542 012701 002552      mov      #datare,r1      ;get data area address of ascii info
2637 030546 063701 002366      add      rsppak+14,r1      ;add the number of byte transfered
2638 030552 105021      1$:      clrb      (r1)+      ;put null characters into data buffer after end of ASCII inf
;
2639 030554 020127 002676      cmp      r1,#prgnam      ;
2640 030560 001374      bne      1$      ;we do this to fake out the DRS macro
2641
2642 030562 112737 000045 002552      movb     #45,datare      ;put the "A" delimiter for the DRS macro
2643 030570 112737 000101 002553      movb     #101,datare+1      ;put the "A" for ascii info for the DRS macro
2644 030576      printx     #PB13      ;New Line <cr><lf>
2645 030616      printx     #datare      ;print the message returned from the controller
2646
2647 030636      clrDUPbuf:
2648 030636 012701 002552      mov      #datare,r1      ;clear out entire data area
2649 030642 105021      2$:      clrb      (r1)+      ;
2650 030644 020127 002676      cmp      r1,#prgnam      ;
2651 030650 001374      bne      2$      ;
2652 030652 000207      rts      pc
2653 ;*****
2654 ;
2655 ; THIS ROUTINE IS TO CHECK ON THE RESPONSE PACKET
2656 ; GOODNESS. THE COMMAND REFERENCE NUMBER, THE END CODE
2657 ; AND THE STATUS ARE TESTED.
2658 ;*****
2659
2660 030654      RSPCHK:
2661
2662 030654 013701 002452      mov      cmdpak,r1
2663 030660 013700 002352      mov      rsppak,r0
2664 030664 020001      cmp      r0,r1      ;compare CRN numbers
2665 030666 001014      bne      1$
2666 030670 013701 002462      mov      cmdpak+10,r1
2667 030674 062701 000200      add      #200,r1
2668 030700 013700 002362      mov      rsppak+10,r0
2669 030704 020001      cmp      r0,r1      ;compare Opcodes
2670 030706 001004      bne      1$
2671 030710 013701 002364      mov      rsppak+12,r1      ;check the status
2672 030714 001001      bne      1$
2673 030716 000207      rts      pc      ;if all checks then return
2674
2675 ;if all doesn't check then a bad packet
2676 030720      1$:      ERRDF      10,df11      ;Bad response packet
2677 030730      PRNTpkt:
2678 030730      Printb     #PB11crn,cmdpak,rsppak      ;Expected CRN XXXX ,Received CRN YYYY
2679 030760 013701 002362      mov      rsppak+10,r1      ;check response opcode reply

```

SIZER Supplied Program Data

```

2680 030764 032701 000200      bit    #200,r1      ;see if a end command response was send
2681 030770 001010      bne     2$              ;No end bit in response packet endcode
2682 030772      printx #PB1lend      ;check if Get Dust Status command
2683 031012 022701 000201      2$:  cmp    #201,r1
2684 031016 001010      bne     3$              ;check if Execute Supplied Program
2685 031020      printx #PB1IGDS
2686 031040 022701 000202      3$:  cmp    #202,r1
2687 031044 001010      bne     4$              ;check if Execute Local Program
2688 031046      printx #PB1IESP
2689 031066 022701 000203      4$:  cmp    #203,r1
2690 031072 001010      bne     5$              ;check if Send Data
2691 031074      printx #PB1ELP
2692 031114 022701 000204      5$:  cmp    #204,r1
2693 031120 001010      bne     6$              ;check if Receive Data
2694 031122      printx #PB1SD
2695 031142 022701 000205      6$:  cmp    #205,r1
2696 031146 001022      bne     7$              ;check if Receive Data
2697 031150      printx #PB1RD
2698 031170      Printb  #PBSF0,r3,r5      ;"type xxx, message number xxxxx is unknow to this program"
2699 031214 022701 000206      7$:  cmp    #206,r1
2700 031220 001010      bne     8$              ;check if Abort Program
2701 031222      printx #PB1AP
2702 031242      8$:  Printb  #PB1lop,cmdpak+10,rsppak+10
2703      ;CMDpkt opcode XXXX,RSPpkt opcode YYYYY
2704      ;find out what kind of status we have
2705 031272 013701 002364      mov     rsppak+12,r1
2706 031276 022701 000000      cmp     #0.,r1
2707 031302 001010      bne     10$
2708 031304      printx #pb1ls0      ;status:  successful
2709 031324 022701 000001      10$:  cmp    #1.,r1
2710 031330 001010      bne     11$
2711 031332      printx #pb1ls1      ;status:  Invalid Command
2712 031352 022701 000002      11$:  cmp    #2.,r1
2713 031356 001010      bne     12$
2714 031360      printx #pb1ls2      ;status:  No Region Available
2715 031400 022701 000003      12$:  cmp    #3.,r1
2716 031404 001010      bne     13$
2717 031406      printx #pb1ls3      ;status:  No Region Suitable
2718 031426 022701 000004      13$:  cmp    #4.,r1
2719 031432 001010      bne     14$
2720 031434      printx #pb1ls4      ;status:  Program Not Known
2721 031454 022701 000005      14$:  cmp    #5.,r1
2722 031460 001010      bne     15$
2723 031462      printx #pb1ls5      ;status:  Load Failure
2724 031502 022701 000006      15$:  cmp    #6.,r1
2725 031506 001010      bne     16$
2726 031510      printx #pb1ls6      ;status:  Standalone
2727 031530 022701 000011      16$:  cmp    #9.,r1
2728 031534 001010      bne     19$
2729 031536      printx #pb1ls9      ;status:  Host Buffer Access error
2730 031556      19$:  Printb  #PB1sts,rsppak+12      ;Response packet status XXXX
2731 031556      jmp     dropunt      ;drop unit and go on
2732 031602 000137 037420
2733
2734      ;*****
2735      ;
2736

```

SIZER Supplied Program Data

```

2737
2738
2739 031606
2740 031606 032714 100000
2741 031612 001001
2742 031614 000207
2743 031616
2744 031626 011401
2745 031630 022701 001000
2746 031634 001010
2747 031636
2748 031656 022701 100001
2749 031662 001010
2750 031664
2751 031704 022701 100002
2752 031710 001010
2753 031712
2754 031732 022701 100003
2755 031736 001010
2756 031740
2757 031760 022701 100004
2758 031764 001010
2759 031766
2760 032006 022701 100005
2761 032012 001010
2762 032014
2763 032034 022701 100006
2764 032040 001010
2765 032042
2766 032062 022701 100007
2767 032066 001010
2768 032070
2769 032110 022701 100010
2770 032114 001010
2771 032116
2772 032136 022701 100011
2773 032142 001010
2774 032144
2775 032164 022701 100012
2776 032170 001010
2777 032172
2778 032212 022701 100013
2779 032216 001010
2780 032220
2781 032240 022701 100014
2782 032244 001010
2783 032246
2784 032266 022701 100015
2785 032272 001010
2786 032274
2787 032314 022701 100016
2788 032320 001010
2789 032322
2790 032342 022701 100017
2791 032346 001010
2792 032350
2793 032370 022701 100020

```

```

;
;*****
BIT15T:
    bit    #bit15,(r4)
    bne    100$
    rts    pc
100$:     ERRDF 9,df12          ;Fatal SA error
    mov    (r4),r1
    cmp    #1000,r1
    bne    1$
    printx #pb1201            ;
    cmp    #100001,r1
    bne    2$
    printx #pb1202            ;
    cmp    #100002,r1
    bne    3$
    printx #pb1203            ;
    cmp    #100003,r1
    bne    4$
    printx #pb1204            ;
    cmp    #100004,r1
    bne    5$
    printx #pb1205            ;
    cmp    #100005,r1
    bne    6$
    printx #pb1206            ;
    cmp    #100006,r1
    bne    7$
    printx #pb1207            ;
    cmp    #100007,r1
    bne    8$
    printx #pb1208            ;
    cmp    #100010,r1
    bne    9$
    printx #pb1209            ;
    cmp    #100011,r1
    bne    10$
    printx #pb1210            ;
    cmp    #100012,r1
    bne    11$
    printx #pb1211            ;
    cmp    #100013,r1
    bne    12$
    printx #pb1212            ;
    cmp    #100014,r1
    bne    13$
    printx #pb1213            ;
    cmp    #100015,r1
    bne    14$
    printx #pb1214            ;
    cmp    #100016,r1
    bne    15$
    printx #pb1215            ;
    cmp    #100017,r1
    bne    16$
    printx #pb1216            ;
    cmp    #100020,r1

```

K5

.MAIN. MACRO V05.03 Tuesday 10 Jun-86 13:21 Page 25 8

SEQ 006.

SIZER Supplied Program Data

```

2794 032374 001010      bne      17$
2795 032376             printx   #pb1217      ;
2796 032416 022701 100021 17$:    cmp      #100021,r1
2797 032422 001010      bne      18$
2798 032424             printx   #pb1218      ;
2799 032444 022701 100022 18$:    cmp      #100022,r1
2800 032450 001010      bne      19$
2801 032452             printx   #pb1219      ;
2802 032472 022701 100023 19$:    cmp      #100023,r1
2803 032476 001010      bne      20$
2804 032500             printx   #pb1220      ;
2805 032520 022701 100024 20$:    cmp      #100024,r1
2806 032524 001010      bne      21$
2807 032526             printx   #pb1221      ;
2808 032546 022701 100025 21$:    cmp      #100025,r1
2809 032552 001010      bne      22$
2810 032554             printx   #pb1222      ;
2811 032574 022701 100026 22$:    cmp      #100026,r1
2812 032600 001010      bne      23$
2813 032602             printx   #pb1223      ;
2814 032622             printb   #pb12,r1      ;SA value: xxxxx
2815 032622             jmp      dropunt      ;drop unit and go on
2816 032644 000137 037420
2817
2818 ;*****+*****
2819 ;      Unexpected Interrupt Server
2820 ;
2821 ;*****+*****
2822 032650      intsrv:
2823
2824             ERRSF      8,sf100 ;Fatal SA error
2825             docln      ;do clean up and ou+
2826 032662 000137 037420      jmp      dropunt      ;drop test unit and go pass
2827
2828

```

SIZER Supplied Program Data

```

2830 032666      BGNPROT
2831 032666      .WORD -1
2832 032670      .WORD 1
2833 032672      .WORD 1
2834 032674      ENDPROT
2835
2836 032674      BGNINIT
2837 032674      READEF      #EF.CONTINUE
2838 032702      BCOMPLETE  conton
2839 032704      READEF      #EF.NEW
2840 032712      BNCOMPLETE  next
2841 032714      SETUP:
2842 032714      012737 177777 002310      mov      #-1,LOGUNIT      ;Initialize logical unit nbr
2843 032722      NEXT:
2844 032722      005237 002310      .nc      LOGUNIT      ;Point to next logical unit
2845 032726      023737 002310 002012      cmp      LOGUNIT,L$UNIT      ;Have we passed maximum?
2846 032734      001002      bne      1$      ;No
2847 032736      000137 033114      jmp      ABORT      ;Yes, abort the pass
2848 032742      1$:
2849 032742      GPHARD LOGUNIT,PLOC      ;Get the P table
2850 032754      BNCOMPLETE NEXT      ;if not available get next unit
2851
2852 032756      013700 002314      mov      ploc,r0
2853 032762      010037 002316      mov      r0,ptbl      ;store the Ptable address for unit
2854 032766      012037 002324      mov      (r0)+,ipreg      ;store IPreg address into register
2855 032772      012037 002326      mov      (r0)+,vector      ;store vector
2856 032776      012037 002330      mov      (r0)+,unit      ;store logical drive number
2857 033002      012037 002334      mov      (r0)+,sernbr      ;store the serial number
2858 033006      012037 002336      mov      (r0)+,untflgs
2859
2860 033012      005037 002540      conton: clr      LSTCRN      ;basic initialization stuff
2861 033016      005037 002544      clr      LSTVCT
2862 033022      005037 002546      clr      LOPRGI
2863 033026      005037 002550      clr      HIPRGI
2864
2865 033032      013746 000004      1$:      mov      @#4,(sp)      ;test to see if controller is there
2866 033036      012737 033052 000004      mov      #$2,@#4
2867 033044      005077 147254      clr      @IPreg
2868 033050      000410      br      $3      ;get controller into know state
2869
2870 033052      $2:      ERRDF 7,DF4      ;NXM trap at controller IP address
2871 033062      dodu LOGUNIT      ;drop unit
2872 033070      000714      br      next      ;get new unit
2873
2874 033072      012637 000004      $3:      mov      (sp)+,@#4      ;move value back into location 4
2875
2876 033076      012700 000076      mov      #76,r0      ;clean out all packets and interrupt flags
2877 033102      012701 002346      mov      #rsp1,r1      ;and the command area
2878 033106      005021      $4:      clr      (r1)+
2879 033110      077002      sob      r0,$4
2880
2881 033112      000401      br      end
2882
2883 033114      ABORT:
2884 033114      DOCLN      ;Do clean up and abort the pass
2885 033116      END:
2886 033116      ENDINIT      ;Finished

```

M5

.MAIN. MACRO V05.03 Tuesday 10 Jun-86 13:21 Page 26 1

SEQ 0064

SIZER Supplied Program Data

```
2887
2888
2889 033120      BGNAUTO
2890 033120      DODU LOGUNIT
2891 033126      ENDAUTO
2892
2893 033130      BGNCLN
2894 033130      005077 147170      clr      @IPreg      ;get controller into know state
2895 033134      Break                               ;waste some time
2896 033136      ENDCLN
2897
2898 033140      BGNDU
2899 033140      printf #DRPunt,unit
2900 033164      ENDDU
2901
```

SIZER Supplied Program Data

2903	033166				BGNTST 1				
2904	033166	004737	023550		jsr	pc,hrd'nt		;init the controller	
2905	033172	032737	010000	002336	bit	#bit12,untflgs		;check if just want to park the heads	
2906	033200	001402			beq	3\$			
2907	033202	000137	037420		jmp	dropunt		;jump to end of test where heads are automatically p	
arked									
2908	033206	122737	000023	002340	3\$:	cmpb	#Mrqdx3,mdlnbr	;check if RQDX3 controller	
2909	033214	001403			beq	2\$			
2910	033216	042737	100000	002336	bic	#bit15,untflgs		;if other then RQDX3 than 'mposs'ble to run auto siz	
er or in auto mo									
2911	033224	032737	100000	002336	2\$:	bit	#bit15,untflgs	;test if auto mode 's enabled	
2912	033232	001404			beq	1\$		;if not skip the auto sizer rout ne	
2913	033234	004737	024620		jsr	pc,AUTOsizer		;if 't is then run AUTO SIZER on the controller	
2914	033240	004737	026474		jsr	pc,AUTOd's		;display information from autosizer routine	
2915									
2916	033244				1\$:			; ...	
2917	033244	005077	147054		clr	@IPreg		;can any spurious interrupts	
2918	033250				printb	#ASMSGT		; ...	
2919	033270				ELPcmd:				
2920	033270	000401			br	4\$		; set this to a NOP for APT compatability	
2921	033272	000415			br	3\$		; skip manual question	
2922	033274	005037	002322		4\$:	clr	boot	; WARNING - remove boot diskette f rst	
2923	033300				GMANIL	bot.dev,BOOT, 1,YES		; Insert new diskette	
2924								; DO you want to continue	
2925	033314	005737	002322		tst	BOOT			
2926	033320	001002			bne	3\$		; Yes, run format	
2927	033322	000137	037420		jmp	dropunt		; No, drop unit	
2928	033326				3\$:				
2929									
2930	033326	004737	023550		jsr	pc,hrdint		; Reinit ctrl in case of unknown state	
2931	033332				printb	#pb9,mdl nbr		; Print the disk controller model number	
2932	033356				printb	#pb10,mcdrnbr		; Print microcode version number in dec.	
2933									
2934	033402	032737	100000	002336	bit	#bit15,untflgs		;test if auto mode is enabled	
2935	033410	001011			bne	1\$		;branch if in auto mode else	
2936	033412				GMANID	ASK.prg,PRGnam,A,-1,6.,6.,yes		;ask for the User what local program he wants to run	
2937	033432	000411			br	2\$			
2938	033434				1\$:				
2939	033434	012737	047506	002676	mov	#"F0,PRGnam		;place "FORMAT" into ascii buffer if in auto mode	
2940	033442	012737	046522	002700	mov	#"RM,PRGnam+2			
2941	033450	012737	052101	002702	mov	#"AT,PRGnam+4			
2942	033456				2\$:				
2943	033456				EXLCPRG	PRGnam		;Execute Local program "FORMAT" or what ever they wr	
ote									
	033456	032737	100000	002534	ELP6:	bit	#bit15,cmdrng+2	;test ownership of ring make sure we own 't	
	033464	001374			bne	ELP6		;if we don't own it wa t until we do	
	033466	012737	000022	002446	mov	#22,cmdlen		;load lenght of packet to be send	
	033474	112737	000000	002450	movb	#0,cmdlen+2		;load msg type and credit	
	033502	112737	000002	002451	movb	#dup.id,cmdlen+3		;load DUP connection ID	
	033510	005237	002452		inc	cmdpak		;load new CRN	
	033514	005037	002454		clr	cmdpak+2			
	033520	005037	002456		clr	cmdpak+4			
	033524	005037	002460		clr	cmdpak+6			
	033530	012737	000003	002462	mov	#op.elp,cmdpak+10		;load up opcode	
	033536	012737	000001	002464	mov	#stdaln,cmdpak+12		;stand alone modifier	
	033544	012700	000006		mov	#6,r0		;6 letters transfer	
	033550	012701	002466		mov	#cmdpak+14,r1		;starting address to place program name	
	033554	012702	002676		mov	#PRGnam,r2		;start of Program Name	
	033560	112221			rfdj6:	movb	(r2)+,(r1)+	;add 2 to bycnt then store	
	033562	077002			sob	r0,rfdj6			

SIZER Supplied Program Data

033564	012777	033626	146534	mov	#RFD6,@vector	;New vector place
033572	012737	002352	002526	mov	#rsppak,rsprng	;load response packet area into ring
033600	012737	002452	002532	mov	#cmdpak,cmdrng	;load command packet area into ring
033606	012737	140000	002530	mov	#140000,RSPRNG+2	;Port ownership bit.
033614	012737	100000	002534	mov	#bit15,CMDRNG+2	
033622	004737	022760		jsr	pc,POLLWT	;Go to poll and wait routine.

;*****

033626				RFD6:		;Intr to here.
033626	062706	000006		add	#6,sp	;fix stack for interrupt (4), pollwt subrtn (2)
033632	012777	032650	146466	mov	#intsrvc,@vector	;Change vector
033640	004737	030654		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.

2944								
2945	033644	122737	000011	002371	cmpb	#bit3+bit0,rsppak+17	;is this program a standalone.DUP d'alog type	
2946	033652	001406			beq	1\$		
2947	033654				ERRDF	2,DF3	;Device Fatal can't do remote programs	
2948	033664	000137	037420		jmp	dropunt	;drop unit and go on	
2949	033670				1\$:			
2950	033670				RCDcmd:			
2951	033670				RCVDAT	#datare,#80.		
	033670	032737	100000	002534	RCD7:	bit	#bit15,cmdrng+2	;test ownership of ring make sure we own it
	033676	001374			bne	RCD7	;if we don't own it wait until we do	
	033700	012737	000034	002446	mov	#34,cmdlen	;load lenght of packet to be send	
	033706	112737	000000	002450	movb	#0,cmdlen+2	;load msg type and credit	
	033714	112737	000002	002451	movb	#dup.id,cmdlen+3	;load DUP connection ID	
	033722	005237	002452		inc	cmdpak	;load new CRN	
	033726	005037	002454		clr	cmdpak+2		
	033732	005037	002456		clr	cmdpak+4		
	033736	005037	002460		clr	cmdpak+6		
	033742	012737	000005	002462	mov	#op.rec,cmdpak+10	;load up opcode	
	033750	005037	002464		clr	cmdpak+12	;no modifiers	
	033754	012737	000120	002466	mov	#80.,cmdpak+14		
	033762	005037	002470		clr	cmdpak+16		
	033766	012737	002552	002472	mov	#datare,cmdpak+20	;load address of buffer descriptor	
	033774	005037	002474		clr	cmdpak+22		
	034000	005037	002476		clr	cmdpak+24		
	034004	005037	002500		clr	cmdpak+26		
	034010	005037	002502		clr	cmdpak+30		
	034014	005037	002504		clr	cmdpak+32		

034020	012777	034062	146300	mov	#RFD7,@vector	;New vector place
034026	012737	002352	002526	mov	#rsppak,rsprng	;load response packet area into ring
034034	012737	002452	002532	mov	#cmdpak,cmdrng	;load command packet area into ring
034042	012737	140000	002530	mov	#140000,RSPRNG+2	;Port ownership bit.
034050	012737	100000	002534	mov	#bit15,CMDRNG+2	
034056	004737	022760		jsr	pc,POLLWT	;Go to poll and wait routine.

;*****

034062				RFD7:		;Intr to here.
034062	062706	000006		add	#6,sp	;fix stack for interrupt (4), pollwt subrtn (2)
034066	012777	032650	146232	mov	#intsrvc,@vector	;Change vector

C6

SIZER Supplied Program Data

034074	004737	030654	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.
2952			;;		
2953			;;	get	
2954			;;	r3 = type	
2955			;;	r4 = SA adrs	
2956			;;	r5 = sub number	
2957			;;		
2958	034100	113703	002553	DUPDLG: movb	datare+1,r3 ;get dup type info
2959	034104	006203		asr	r3
2960	034106	006203		asr	r3
2961	034110	006203		asr	r3
2962	034112	006203		asr	r3
2963	034114	042703	177760	bic	#type,r3 ;mask off all but DUP type
2964	034120	013705	002552	mov	datare,r5 ;get dup message number info
2965	034124	042705	170000	bic	#msgnbr,r5 ;clear out top 4 bits
2966					
2967					
2968				;;	
2969				;;	Check for the type.
2970				;;	if QUESTION type, it will be answered by sending
2971				;;	an answer through a Send command which will be followed
2972				;;	by a Receive command to await further instructions.
2973				;;	
2974				;;	If a DEFAULT QUESTION type is given an answer will
2975				;;	either be given or a blank send command returned.
2976				;;	Either way we will do a Send command followed by a
2977				;;	Receive command.
2978				;;	
2979				;;	if INFORMATIONAL type, check message number and type
2980				;;	information according to message number given.
2981				;;	
2982				;;	if FATAL ERROR type, check message number and print
2983				;;	error message accordingly. No other commands will
2984				;;	be given following this type of command.
2985				;;	
2986				;;	If TERMINATION type check the message number and print the
2987				;;	correct message. Usually this implies a succesful
2988				;;	end to the formatter. After this command we exit the program
2989				;;	
2990				;;	If SPECIAL type we are asking for the FCT table to be passed
2991				;;	to the RQDX3 controller. We will send the table with a Send
2992				;;	command and then to a Receive command to proceed.
2993				;;	
2994	034130	022703	000001	qstn: cmp	#Question,r3 ;test for "quest'on" subtype
2995	034134	001117		bne	dfqstn ;if not branch
2996	034136	032737	020000 002336	bit	#bit13,unflgs ;see if we are working on a known controller
2997	034144	001077		bne	qnbra ;if not type out ascii
2998	034146	122737	000106 002676	cmpb	#'F,prgram ;if running the format program then print info
2999	034154	001073		bne	qnbra ;else just go for an answer
3000					
ult	3001	034156	004737 030636	qnbr0: jsr	pc.clrDUPbuf ;clear out data buffer so DRS macros don't show defa
	3002	034162	022705 000000	cmp	#0,r5 ;check for message number
	3003	034166	001036	bne	qnbr7 ;check for next message number
	3004	034170	032737 100000 002336	bit	#bit15,unflgs

D6

SIZER Supplied Program Data

```

3005 034176 001011          bne      1$
3006 034200          GMANID  qfdat,DATE,A,177777,10.,10.,no      ;DATE MM DD YYYY ?
3007 034220 000417          br       2$
3008 034222 012737 033060 002552 1$:  mov     #"06,dateare      ;The date 's not used anyway so any date will do
3009 034230 012737 030455 002554      mov     #" 1,dateare+2      ;I'll be celebrating this day
3010 034236 012737 026467 002556      mov     #"7-,dateare+4
3011 034244 012737 034461 002560      mov     #"19,dateare+6
3012 034252 012737 033070 002562      mov     #"86,dateare+10
3013 034260 000137 035016      2$:  jmp      SDTcmd              ;branch to Send Data command
3014
3015 034264 022705 000007      qnbr7:  cmp     #7,r5              ;check for message number
3016 034270 001025          bne      qnbr8              ;check for next message number
3017 034272 032737 100000 002336      bit     #bit15,unflgs
3018 034300 001011          bne      1$
3019 034302          GMANID  qfser,DATE,A,177777,8.,10.,NO      ;SERIAL NUMBER 9 d'g'its ?
3020 034322 000406          br       2$
3021 034324 013700 002334      1$:  mov     sernbr,r0
3022 034330 012701 002552      mov     #dateare,r1      ;place to stick asc'
3023 034334 004737 030346      jsr     pc,OCTASC      ;convert octal to decimal asc'
3024 034340 000137 035016      2$:  jmp      SDTcmd
3025
3026 034344 004737 030542      qnbr8:  jsr     pc,typDUPbuf      ;type out ASCII sent by disk controller
3027 034350          GMANID  ASK.ANSWER,DATE,A,177777,0.,10.,YES ;give 't an answer
3028 034370 000137 035016      jmp      SDTcmd              ;branch to Send Data command
3029
3030
3031 034374 022703 000002      dfqstn: cmp     #DefQuest,r3      ;test for "Default Question" subtype
3032 034400 001402          beq      1$
3033 034402 000137 035232      jmp     infrm              ;if not branch
3034 034406 032737 020000 002336 1$:  bic     #bit13,unflgs      ;see if we are working on a known controller
3035 034414 001402          beq      2$
3036 034416 000137 034772      jmp     dqnbr8              ;if not type out asc'i
3037 034422 122737 000106 002676 2$:  cnpb    #'F,prgnam      ;if running the format program then print 'nfo
3038 034430 001160          bne      dqnbr8              ;else just go for an answer
3039
3040 034432 004737 030636      dqnbr1: jsr     pc,clrDUPbuf      ;clear out data buffer so DRS macros don't show defa
ult
3041 034436 022705 000001          cmp     #1,r5              ;check for message number
3042 034442 001043          bne      dqnbr4              ;check for next message number
3043
3044 034444 032737 100000 002336      bit     #bit15,unflgs      ;put in message number
3045 034452 001011          bne      3$
3046 034454          GMANID  dfunt,DATE,A,177777,0,3,YES      ;Ask for UNIT NUMBER 0-255 ?
3047 034474 000406          br       4$
3048 034476 013700 002330      3$:  mov     unit,r0              ;get unit number if in auto mode from Hardware P tab
3049 034502 012701 002552          mov     #dateare,r1      ;store decimal ascii conversion in data area
3050 034506 004737 030346      jsr     pc,OCTASC      ;convert octal to asc'i decimal in data area
3051
3052 034512 012701 002552      4$:  mov     #dateare,r1      ;address of ascii decimal data
3053 034516 012700 002330      mov     #unit,r0      ;address to store octal conversion
3054 034522 004737 030434      jsr     pc,ASCDEC      ;convert ascii decimal to octal
3055 034526 022737 000003 002330 2$:  cmp     #3,unit      ;make sure unit number is less than 4 or between 0-3
3056 034534 002004          bge      1$
3057 034536 162737 000004 002330      sub     #4,unit      ;subtract 4 until unit 's less than four
3058 034544 000770          br       2$
3059 034546          1$:
3060
3061 034546 000137 035016      jmp      SDTcmd              ;branch to Send Data command

```

SIZER Supplied Program Data

```

3062
3063 034552 022705 000004      dqnbr4: cmp      #4,r5      ;check for message number
3064 034556 001021      bne      dqnbr5      ;check for next message number
3065 034560 012737 000116 002552      mov      #'N,datare ;set the default for NO
3066 034566 032737 100000 002336      bit      #bit15,unflgs
3067 034574 001010      bne      1$
3068 034576      G1MANID dfbad,DATAARE,A,177777,0,1,YES ;Use existing bad block information (Y or N)?
3069 034616 000137 035016      1$: jmp      SDTcmd ;branch to Send Data command
3070
3071 034622 022705 000005      dqnbr5: cmp      #5,r5      ;check for message number
3072 034626 001021      bne      dqnbr6      ;check for next message number
3073 034630 012737 000131 002552      mov      #'Y,datare ;Set the default for YES
3074 034636 032737 100000 002336      bit      #bit15,unflgs
3075 034644 001010      bne      1$
3076 034646      G1MANID dfdown,DATAARE,A,177777,0,1,YES ;Use Down Line Load (Y or N)?
3077 034666 000137 035016      1$: jmp      SDTcmd ;branch to Send Data command
3078
3079 034672 022705 000006      dqnbr6: cmp      #6,r5      ;check for message number
3080 034676 001035      bne      dqnbr6      ;check for next message number
3081 034700 012737 000116 002552      mov      #'N,datare ;set the default for NO
3082 034706 032737 100000 002336      bit      #bit15,unflgs
3083 034714 001414      beq      1$
3084      ;NO, ask question
3085 034716 013701 002330      mov      unit,r1 ;Yes see if RD51
3086 034722 006301      asl      r1 ;first cylinder entry
3087 034724 062701 026464      add      #msg+4,r1
3088 034730 023711 003102      cmp      UIT0+UITsiz 2,(r1)
3089 034734 001014      bne      2$
3090      ; point to current unit entry
3091      ; Is it an RD51?
3092 034736 012737 000131 002552      mov      #'Y,datare
3093 034744 000410      br      2$
3094 034746      ; NO, all done
3095 034746      1$: G1MANID dfcon,DATAARE,A,177777,0,1,YES ; YES, make question answer yes because
3096 034766 000137 035016      2$: jmp      SDTcmd ; NO FCT tables on RD51
3097      ; set the default for NO
3098      ; and skip question
3099      ;if unknown use default and continue
3100 034772      ;who knows maybe it will be useful some day
3101 034772 004737 030542      dqnbr6: jsr      pc,typDUPbuf ;type out ASCII sent by disk controller
3102      ;type out ASCII sent by disk controller
3103 034776      G1MANID ASK.ANSWER,DATAARE,A,177777,0.,10.,YES ;give it an answer
3104      ;give it an answer
3105 035016      SDTcmd: SENDDAT #datare,#10. ;sent the answer
3106 035016 032737 100000 002534      SDT10: bit      #bit15,cmdrng+2 ;test ownership of ring make sure we own it
3107 035024 001374      bne      SDT10 ;if we don't own it wait until we do
3108 035026 012737 000034 002446      mov      #34,cmdlen ;load lenght of packet to be send
3109 035034 112737 000000 002450      movb     #0,cmdlen+2 ;load msg type and credit
3110 035042 112737 000002 002451      movb     #dup.id,cmdlen+3 ;load DUP connection ID
3111 035050 005237 002452      inc      cmdpak ;load new CRN
3112 035054 005037 002454      clr      cmdpak+2
3113 035060 005037 002456      clr      cmdpak+4
3114 035064 005037 002460      clr      cmdpak+6
3115 035070 012737 000004 002462      mov      #op.sen,cmdpak+10 ;load up opcode
3116 035076 005037 002464      clr      cmdpak+12 ;no modifiers
3117 035102 012737 000012 002466      mov      #10.,cmdpak+14

```

SIZER Supplied Program Data

035110	005037	002470		clr	cmdpak+16	
035114	012737	002552	002472	mov	#datare,cmdpak+20	;load address of buffer descriptor
035122	005037	002474		clr	cmdpak+22	
035126	005037	002476		clr	cmdpak+24	
035132	005037	002500		clr	cmdpak+26	
035136	005037	002502		clr	cmdpak+30	
035142	005037	002504		clr	cmdpak+32	
035146	012777	035210	145152	mov	#RFD10,@vector	;New vector place
035154	012737	002352	002526	mov	#rsppak,rsprng	;load response packet area into ring
035162	012737	002452	002532	mov	#cmdpak,cmdrng	;load command packet area into ring
035170	012737	140000	002530	mov	#140000,RSPRNG+2	;Port ownership bit.
035176	012737	100000	002534	mov	#bit15,CMDRNG+2	
035204	004737	022760		jsr	pc,POLLWT	;Go to poll and wait routine.
;*****						
035210				RFD10:		;Intr to here.
035210	062706	000006		add	#6,sp	;fix stack for interrupt (4), pollwt subrtn (2)
035214	012777	032650	145104	mov	#intsrvc,@vector	;Change vector
035222	004737	030654		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. do another receive cmd
3107	035226	000137	033670	jmp	RCDcmd	
3108						
3109						
3110						
3111	035232	022703	000003	infrm:	cmp #Inform,r3	;test for "Informational" subtype
3112	035236	001046		bne	term	;if not branch
3113	035240	032737	020000 002336	bit	#bit13,unflgs	;see if we are working on a known controller
3114	035246	001036		bne	inbra	;if not type out ascii
3115	035250	122737	000106 002676	cmpb	#'F,prgram	;if running the format program then print info
3116	035256	001032		bne	inbra	
3117						
3118	035260	022705	000000	inbr0:	cmp #0,r5	;check for message number
3119	035264	001012		bne	inbr1	;check for next message number
3120	035266	004737	030636	jsr	pc,clrDUPbuf	;clear out DUP buffer so there is no echo on last AS
CII						
3121	035272			printf	#sfbegt	;format begun
3122	035312	022705	000001	inbr1:	cmp #1,r5	;check for message number
3123	035316	001012		bne	inbra	;check for next message number
3124	035320	004737	030636	jsr	pc,clrDUPbuf	;clear out DUP buffer so there is no echo on last AS
CII						
3125	035324			printf	#sfdont	;format complete
3126						
3127	035344	004737	030542	inbra:	jsr pc,typDUPbuf	;type out ASCII sent by disk controller
3128	035350	000137	033670	jmp	RCDcmd	;do another receive command
3129						
3130						
3131						
3132	035354	022703	000004	term:	cmp #terminat,r3	;test for termination type
3133	035360	001116		bne	ftler	;if not branch
3134	035362	032737	020000 002336	bit	#bit13,unflgs	;see if we are working on a known controller
3135	035370	001076		bne	tnbra	;if not type out ascii
3136	035372	122737	000106 002676	cmpb	#'F,prgram	;if running the format program then branch to error
routine						
3137	035400	001072		bne	tnbra	
3138						
3139	035402	022705	000014	tnbr12:	cmp #12.,r5	;test for sub number #1

SIZER Supplied Program Data

3140	035406	001012		bne	tnbr13		;branch if not sub number #1
3141	035410			printf	#sffcut		
3142	035430	000137	037420	jmp	dropunt		;drop test unit and end pass
3143							
3144	035434	022705	000015	tnbr13:	cmp	#13.,r5	;test for msg number
3145	035440	001052		bne	tnbra		;branch if not right number
3146	035442			printf	#sffcnt		
3147	035462	032737	100000 002336	bit	#bit15,untflgs		;are we in auto mode
3148	035470	001434		beq	2\$		; NO, then we are all done
3149							; YES, is this an RX33
3150	035472	013701	002330	mov	unit,r1		; first cylinder entry
3151	035476	006301		asl	r1		
3152	035500	062701	026464	add	#msg+4,r1		; point to current unit entry
3153	035504	022711	000003	cmp	#3,(r1)		; Is it an RX33?
3154	035510	001024		bne	2\$		; NO, all done
3155							; YES, ask if it wants to continue or not
3156							
3157	035512	005077	144606	GMANIL	clr	@IPreg	; reinit the controller stop spurious interrupts
3158	035516			bot.con,	BOOT, 1,YES		; Do you want to format another?
3159							
3160	035532	005737	002322	tst	BOOT		; Yes, execute local program
3161	035536	001007		bne	1\$		; No, tell him to insert bootable media
3162							
3163	035540			GMANIL	bot.rep,	BOOT, 1,YES	; Please insert boot media and h't return
3164	035554	000402		br	2\$		
3165	035556	000137	033270	1\$:	jmp	ELPcmd	
3166	035562	000137	037420	2\$:	jmp	dropunt	
3167							
3168	035566	004737	030542	tnbra:	jsr	pc,typDUPbuf	;type out ASCII sent by d'sk controller
3169	035572			printf	#PF2		;print finished local program without procedure error
3170	035612	000137	037426	jmp	etst		;end DUP diaglog but stay in test loop
3171							
3172							
3173	035616	022703	000005	ftler:	cmp	#Ftlerr,r3	;test for "Fatal Error" subtype
3174	035622	001402		beq	1\$		
3175	035624	000137	037100	jmp	spcl		;if not branch
3176	035630	032737	020000 002336	1\$:	bit	#bit13,untflgs	;see if we are working on a known controller
3177	035636	001004		bne	3\$		;if not type out ascii
3178	035640	122737	000106 002676	cmpb	#'F,prgram		;if running the format program then branch to error
routine							
3179	035646	001414		beq	2\$		
3180	035650	004737	030542	3\$:	jsr	pc,typDUPbuf	;type out ASCII sent by disk controller
3181	035654			printf	#DF15		;Fatal error reported when running local program
3182	035674	000137	037420	jmp	dropunt		;drop unit and end pass
3183							
3184	035700			2\$:	ERRHRD	1,HRD0	;Hard device error
3185							
3186	035710	022705	000001	fnbr1:	cmp	#1,r5	;test for sub number #1
3187	035714	001012		bne	fnbr2		;branch if not sub number #1
3188	035716			gsts:			
3189	035716			printb	#efstat		; "GET STATUS failure"
3190	035736	000137	037420	jmp	dropunt		;drop unit and end pass
3191							
3192	035742	022705	000002	fnbr2:	cmp	#2.,r5	;test for msg number
3193	035746	001012		bne	fnbr3		;branch if not right number
3194	035750			printf	#efsndt		
3195	035770	000137	037420	jmp	dropunt		;drop unit and end pass
3196							

SIZER Supplied Program Data

3197	035774	022705	000003	fnbr3:	cmp	#3.,r5	;test for msg number
3198	036000	001012			bne	fnbr4	;branch if not right number
3199	036002				printf	#efcmdt	;
3200	036022	000137	037420		jmp	dropunt	;drop unit and end pass
3201				fnbr4:	cmp	#4.,r5	;test for msg number
3202	036026	022705	000004		bne	fnbr5	;branch if not right number
3203	036032	001012			printf	#efrcvt	;
3204	036034				jmp	dropunt	;drop unit and end pass
3205	036054	000137	037420				
3206				fnbr5:	cmp	#5.,r5	;test for msg number
3207	036060	022705	000005		bne	fnbr6	;branch if not right number
3208	036064	001012			printf	#efbust	;
3209	036066				jmp	dropunt	;drop unit and end pass
3210	036106	000137	037420				
3211				fnbr6:	cmp	#6.,r5	;test for msg number
3212	036112	022705	000006		bne	fnbr7	;branch if not right number
3213	036116	001012			printf	#efinit	;
3214	036120				jmp	dropunt	;drop unit and end pass
3215	036140	000137	037420				
3216				fnbr7:	cmp	#7.,r5	;test for msg number
3217	036144	022705	000007		bne	fnbr8	;branch if not right number
3218	036150	001012			printf	#efnut	;
3219	036152				jmp	dropunt	;drop unit and end pass
3220	036172	000137	037420				
3221				fnbr8:	cmp	#8.,r5	;test for msg number
3222	036176	022705	000010		bne	fnbr9	;branch if not right number
3223	036202	001012			printf	#efdxft	;
3224	036204				jmp	dropunt	;drop unit and end pass
3225	036224	000137	037420				
3226				fnbr9:	cmp	#9.,r5	;test for msg number
3227	036230	022705	000011		bne	fnbr10	;branch if not right number
3228	036234	001012			printf	#effcct	;
3229	036236				jmp	dropunt	;drop unit and end pass
3230	036256	000137	037420				
3231				fnbr10:	cmp	#10.,r5	;test for msg number
3232	036262	022705	000012		bne	fnbr11	;branch if not right number
3233	036266	001012			printf	#efsekt	;
3234	036270				jmp	dropunt	;drop unit and end pass
3235	036310	000137	037420				
3236				fnbr11:	cmp	#11.,r5	;test for msg number
3237	036314	022705	000013		bne	fnbr12	;branch if not right number
3238	036320	001012			printf	#efrcct	;
3239	036322				jmp	dropunt	;drop unit and end pass
3240	036342	000137	037420				
3241				fnbr12:	cmp	#12.,r5	;test for msg number
3242	036346	022705	000014		bne	fnbr13	;branch if not right number
3243	036352	001012			printf	#eflbft	;
3244	036354				jmp	dropunt	;drop unit and end pass
3245	036374	000137	037420				
3246				fnbr13:	cmp	#13.,r5	;test for msg number
3247	036400	022705	000015		bne	fnbr14	;branch if not right number
3248	036404	001012			printf	#effcwt	;
3249	036406				jmp	dropunt	;drop unit and end pass
3250	036426	000137	037420				
3251				fnbr14:	cmp	#14.,r5	;test for msg number
3252	036432	022705	000016		bne	fnbr15	;branch if not right number
3253	036436	001012					

SIZER Supplied Program Data

3254	036440			printf	#efrcrt	
3255	036460	000137	037420	no	dropunt	;drop unit and end pass
3256						
3257	036464	022705	000017	fnbr15: cmp	#15.,r5	;test for msg number
3258	036470	001012		bne	fnbr16	;branch if not right number
3259	036472			printf	#efrcwt	
3260	036512	000137	037420	jmp	dropunt	;drop unit and end pass
3261						
3262	036516	022705	000020	fnbr16: cmp	#16.,r5	;test for msg number
3263	036522	001012		bne	fnbr17	;branch if not right number
3264	036524			printf	#efrcft	
3265	036544	000137	037420	jmp	dropunt	;drop unit and end pass
3266						
3267	036550	022705	000021	fnbr17: cmp	#17.,r5	;test for msg number
3268	036554	001012		bne	fnbr18	;branch if not right number
3269	036556			printf	#effcrt	
3270	036576	000137	037420	jmp	dropunt	;drop unit and end pass
3271						
3272	036602	022705	000022	fnbr18: cmp	#18.,r5	;test for msg number
3273	036606	001012		bne	fnbr19	;branch if not right number
3274	036610			printf	#effcnt	
3275	036630	000137	037420	jmp	dropunt	;drop unit and end pass
3276						
3277	036634	022705	000023	fnbr19: cmp	#19.,r5	;test for msg number
3278	036640	001012		bne	fnbr20	;branch if not right number
3279	036642			printf	#effcdt	
3280	036662	000137	037420	jmp	dropunt	;drop unit and end pass
3281						
3282	036666	022705	000024	fnbr20: cmp	#20.,r5	;test for msg number
3283	036672	001012		bne	fnbr21	;branch if not right number
3284	036674			printf	#eftmot	
3285	036714	000137	037420	jmp	dropunt	;drop unit and end pass
3286						
3287	036720	022705	000025	fnbr21: cmp	#21.,r5	;test for msg number
3288	036724	001012		bne	fnbr22	;branch if not right number
3289	036726			printf	#efillt	
3290	036746	000137	037420	jmp	dropunt	;drop unit and end pass
3291						
3292	036752	022705	000026	fnbr22: cmp	#22.,r5	;test for msg number
3293	036756	001012		bne	fnbr23	;branch if not right number
3294	036760			printf	#efwart	
3295	037000	000137	037420	jmp	dropunt	;drop unit and end pass
3296						
3297	037004	022705	000027	fnbr23: cmp	#23.,r5	;test for msg number
3298	037010	000412		br	fnbr24	;branch if not right number
3299	037012			printf	#efinpt	
3300	037032	000137	037420	jmp	dropunt	;drop unit and end pass
3301						
3302						
3303	037036	022705	000030	fnbr24: cmp	#24.,r5	;test for msg number
3304	037042	001012		bne	1\$	
3305	037044			printf	#efmedt	
3306	037064	000137	037420	jmp	dropunt	;drop unit and end pass
3307						
3308	037070	004737	030542	1\$: jsr	pc,typDUPbuf	;type out ASCII sent by disk controller
3309	037074	000137	037420	jmp	dropunt	;drop unit and end pass
3310						

SIZER Supplied Program Data

```

3311
3312
3313
3314 037100 022703 000006      spcl:  cmp    #specl,r3      ;test for special type
3315 037104 001137              bne     unkwn          ;branch if not known
3316 037106 032737 020000 002336 bit    #bit13,untflgs  ;see if we are working on a known controller
3317 037114 001004              bne     2$             ;if not type out ascii
3318 037116 122737 000106 002676 cmpb   #'F,prgnam    ;if running the format program then print info
3319 037124 001414              beq     1$             ;type out ASCII sent by disk controller
3320 037126 004737 030542      2$:  jsr     pc,typDUPbuf  ;special command issued by local program did not kno
3321 037132              prntf   #DF16
w how to handle
3322 037152 000137 037404              jmp     unkwn          ;report error
3323
3324 037156 022705 000002      1$:  cmp     #2,r5          ;test for message number 1
3325 037162 001110              bne     unkwn          ;branch if not known
3326 037164 004737 027542      jsr     pc,blduit      ;go get or build UIT table
3327 037170              SENDDAT UITadr,#UITsiz    ;sent Unit Information table
037170 032737 100000 002534 SDT11: bit    #bit15,cmdrng+2  ;test ownership of ring make sure we own it
037176 001374              bne     SDT11          ;if we don't own it wait until we do
037200 012737 000034 002446      mov     #34,cmdlen    ;load length of packet to be send
037206 112737 000000 002450      movb   #0,cmdlen+2  ;load msg type and credit
037214 112737 000002 002451      movb   #dup.id,cmdlen+3 ;load DUP connection ID
037222 005237 002452              inc     cmdpak        ;load new CRN
037226 005037 002454              clr     cmdpak+2
037232 005037 002456              clr     cmdpak+4
037236 005037 002460              clr     cmdpak+6
037242 012737 000004 002462      mov     #op.sen,cmdpak+10 ;load up opcode
037250 005037 002464              clr     cmdpak+12    ;no modifiers
037254 012737 000104 002466      mov     #UITs'z,cmdpak+14
037262 005037 002470              clr     cmdpak+16
037266 013737 002320 002472      mov     UITadr,cmdpak+20 ;load address of buffer descriptor
037274 005037 002474              clr     cmdpak+22
037300 005037 002476              clr     cmdpak+24
037304 005037 002500              clr     cmdpak+26
037310 005037 002502              clr     cmdpak+30
037314 005037 002504              clr     cmdpak+32

037320 012777 037362 143000      mov     #RFD11,@vector  ;New vector place
037326 012737 002352 002526      mov     #rsppak,rsprng  ;load response packet area into ring
037334 012737 002452 002532      mov     #cmdpak,cmdrng  ;load command packet area into ring
037342 012737 140000 002530      mov     #140000,RSPRNG+2 ;Port ownership bit.
037350 012737 100000 002534      mov     #bit15 CMDRNG+2
037356 004737 022760              jsr     pc,POL T      ;Go to poll and wait routine.

;*****

037362              RFD11:
037362 062706 000006              add     #6,sp          ;Intr to here.
037366 012777 032650 142732      mov     #intsrv,@vector ;fix stack for interrupt (4), pollwt subrtn (2)
037374 004737 030654              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.
                                ;do another receive cmd

3328 037400 000137 033670              jmp     RCDcmd
3329
3330
3331 037404              unkwn:  ERRSF   0,SF0          ; system error unknown response

```

SIZER Supplied Program Data

3332	037414	004737	030730	jsr	pc,PRNTpkt	;type out packet information
3333						
3334	037420			dropunt:		
3335	037420			DODU	LOGUNIT	;drop the unit
3336						
3337	037426			etst:		
3338	037426	023727	002340	000023	cmp	mdl nbr, #mrqdx3
3339	037434	001014			bne	1\$
3340	037436	004737	023550		jsr	pc,hrdint
3341	037442			printb	#Parkdrv	;tell user to wait while parking heads
3342	037462	004737	024620		jsr	pc,autosizer
3343	037466			1\$:		;go park heads
3344	037466				docln	;take controller offline
3345	037470			ENDTST		

SIZER Supplied Program Data

```

3347 037472      BGNHRD
3348
3349 037474      GPRMA ip.adr,0,0,160000,177776,YES ;Get IP reg addr (170000-177776)
3350                                     ;place in word 2 of the table
3351                                     ;default value is from default
3352                                     ;table.
3353
3354 037504      GPRMA vec.adr,2,0,0,776,YES ;Get the vector addr (octal 0-776)
3355                                     ;place in word
3356                                     ;default value is from default
3357                                     ;table.
3358
3359 037514      GPRML prk.hds,10,bit12,YES ;ask if they want to just park the heads
3360
3361 037522      XFERT label0 ;If last gprml input is true (y) transfer
3362
3363 037524      GPRML auto.md,10,bit15,YES ;ask if they want to go in to auto mode
3364                                     ;This will format the drive using the autosizer
3365
3366 037532      XFERF label0 ;If last gprml input is false (n) transfer
3367                                     ;control to label.
3368
3369 037534      GPRMD drv.nbr,4,D,-1,0,255.,YES ;Get the logical drive (decimal 0-255)
3370                                     ;place in word
3371                                     ;default value is from default
3372                                     ;table.
3373
3374
3375 037546      GPRMD ser.nbr,6,D,1,1,012345.,YES ;Get the drive serial number
3376                                     ;place in word
3377                                     ;default value is from default
3378                                     ;table.
3379
3380
3381 037560      label0:
3382
3383 037560      exit hrd
3384 037562      ENDHRD
3385
3386
3387 037562      LASTAD
3388 037566      L$LAST::
3389 037566      ENDMOD
3390      .END

```

000001

Symbol table

A	= 000000	BIT9	= 001000 G	C\$PNTR-	000014	DRVTX4	005165	FNBR18	036602
ABORT	033114	BLDUIT	027542	C\$PNTF-	000017	DRVTX5	005261	FNBR19	036634
ABRT3	024450	BOE	= 000400 G	C\$PNTS=	000016	DRVTX6	005355	FNBR2	035742
ADR	= 000020 G	BOOT	002322	C\$PNTX	000015	DRVTX7	005450	FNBR20	036666
ASCDEC	030434	BOT.CO	007673	C\$PUTB-	000072	DRV.NB	004222	FNBR21	036720
ASK.AN	020543	BOT.DE	007147	C\$PUTW=	000073	DUPDLG	034100	FNBR22	036752
A.K.DB	006746	BOT.RE	007603	C\$QIO	= 000377	DUP.ID-	000002	FNBR23	037004
ASK.LB	007021	CINTR	002522	C\$RDBU-	000007	EFCBST	021517	FNBR24	037036
ASK.PR	006625	CLRDUP	030636	C\$REFG=	000047	EFCMDT	021435	FNBR3	035774
ASK.RB	007074	CMDLEN	002446	C\$REL	= 000077	EFDXFT	021642	FNBR4	036026
ASK.XB	006673	CMDPAK	002452	C\$RESE	000033	EFFCCT	021731	FNBR5	036060
ASMSG	005637	CMDRNG	002532	C\$REVI=	000003	EFFCDT	022425	FNBR6	036112
ASMSGT	006507	CONT	030344	C\$RFLA=	000021	EFFCNT	022401	FNBR7	036144
ASMSG1	005733	CONTON	033012	C\$RPT	= 000025	EFFCRT	022356	FNBR8	036176
ASMSG2	006234	C\$AU	= 000052	C\$SEFG-	000046	EFFCW	022211	FNBR9	036230
ASMSG3	006257	C\$AUTO=	000061	C\$SPRI-	000041	EFILLT	022511	FTLER	035616
ASMSG4	006341	C\$BRK	= 000022	C\$SVEC=	000037	EFINLT	021543	FTLERR=	000005
ASMSG5	006411	C\$BSEG=	000004	C\$TOME-	000076	EFINPT	022664	F\$AU	= 000015
ASMSG6	006463	C\$BSUB=	000002	DATA	002552	EFLBFT	022126	F\$AUTO=	000020
ASMSG7	006021	C\$CLCK=	000062	DBN	002720	EFMEDT	022705	F\$BGN	= 000040
ASMSG8	006066	C\$CLEA=	000012	DECTBL	030420	EFNUT	021606	F\$CLEA=	000007
ASMSG9	006152	C\$CLOS=	000035	DEFQUE=	000002	EFRCC	022037	F\$DU	= 000016
ASSEMB	= 000010	C\$CLP1=	000006	DFBAD	020300	EFRCCFT	022341	F\$END	= 000041
AUTO	025314 G	C\$CPBF=	000074	DFCON	020400	EFRCCRT	022272	F\$HARD=	000004
AUTOBL	030060	C\$CPME=	000075	DFDWN	020350	EFRCCV	021466	F\$HW	= 000013
AUTODI	026474	C\$CVEC=	000036	DFPTBL	002276 G	EFRCCW	022315	F\$INIT=	000006
AUTOEN	026474	C\$DCLN=	000044	DFQSTN	034374	EFSEKT	022020	F\$JMP	= 000050
AUTOSI	024620	C\$DODU=	000051	DFUNT	020237	EFSNOT	021407	F\$MOD	= 000000
AUTOSZ	025270	C\$DRPT=	000024	DF1	011141	EFSTAT	021360	F\$MSG	= 000011
AUTO.M	004305	C\$DU	= 000053	DF11	011425	EFTMOT	022462	F\$PROT=	000021
B	= 000011	C\$EDIT=	000003	DF12	011462	EFUNRG	022730	F\$PWR	= 000017
BIT0	= 000001 G	C\$ERDF=	000055	DF13	011516	EFWART	022563	F\$RPT	= 000012
BIT00	= 000001 G	C\$ERHR=	000056	DF14	011572	EF.CON=	000036 G	F\$SEG	= 000003
BIT01	= 000002 G	C\$ERRO=	000060	DF15	011653	EF.NEW=	000035 G	F\$SOFT=	000005
BIT02	= 000004 G	C\$ERSF=	000054	DF16	011743	EF.PWR=	000034 G	F\$SRV	= 000010
BIT03	= 000010 G	C\$ERSO=	000057	DF2	011203	EF.RES=	000037 G	F\$SUB	= 000002
BIT04	= 000020 G	C\$ESCA=	000010	DF3	011252	EF.STA=	000040 G	F\$SW	= 000014
BIT05	= 000040 G	C\$ESEG=	000005	DF4	011362	ELPCMD	033270	F\$TEST=	000001
BIT06	= 000100 G	C\$ESUB=	000003	DIAGMC=	000000	ELP6	033456	GDS CMD	024264
BIT07	= 000200 G	C\$ETST=	000001	DNINT	024616	END	033116	GDS0	023060
BIT08	= 000400 G	C\$EXIT=	000032	DOUDC	026366 G	ERSEK0=	000003	GDS2	024264
BIT09	= 001000 G	C\$FREQ=	000101	DOURET	026450	ERUDON=	000001	GOBIT	024250
BIT1	= 000002 G	C\$FRME=	000100	DO.CON	004426	ERUINT=	000002	GSTSF	035716
BIT10	= 002000 G	C\$GETB=	000026	DQNBR	034772	ESP4	024620	G\$CNT0=	000200
BIT11	= 004000 G	C\$GETW=	000027	DQNBR1	034432	ETST	037426	G\$DELM=	000372
BIT12	= 010000 G	C\$GMAN=	000043	DQNCR4	034552	EVL	= 000004 G	G\$DISP=	000003
BIT13	= 020000 G	C\$GPHR=	000042	DQNBR5	034622	E\$END	= 002100	G\$EXCP=	000400
BIT14	= 040000 G	C\$GPRI=	000040	DQNBR6	034672	E\$LOAD=	000035	G\$HILI=	000002
BIT15	= 100000 G	C\$INIT=	000011	DROPUN	037420	FNBR1	035710	G\$LOLI=	000001
BIT15T	031606	C\$INLP=	000020	DRPUNT	017771	FNBR10	036262	G\$NO	= 000000
BIT2	= 000004 G	C\$MANI=	000050	DRVTXA	004462	FNBR11	036314	G\$OFFS=	000400
BIT3	= 000010 G	C\$MAP	= 000102	DRVTXB	004511	FNBR12	036346	G\$OFST=	000376
BIT4	= 000020 G	C\$MEM	= 000031	DRVTXC	005543	FNBR13	036400	G\$PRMA=	000001
BIT5	= 000040 G	C\$MMU	= 000103	DRVTX0	004605	FNBR14	036432	G\$PRMD=	000002
BIT6	= 000100 G	C\$MSG	= 000023	DRVTX1	004701	FNBR15	036464	G\$PRML=	000000
BIT7	= 000200 G	C\$OPNR=	000034	DRVTX2	004775	FNBR16	036516	G\$RADA=	000140
BIT8	= 000400 G	C\$OPNW=	000104	DRVTX3	005071	FNBR17	036550	G\$RADB=	000000

Symbol table

G\$RADD=	000040	L\$CCP	002106	G	MRQDX1=	000007	PB11EN	014673	PRI01 =	000040	G
G\$RADL=	000120	L\$CLEA	033130	G	MRQDX3=	000023	PB11ES	015002	PRI02 =	000100	G
G\$RADO=	000020	L\$CO	002032	G	MSECA =	007570	PB11GD	014752	PRI03 =	000140	G
G\$XFER=	000004	L\$DEPO	002011	G	MSEND	026430	PB11OP	014565	PRI04 =	000200	G
G\$YES =	000010	L\$DESC	002126	G	MSG	026450	PB11RD	015113	PRI05 =	000240	G
HIPRGI	002550	L\$DESP	002076	G	MSGDAT	026462	PB11SD	015071	PRI06 =	000300	G
HOE =	100000	L\$DEVP	002060	G	MSGLEN=	000014	PB11ST	014637	PRI07 =	000340	G
HRDINT	023550	L\$DISP	002124	G	MSGNBR=	170000	PB11SO	015162	PRK.HD	004176	
HRDO	012250	L\$DLY	002116	G	MSIN	026410	PB11S1	015207	FRNTPK	030730	
IBE =	010000	L\$DTP	002040	G	MSWAIT	026404	PB11S2	015241	PS0 =	000000	
IDU =	000040	L\$DTYP	002034	G	NEXT	032722	PB11S3	015277	PS7 =	000340	
IER =	020000	L\$DU	033140	G	OCTASC	030346	PB11S4	015334	PTBL	002316	
INBRA	035344	L\$DUT	002072	G	OP.ABR-	000006	PB11S5	015370	QFDAT	020206	
INBRO	035260	L\$DVTY	002166	G	OP.DD =	000001	PB11S6	015417	QFSER	020467	
INBR1	035312	L\$EF	002052	G	OP.ELP=	000003	PB11S9	015444	QFUIT	020131	
INFORM=	000003	L\$ENVI	002044	G	OP.END=	000200	PB11W0	015507	QNBRA	034344	
INFRM	035232	L\$ETP	002102	G	OP.ESP=	000002	PB11W1	015573	QNBRO	034156	
INTSRV	032650	L\$EXP1	002046	G	OP.GDS=	000001	PB12	017674	QNB7	034264	
IPREG	002324	L\$EXP4	002064	G	OP.RD =	000003	PB1201	015664	QSTN	034130	
IP ADR	004144	L\$EXP5	002066	G	OP.REC=	000005	PB1202	015750	QUESTI=	000001	
ISR =	000100	L\$HARD	037474	G	OP.RES=	000000	PB1203	016035	RBN	002746	
IXE =	004000	L\$HIME	002120	G	OP.SEN=	000004	PB1204	016106	RCDCMD	033670	
I\$AU =	000041	L\$HPCP	002016	G	OP.SI1=	000005	PB1205	016147	RCD5	025056	
I\$AUTO=	000041	L\$HPTP	002022	G	OP.S01=	000007	PB1206	016210	RCD7	033670	
I\$CLK =	100006	L\$HW	002276	G	OP.SRD=	000044	PB1207	016262	RD.MOD=	000300	
I\$CLN =	000041	L\$ICP	002104	G	OP.SRP=	000100	PB1208	016335	RETRY =	000367	
I\$DU =	000041	L\$INIT	032674	G	OP.SRX=	000054	PB1209	016371	RFDJ6	033560	
I\$HRD =	000041	L\$LADP	002026	G	O\$APTS=	000000	PB1210	016472	RFD0	023226	
I\$INIT=	000041	L\$LAST	037566	G	O\$AU =	000000	PB1211	016534	RFD10	035210	
I\$MOD =	000041	L\$LOAD	002100	G	O\$BGNR=	000000	PB1212	016570	RFD11	037362	
I\$MSG =	000041	L\$LUN	002074	G	O\$BGNS=	000000	PB1213	016645	RFD2	024412	
I\$PROT=	000040	L\$MREV	002050	G	O\$DU =	000001	PB1214	016711	RFD3	024576	
I\$PTAB=	000041	L\$NAME	002000	G	O\$ERRT=	000000	PB1215	016762	RFD4	025040	
I\$PWR =	000041	L\$PRIO	002042	G	O\$GNSW=	000000	PB1216	017023	RFD5	025250	
I\$RPT =	000041	L\$PROT	032666	G	O\$POIN=	000001	PB1217	017117	RFD6	033626	
I\$SEC =	100016	L\$PRT	002112	G	O\$SETU=	000001	PB1218	017214	RFD7	034062	
I\$SEG =	000041	L\$REPP	002062	G	PARKDR	006512	PB1219	017271	RINTR	002524	
J\$SETU=	000041	L\$REV	002010	G	PARKIT	026160	PB1220	017330	RSPCHK	030654	
J\$SRV =	000041	L\$SPC	002056	G	PBF0	013422	PB1221	017415	RSPPAK	002352	
I\$SUB =	000041	L\$SPCP	002020	G	PBF1	013522	PB1222	017464	RSPRNG	002526	
I\$TST =	000041	L\$SPTP	002024	G	PBF10	014455	PB1223	017557	RSP1	002346	
I\$UDC =	100002	L\$STA	002030	G	PBF2	013651	PB13	013332	RW\$PLL=	140002	
J\$JMP =	000167	L\$TEST	002114	G	PBF3	013725	PB3	012516	R\$CMD =	140012	
LABEL0	037560	L\$TIML	002014	G	PBF4	014021	PB4	012564	R\$DAT =	140010	
LBN	002733	L\$UNIT	002012	G	PBF5	014064	PB5	012636	R\$FPS =	140006	
LOCAL	002312	L10000	002310		PBF6	014131	PB6	012727	SDTCMD	035016	
LOE =	040000	L10002	033116		PBF7	014226	PB7	013031	SDT10	035016	
LOGUNI	002310	L10003	033126		PBF8	014325	PB8	013063	SDT11	037170	
LOPRGI	002546	L10004	033136		PBF9	014415	PB9	013117	SERNBR	002334	
LOT =	000010	L10005	033164		PBSF0	017723	PF2	013335	SER.NB	004250	
LSTCMD	002542	L10006	037470		PB0	012405	PLQC	002314	SETUP	032714	
LSTCRN	002540	L10007	037562		PB1	012434	PNT =	001000	SFBEGT	020550	
LSTVCT	002544	MANBLD	027556		PB10	013161	POLLW	022760	SFCYLT	021234	
L\$ACP	002110	MAXDRV=	000004		PB11	013223	POLLWT	022760	SFD8BT	021016	
L\$APT	002036	MCDNBR	002342		PB11AP	015140	PRGNAM	002676	SFDONT	020571	
L\$AUT	002070	MDLNBR	002340		PB11CR	014515	PRI =	002000	SFFCNT	021307	
L\$AUTO	033120	MOD1	002000	G	PB11EL	015037	PRI00 =	000000	SFFCUT	021255	

Symbol table

SFRBBT	021221	STEPRX	025656	TBQ28	010735	T\$NEST-	177777	UIT2	003210
SFRCBT	020736	SVCGBL	000000	TBQ29	010765	T\$NS0 =	000000	UIT3	003314
SFREVT	020615	SVCINS	177777	TBQ3	010076	T\$NS1 =	000004	UIT4	003420
SFR1T	020637	SVCSUB	177777	TBQ30	011016	T\$PTHV=	***** GX	UIT5	003524
SFR2T	020671	SVCTAG	177777	TBQ31	011044	T\$PTNU=	000000	UIT6	003630
SFTRYT	021156	SVCTST	177777	TBQ32	011106	T\$SAVL =	177777	UIT7	003734
SF10	012273	S\$LSYM	010000	TBQ4	010120	T\$SEGL =	177777	UNIT	002330
SFT1	012344	S\$BUG	026454	TBQ5	010142	T\$SIZE=	***** GX	UNKN	037404
SFXBBT	021076	S\$FLA	026456	TBQ6	010164	T\$SUBN=	000000	UNTD SZ	000002
SFO	012065	S\$RTI	026364 G	TBQ7	010206	T\$TAGL =	177777	UNTFLG	002336
SF1	012134	S\$UDC	026324 G	TBQ8	010230	T\$TAGN=	010010	UNT.NB	006563
SF100	012175	S\$UDI	026342	TBQ9	010252	T\$TEMP=	000000	VECTOR	002326
SIZDRV	026206	TBLBLD	030342	TERM	035354	T\$TEST=	000001	VEC.AD	004157
SIZEND	026222	TBQ0	007745	TERMIN=	000004	T\$TSTM=	177777	WARNIN	004326
SIZEXI	026270 G	TBQ1	010032	TIMOUT	024150	T\$TSTS=	000001	WRNGST	024210
SIZFLP	025614	TBQ10	010274	TNBR	035566	T\$AUT=	010003	W\$CMD =	140022
SIZFPS	025604	TBQ11	010317	TNBR12	035402	T\$CLE=	010004	W\$DAT =	140020
SIZIN	026116	TBQ12	010346	TNBR13	035434	T\$DU =	010005	W\$FPL =	140004
SIZLOP	025522 G	TBQ13	010405	TYPASC	020032	T\$HAR=	010007	XBN	002705
SIZNON	025510	TBQ14	010417	TYPDUP	030542	T\$HW =	010000	X\$ALWA=	000000
SIZRD	026202	TBQ15	010436	TYPE =	177760	T\$INI=	010002	X\$FALS=	000040
SIZRX	025754	TBQ16	010447	T\$ARGC=	000001	T\$PRO=	010001	X\$OFFS=	000400
SIZRX3	026050	TBQ17	010474	T\$CODE=	001004	T\$TES=	010006	X\$TRUE=	000020
SIZSET	025432	TBQ18	010513	T\$ERRN=	000000	T1	033166 G	\$2	033052
SIZWIN	026056	TBQ19	010532	T\$EXCP=	000000	UAM	000200 G	\$3	033072
SIZWT	025412	TBQ2	010054	T\$FLAG=	000041	UIN	002344	\$4	033106
SPCL	037100	TBQ20	010565	T\$FREE=	***** GX	UITADR	002320	.A.DEF=	000040
SPECL	000006	TBQ21	010615	T\$GMAN=	000000	UITDF	004040	.A.FAT=	000120
SP2INT	023672	TBQ22	010647	T\$HILI=	030071	UITLOC	030312	.A.INF=	000060
SP3INT	023762	TBQ23	010662	T\$LAST=	000001	UITOTH=	000010	.A.QUE=	000020
SP4INT	024042	TBQ24	010675	T\$LOLI=	000001	UITSIZ=	000104	.A.TER=	000100
STDALN	000001	TBQ25	010710	T\$LSYM=	010000	UITO	003000	.A.TYP=	000020
STEPMO	025750	TBQ26	010723	T\$LTNO=	000001	UIT1	003104	.B.SPL=	000140
STEPOU	025734								

. ABS. 037566 000 (RW,I,GBL,ABS,OVR)
 000000 001 (RW,I,LCL,REL,CON)
 Errors detected: 0

*** Assembler statistics

Work file reads: 332
 Work file writes: 338
 Size of work file: 39752 Words (156 Pages)
 Size of core pool: 19684 Words (75 Pages)
 Operating system: RSX 11M/PLUS (Under VAX/VMS)

Elapsed time: 00:04:22.27
 ZRQCCO.ZRQCCO.LST/-SP=SVC35R/ML,ZRQCCO.MAC