

RM03,02

PERFORMANCE EXERCISER
CZRMBB0

AH-995B-MC

JAN 1978

COPYRIGHT © 1977

digital

FICHE 1 OF 2

MADE IN USA

RM03,02

PERFORMANCE EXERCISER
CZRMBB0

AH-995B-MC

COPYRIGHT © 1977

FICHE 2 OF 2

JAN 1978

digital

MADE IN USA

EDFICZRMACEQ
PDP8040001
CZRM88.P11

00010000 780105
CZRM880 RM03/2 PERF EXER
21-NOV-77 15:34

POP10 411 HDRICZRM88SEQ
MACY11 30(1046) 22-NOV-77 10:06 PAGE 2

00010000 780105

.REM 2

IDENTIFICATION

PRODUCT CODE: AC-A994B-MC

PRODUCT NAME: CZRM880 RM03/RM02 PERFORMANCE EXERCISER

DATE CREATED: AUGUST 1977

MAINTAINER: DIAGNOSTIC ENGINEERING

AUTHOR: C. CHEN

COPYRIGHT (C) 1977 DIGITAL EQUIPMENT CORP., MAYNARD, MASS.

THE INFORMATION IN THIS STATEMENT IS SUBJECT TO CHANGE
WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A
COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

ACTUAL DISTRIBUTION OF THE SOFTWARE DESCRIBED IN THIS
DOCUMENT WILL BE SUBJECT TO TERMS AND CONDITIONS TO
BE ANNOUNCED ON SOME FUTURE DATE BY DIGITAL EQUIPMENT
CORPORATION.

DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR
RELIABILITY OF ITS SOFTWARE ON EQUIPMENT WHICH IS
NOT SUPPLIED BY DEC.

THIS SOFTWARE IS FURNISHED TO PURCHASER UNDER A LICENSE
TO USE ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH
INCLUSION OF DEC'S COPYRIGHT NOTICE) ONLY FOR USE IN SUCH
SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING
BY DEC.

CONTENTS

1. ABSTRACT
2. REQUIREMENTS
 - 2.1 EQUIPMENT
 - 2.2 MEDIA
 - 2.3 PRELIMINARY PROGRAMS
3. OPERATING THE PROGRAM
 - 3.1 LOADING THE PROGRAM
 - 3.2 STARTING THE PROGRAM
 - 3.3 RESTARTING THE PROGRAM
 - 3.4 PROGRAM CONTROL
 - 3.5 PASS/TEST TERMINATION
 - 3.5.1 PASS TERMINATION
 - 3.5.2 TEST TERMINATION
 - 3.6 RUN TIME
 - 3.6.1 DATA TRANSFER MODE
 - 3.6.2 SEEK VERIFICATION MODE
 - 3.7 UNIBUS & VECTOR ADDRESSES
 - 3.8 DUAL PORT OPERATION
 - 3.9 APT
 - 3.10 APT ENVIRONMENTAL TABLE DEFINITIONS
4. CONTROLLING THE PROGRAM
 - 4.1 DATE & OPERATOR IDENTIFICATION
 - 4.2 PARAMETERS
 - 4.2.1 PROGRAM CONTROL PARAMETERS
 - 4.2.2 PERIPHERAL DEVICE ADDRESSES
 - 4.2.3 PARAMETERS FOR THE FIRST OPERATION
 - 4.3 SWITCH REGISTER SETTINGS
 - 4.4 KEYBOARD COMMANDS
 - 4.4.1 'T' COMMAND
 - 4.4.2 'D' COMMAND
 - 4.4.3 'S' COMMAND
 - 4.4.4 'W' COMMAND
 - 4.4.5 'R' COMMAND
 - 4.4.6 'WT' COMMAND
 - 4.4.7 GENERAL COMMAND INFORMATION
5. PERFORMANCE SUMMARY TYPEOUT
 - 5.1 PERFORMANCE SUMMARY TYPEOUT EXPLANATION
 - 5.2 HARD/SOFT ERROR DEFINITIONS
 - 5.2.1 HARD ERRORS
 - 5.2.2 SOFT ERRORS
6. DATA CHECKING & ERROR RECOVERY
 - 6.1 DATA BUFFER COMPARISON
 - 6.2 VERIFICATION OF DATA WRITTEN

103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121

- 6.3 SECTOR REFORMATTING
- 6.4 BAD TRACK/SECTOR FLAGGING
- 7. ERROR MESSAGES
 - 7.1 ERROR DESCRIPTION LINES
 - 7.2 DETAIL ERROR LINES
- 8. PROGRAM DESCRIPTION
 - 8.1 HOW THE PROGRAM OPERATES
 - 8.2 DUAL PORT OPERATION
 - 8.3 HOW VARIABLES ARE SELECTED FOR EACH OPERATION
 - 8.4 DATA PATTERNS
- 9. RM03 SOFTWARE DRIVER DOCUMENT

1. ABSTRACT

THE RM03 PERFORMANCE EXERCISER PROGRAM IS DESIGNED TO PERFORM AN INTERACTIVE TEST ON RM03 DISK DRIVES CONNECTED TO A MASSBUS SUBSYSTEM. THE DRIVES MAY BE CONTROLLED BY EITHER AN RH11 OR AN RH70. IN ADDITION TO PERFORMING AN INTERACTIVE TEST OF THE DISK DRIVES ON THE SUBSYSTEM, THE PROGRAM IS INTENDED TO BE USED TO VERIFY THAT THE DRIVES UNDER TEST ARE PERFORMING TO THEIR DATA ERROR RATE AND SEEK ERROR RATE (SEE ERROR RATE SPECIFICATIONS).

THE RM03 PERFORMANCE EXERCISER PROGRAM WILL EXERCISE DRIVES CONNECTED AS EITHER SINGLE OR DUAL PORT UNITS. DUAL PORT DRIVES ARE TESTED BY LOADING AND RUNNING THE PROGRAM FROM BOTH CONTROLLING SYSTEMS. THE PROGRAM WILL EXERCISE A MIXED SYSTEM OF DUAL PORT AND SINGLE PORT DRIVES.

TO OBTAIN INTERACTIVE TESTING, OPERATIONS ON THE DRIVES ARE OVERLAPPED (OTHER DRIVES ARE PERFORMING SEEK/SEARCH OPERATIONS WHILE ONE DRIVE IS PERFORMING A DATA TRANSFER OR WRITE CHECK OPERATION). OPERATIONS AMONG THE DRIVES ARE OPTIMIZED SO THAT A HIGH SUBSYSTEM DATA TRANSFER RATE OR A HIGH POSITIONING OPERATION RATE IS MAINTAINED.

THE PERFORMANCE OF EACH DRIVE IS MONITORED BY THE PROGRAM. IF A DRIVE EXCEEDS A PRESET NUMBER OF ERRORS IN ANY OF SEVERAL CATEGORIES, THAT DRIVE IS AUTOMATICALLY DEASSIGNED. (THE OPERATOR MAY OVERRIDE THE AUTOMATIC DEASSIGNMENT FEATURE.) THE PROGRAM REPORTS PERFORMANCE STATISTICS FOR EACH DRIVE BEING EXERCISED ON REQUEST FROM THE OPERATOR OR AUTOMATICALLY AT AN INTERVAL DETERMINED BY THE OPERATOR.

ALL DATA TRANSFER COMMANDS ARE USED (I.E., WRITE DATA, WRITE HEADER & DATA, READ DATA, AND READ HEADER & DATA) AS WELL AS WRITE CHECK DATA AND WRITE CHECK HEADER & DATA COMMANDS. RECALIBRATE AND READ-IN PRESET COMMANDS ARE USED AT STARTUP AND DRIVE INITIALIZATION. RECALIBRATE, OFFSET, AND RETURN-TO-CENTERLINE COMMANDS ARE USED DURING ERROR RECOVERY.

THE DATA TRANSFER COMMANDS ARE SELECTED RANDOMLY EXCEPT FOR THE WRITE CHECK COMMANDS. THE WRITE CHECK COMMANDS ARE USED TO VERIFY A PREVIOUS WRITE OPERATION. THUS, WHEN A WRITE COMMAND IS SELECTED, THE DATA WRITTEN IS VERIFIED BY THE APPROPRIATE WRITE CHECK COMMAND.

DEPENDING UPON WHETHER PROGRAM HAS BEEN LOADED VIA APT AUTOMATIC MODE OR APT DUMP MODE WILL DETERMINE WHETHER: PROGRAM/OPERATOR COMMUNICATIONS ARE THROUGH THE KEYBOARD, DYNAMIC PROGRAM OPTIONS ARE SELECTED VIA SWITCH REGISTER SETTINGS AND ERRORS ARE REPORTED ON THE TELETYPE.

ALL COMMANDS, DATA PATTERNS, AND DATA BUFFER SIZES ARE SELECTED RANDOMLY BY THE PROGRAM. ADDITIONALLY THE ADDRESSES (EG, CYLINDER, TRACK, AND SECTOR) FOR EACH OPERATION ARE SELECTED RANDOMLY.

2. REQUIREMENTS

2.1 EQUIPMENT

REQUIRED

PDP-11 PROCESSOR
16K MEMORY (20K IF THE PROGRAM IS INCLUDED IN AN 'XXDP' CHAIN
TELETYPE
PROGRAM LOADING DEVICE
KW11-L OR KW11-P CLOCK
RH11 OR RH70 WITH 1 RM03 DISK DRIVE

OPTIONAL

ADDITIONAL MEMORY TO A MAXIMUM OF 28K
1 TO 7 ADDITIONAL RM03'S ON THE SAME RH11 OR RH70

2.2 MEDIA

THE RM03 PERFORMANCE EXERCISER PROGRAM REQUIRES FORMATTED DISK
PACKS GENERATED BY THE RM03, RM02 FORMATTER PROGRAM (CZRMACO).
THE PACKS MUST BE FORMATTED IN 32 SECTOR (16 BIT) MODE; THE
ALTERNATE (30 SECTOR - 18 BIT) MODE IS NOT SUPPORTED.

2.3 PRELIMINARY PROGRAMS

RM03 DISKLESS DIAGNOSTIC
RM03 FUNCTIONAL TEST

176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215

3. OPERATING THE PROGRAM

3.1 THE PROGRAM MAY BE LOADED FROM PAPER TAPE USING THE ABSOLUTE LOADER OR IT MAY BE LOADED FROM THE APPROPRIATE 'XXDP' MEDIA USING THE ASSOCIATED LOADER. THE PROGRAM MAY BE INCLUDED IN AN 'XXDP' CHAIN. IF THE PROGRAM IS BEING RUN ON A PROCESSOR WITH 16K, THE 'XXDP' LOADER WILL NOT BE PRESERVED. THE PROGRAM MUST BE RUN ON A SYSTEM WITH 20K OR MORE TO PRESERVE THE 'XXDP' LOADER. THE 'ABSOLUTE' LOADER WILL BE PRESERVED IN A 16K SYSTEM, HOWEVER.

3.2 THE PROGRAM STARTS AT LOCATION 200(8). PARAMETERS NOT INCLUDED IN THE TELETYPE DIAGLOGUE GROUP MUST BE CHANGED BEFORE THE PROGRAM IS STARTED.

THE PROGRAM PERFORMS "TEST" OPERATION TO ALL DRIVES UNDER TESTING AT THE 200(8) STARTING LOCATION AUTOMATICALLY.

3.3 START THE PROGRAM AT LOCATION 204(8) IF THE RH11 OR THE RH70 IS NOT AT ADDRESS 176700.

3.4 PROVIDED THE PROGRAM HAS BEEN LOADED AND STARTED VIA THE APT DUMP MODE OR THE DIAGNOSTIC IS RUNNING IN STAND ALONE PROCESSOR/DRIVE OPERATIONS ARE INITIATED AND CONTROLLED BY KEYBOARD COMMANDS AND SWITCH REGISTER SWITCH SETTINGS. IF THIS IS THE PROGRAM'S FIRST START, THE STATUS OF THE DRIVES ON THE SELECTED MASSBUS SUBSYSTEM WILL BE TYPED OUT. ON SUBSEQUENT STARTS, THIS TYPEOUT MAY BE INHIBITED BY SETTING SW<02>= 1.

HOWEVER, IF THE PROGRAM IS LOADED VIA APT SCRIPT MODE ALL SETUP AND SWITCH REGISTER SETTINGS WILL BE PROVIDED THROUGH THE APT E TABLE. TYPEOUTS FROM THE USER DIAGNOSTIC MAY OR MAYNOT BE INHIBITED DEPENDING UPON WHETHER OR NOT THE APPROPRIATE BIT IN THE E TABLE HAS BEEN SET.

3.5 PASS/TEST TERMINATION

A PASS IS DETERMINED BY EITHER BITS READ OR SEEKS PERFORMED. THE NUMBER OF BITS OR SEEKS REQUIRED FOR A PASS IS DERIVED FROM EITHER THE SOFT ERROR RATE SPECIFICATION OR THE SEEK ERROR RATE SPECIFICATION. THE SPECIFICATIONS FOR THE RM03'S SPECIFY NO MORE THAN 1 SOFT ERROR (NON-PACK RELATED) IN 1×10^{19} BITS READ OR NO MORE THAN 1 SEEK ERROR IN 1×10^{16} SEEKS. THE NUMBER OF BITS OR SEEKS DETERMINING A PASS WERE SELECTED TO PROVIDE A 90% CONFIDENCE LEVEL THAT THE DRIVE IS PERFORMING TO THE APPLICABLE SPECIFICATION.

3.5.1 PASS TERMINATION

END OF PASS FOR A SINGLE DRIVE MAY BE DETERMINED BY EITHER OF THE FOLLOWING CONDITIONS: THE END OF PASS CONDITION USED IS DETERMINED BY PARAMETER 'ENDING'.

A. IF PARAMETER 'ENDING' IS 1, END OF PASS OCCURS WHEN THE DRIVE HAS READ 1.875×10^{18} WORDS (3×10^{19} BITS).

216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271

B. IF PARAMETER 'ENDING' IS 0, END OF PASS OCCURS WHEN THE DRIVE HAS PERFORMED 3×10^6 SEKS.

3.5.2 TEST TERMINATION

THE TEST FOR A DRIVE IS TERMINATED (SW<04> = 0) WHEN:

- A. THE DRIVE HAS COMPLETED THE NUMBER OF PASSES SPECIFIED IN PARAMETER 'PASSES'.
- B. THE TOTAL ERRORS ACCUMULATED EXCEED 100.
- C. A FATAL ERROR OCCURS: EM12 OR EM14.
- D. OPERATOR DEASSIGNS THE DRIVE

3.6 RUN TIME

THE EXERCISER PROGRAM MAY BE RUN IN TWO MODES. THE MODE IS DETERMINED BY THE VALUE IN PARAMETER 'SIZE'. IF 'SIZE' IS ONE SECTOR, THE PROGRAM RUNS IN A SEEK HEAVY MODE; IF 'SIZE' APPROACHES $1/2$ TRACK IN SIZE (4096 DECIMAL) THE PROGRAM RUNS IN A DATA TRANSFER HEAVY MODE. THE PROGRAM RUN TIME VARIES GREATLY DEPENDING ON THE OPERATION MODE SELECTED, THE MEMORY AVAILABLE OVER 16K, THE READ/WRITE RATIO PARAMETER - 'RATIO', AND BY SWITCHES 0, 1, & 2.

3.6.1 DATA TRANSFER MODE

- 1 DRIVE - APPROXIMATELY 3.6 HRS (TO REACH 1.875×10^8 WORDS)
- TO
- 8 DRIVES - APPROXIMATELY 16 HRS (FOR ALL DRIVES TO REACH 1.875×10^8 WORDS)

IF THE PROGRAM IS RUN WITH BOTH SW<00> AND SW<01> SET, THE RUN TIMES SHOULD BE ABOUT 20% FASTER.

3.6.2 SEEK VERIFICATION MODE

PARAMETER 'SIZE' = 1 SECTOR (256 WORDS)
PARAMETER 'MAXTRK' = 'MINTRK'
PARAMETER 'MAXSEC' = 'MINSEC'
SW<00> = 1 (READ ONLY MODE)

- 1 DRIVE - APPROXIMATELY 25 HRS (3×10^6 SEKS)
- TO
- 8 DRIVES - APPROXIMATELY 40 HRS (3×10^6 SEKS FOR ALL DRIVES)

3.7 UNIBUS & VECTOR ADDRESSES

THE PROGRAM ASSUMES THE FOLLOWING UNIBUS AND VECTOR ADDRESSES. (REFER TO SECTION 4.2.2 FOR THE LOCATIONS AT WHICH TO CHANGE THESE ADDRESSES.)

UNIT	UNIBUS ADDRESS	VECTOR ADDRESS
----	-----	-----
RH11 OR RH70	176700	254
TTY PRINTER	177564	NOT USED

TTY KEYBOARD	177560	60
KW11-L	177546	100
KW11-P	172542	104

3.8 DUAL PORT OPERATION

- A. LOAD THE RMO3 PERFORMANCE EXERCISER PROGRAM INTO BOTH PROCESSORS.
- B. SWITCH THE 'CONTROLLER SELECT' SWITCH TO 'A/B' ON EACH DRIVE WHICH IS TO BE TESTED AS A DUAL PORT DRIVE; CYCLE THE DRIVES UP.
- C. START THE PROGRAM IN EACH PROCESSOR. RUN THE PROGRAM AS THOUGH EACH PROCESSOR WERE RUNNING INDEPENDENTLY OF THE OTHER.

3.9 APT

THIS PROGRAM IS APT COMPATIBLE TO THE EXTENT THAT APT HOOKS WILL BE IN THE PROGRAM AND WILL WORK THRU THE 'OPTION INTERFACE'.

FOR OTHER INTERFACES, APT MAY ONLY LOAD AND START THE PROGRAM. I.E. LOAD AND DUMP MODE.

AUTOMATIC MODE (MONITOR)

1. THE INPUT DIALOGUE IS BYPASSED.
 2. THE BUSS ADDRESS AND CONTROLLER INTERRUPT VECTOR IS DEFAULTED.
- DUMP MODE: INPUT DIALOGUE AFTER PROGRAM STARTS

3.10 APT ETABLE DEFINITIONS

THE FOLLOWING DEFINITIONS ARE VALID FOR SPECIFYING APT ENVIRONMENTAL TABLE (ETABLE) ENTRIES, VIA RUNNING THE APT UTILITY PROGRAM "TSP":

1. SOFTWARE ENVIRONMENT:

= 1 IF APT SCRIPT MODE
= 0 IF STANDLONE MODE

2. ENVIRONMENT MODE:

BIT 7 = 1 ETABLE DOES SIZING
= 0 PROGRAM DOES SIZING

BIT 6 = 1 SPOOL MESSAGES TO APT IF SCRIPT MODE
= 0 DON'T SPOOL TO APT

BIT 5 = 1 SUPPRESS CONSOLE OUTPUT
= 0 ALLOW CONSOLE OUTPUT

BIT 4 TO BIT 0 ARE NOT USED

384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
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

3. SWITCH 1 (SOFTWARE SWITCH REGISTER)
IF ENVIRONMENT MODE BIT 7 (SIZING BIT) IS SET TO 1,
THE SOFTWARE SWITCH REGISTER WILL BE USED, INSTEAD
OF THE HARDWARE CONSOLE SWITCH REGISTER.
4. SWITCH 2 (USER SWITCH REGISTER)
NOT USED
5. CPU OPTIONS
NOT USED
6. MEMORY TYPES 1-4 AND MAX MEMORY ADDRESSES
NOT USED
7. INTERRUPT VECTOR 1:
USED WHEN ENVIRONMENT MODE BIT 7 = 1;DEFAULT = 254
8. BUS PRIORITY 1:
NOT USED.
9. INTERRUPT VECTOR 2:
NOT USED
10. BUS PRIORITY 2:
NOT USED
11. BASE ADDRESS:
USED WHEN ENVIRONMENT MODE BIT 7 = 1;DEFAULT = 176700
12. DEVICE MAP:
NOT USED
13. CONTROLLER DESCRIPTOR WORDS:
NOT USED
14. CONTROLLER DESCRIPTOR WORDS:
NOT USED

4. CONTROLLING THE PROGRAM

THE FOLLOWING KEYBOARD CONVENTIONS ARE USED BY THE KEYBOARD ENTRY ROUTINES IN THE PROGRAM:

- A. TO DELETE AN INCORRECT CHARACTER FROM AN ENTRY STRING, TYPE A 'RUBOUT' ('RO'). TYPING A 'RO' WILL DELETE SUCESSIVE CHARACTERS FROM THE INPUT.
- B. TO DELETE AN ENTIRE LINE, TYPE A 'CONTROL U' ('IU').
- C. AN ENTRY MUST BE TERMINATED BY EITHER A 'CARRIAGE RETURN' OR A 'PERIOD'. THE 'PERIOD' TERMINATION IS RECOGNIZED BY THE PROGRAM AS A DEFAULT ENTRY REQUEST. WHEN A LINE IS TERMINATED BY A 'PERIOD' INSTEAD OF A 'CARRIAGE RETURN', THE PROGRAM WILL ACCEPT THE ENTERED VALUE AND WILL DEFAULT TO THE PRELOADED VALUES FOR ANY REMAINING ENTRIES.
- D. IF A 'CONTROL C' IS TYPED DURING KEYBOARD ENTRY, THE PROGRAM WILL RETURN TO THE BEGINNING OF THE GROUP BEING ENTERED.

4.1 DATE & OPERATOR IDENTIFICATION

ASSUMING THE DIAGNOSTIC HAS BEEN LOADED BY ANY OTHER MODE OTHER THAN THE APT SCRIPT MODE THE PROGRAM WHEN IT IS INITIALLY STARTED, WILL ASK FOR DATE AND OPERATOR I.D. ENTRIES. (THE REQUEST FOR THESE ENTRIES OCCURS ONLY WHEN THE PROGRAM IS FIRST STARTED AND WILL NOT APPEAR WHEN THE PROGRAM IS RESTARTED.) THESE ENTRIES ARE OPTIONAL AND MAY BE BYPASSED BY ENTERING A 'CARRIAGE RETURN' IN RESPONSE TO THE REQUEST. THE PROGRAM DOES NOT EDIT OR CHECK EITHER ENTRY. UP TO 8 CHARACTERS OF DATE INFORMATION AND UP TO 6 CHARACTERS OF OPERATOR IDENTIFICATION MAY BE ENTERED. BOTH THE DATE AND THE OPERATOR I.D. WILL BE TYPED WHEN THE 'SA' COMMAND IS PREFORMED (SEE SECTION 4.4.3).

4.2 PARAMETERS

WHEN THE PROGRAM IS STARTED FROM LOCATION 204, THE OPERATOR WILL BE ASKED TO ENTER PARAMETERS. THE FOLLOWING MESSAGE WILL BE DISPLAYED:

ENTER PARAMETERS:

THE OPERATOR MUST ENTER A 'Y' TERMINATED BY A CARRIAGE RETURN (CR) IF PARAMETER ENTRIES ARE TO BE MADE. ANY OTHER CHARACTER IS ACCEPTED AS A 'NO' ENTRY. THE PROGRAM WILL IDENTIFY THE PARAMETER BY THE NAME GIVEN BELOW, DISPLAY THE CURRENT VALUE OF THE PARAMETER AND WAIT FOR THE ENTRY. THE PROGRAM WILL TYPE 'INVALID ENTRY' IF THE ENTRY IS NOT CORRECT AND WAIT FOR A CORRECT ENTRY TO BE TYPED.

NOTE: WHEN THE DIAGNOSTIC IS LOADED VIA APT SCRIPT MODE ALL PARAMETERS ARE LOADED FROM THE APT SYSTEM E TABLE. THIS INCLUDES THE SOFTWARE SWITCH REGISTER. THEREFORE A WORST CASE CONDITION WILL BE SET IN THE E TABLE FOR NORMAL AUTOMATIC OPERATION.

4.2.1 KEYBOARD ENTRY PARAMETERS

NAME	BASE	DEFAULT VALUE	VALUE RANGE
----	----	-----	-----
SIZE	10	(SEE NOTE)	
PASSES	10	1	1 - 999
MINUTE	10	5	0 - 256
RANDOM	8	000000	0 OR 1
ENDING	8	000001	0 OR 1
FORMAT	8	000001	0 OR 1
RATIO	8	000003	0 - 7
MESSAGE	8	000001	0 OR 1

FUNCTION

CONTROLS THE MAXIMUM BUFFER
SIZE USED FOR DATA TRANSFERS
NUMBER OF PASSES TO END OF
TEST.
DETERMINES THE INTERVAL (IN
MINUTES) BETWEEN AUTOMATIC
PERFORMANCE SUMMARY TYPEOUTS
ERRORS PRINTED OUT IF SW<07>=0
IF PARAMETER = 0, THE DATA
TRANSFER WORD COUNT IS RANDOMLY
SELECTED BETWEEN 4 AND
THE VALUE IN 'SIZE'.
IF PARAMETER = 1, THE DATA
TRANSFER WORD COUNT WILL
BE THE VALUE IN 'MAXDL'.
IF PARAMETER = 1, END OF PASS
IS DETERMINED BY THE 'WORDS READ'
COUNT.
IF PARAMETER = 0, END OF PASS
IS DETERMINED BY THE NUMBER
OF SEES.
IF PARAMETER = 0: DO NOT
PERFORM WRITE HEADER & DATA
ORDERS; IF PARAMETER > 0
PERFORM WRITE HEADER & DATA
ORDERS
CONTROLS THE APPROXIMATE
RATIO OF READ TO WRITE
ORDERS.

VALUE R/W RATIO

0	15/1
1	7/1
2	6/2
3	5/3
4	4/4
5	3/5
6	2/6
7	1/7

IF PARAMETER = 1, DO NOT PRINT
ERROR MESSAGES FOR DATA ERRORS
OCCURRING AT LOCATIONS DEFINED
BY THE OPERATOR AS BAD PACK
LOCATION.
IF PARAMETER = 0, PRINT ERROR
MESSAGES ASSOCIATED WITH BAD
PACK LOCATIONS.

PATTERN 10 000000 0 - 15

IF PARAMETER=0, DATA
PATTERN IS RANDOMLY
SELECTED.
IF PARAMETER>0, SPECIFIES
ONE OF THE 15 PATTERNS.
THE SELECTED DATA PATTERN
IS POINTED BY THE PARAMETER
"PATTERN".
IF PARAMETER=0, RANDOM
DATA BLOCK ADDRESS IS
USED IN "TEST" COMMAND.
IF PARAMETER=1, SEQUENTIAL
DATA BLOCK IS USED IN
"TEST" COMMAND.

HEADER 8 000001 0 OR 1

NOTE: THE PROGRAM WILL SELECT A MAXIMUM BUFFER SIZE WHICH
IS DETERMINED BY THE MEMORY AVAILABLE. THE MAXIMUM BUFFER
SIZE ASSIGNED BY THE PROGRAM IS 8192 (10) WORDS. THE
OPERATOR MAY SPECIFY ANY OTHER MAXIMUM SIZE AS LONG AS THE
VALUE SPECIFIED IS AT LEAST 4 WORDS BUT NO LARGER THAN
THE INITIAL VALUE OF 'MAXDL' DETERMINED BY THE PROGRAM.

4.2.2 PERIPHERAL ADDRESSES AND OTHER LOCATIONS OF INTEREST

TO ALTER THESE LOCATIONS, THE OPERATOR MUST MAKE MANUAL ENTRIES
BEFORE THE PROGRAM IS STARTED. THE KEYBOARD ENTRY ROUTINE DOES
NOT PROVIDE ACCESS TO THESE LOCATIONS.

LOC	TAG	CONTENTS	FUNCTION
1160	\$TKS	177560	TTY KEYBOARD STATUS REGISTER
1162	\$TKB	177562	TTY KEYBOARD BUFFER REGISTER
1164	\$TPS	177564	TTY PRINTER STATUS REGISTER
1166	\$TPB	177566	TTY PRINTER BUFFER REGISTER
1274	\$LKCSR	172540	ADDRESS OF KW11-P STATUS REGISTER
1276	\$LKCSB	172542	ADDRESS OF KW11-P COUNTER BUFFER
1300	\$LPVEC	104	KW11-P VECTOR ADDRESS
1302	\$LKS	177546	ADDRESS OF KW11-L STATUS REGISTER
1304	\$LLVEC	100	KW11-L VECTOR ADDRESS
1312	HZ	74	74 (60 DECIMAL) IF SYSTEM IS 60 HZ; 62 (50 DECIMAL) IF SYSTEM IS 50 HZ.

THE RH11-RH70 ADDRESS AND VECTOR MAY BE CHANGED WHEN THE PROGRAM
IS STARTED FROM LOCATION 204(8) OR IF THE PROGRAM DOES NOT RECEIVE
A RESPONSE WHEN IT ACCESSES THE DEFAULT RH11-RH70 ADDRESS.

4.2.3 PARAMETERS FOR THE FIRST OPERATION

THE FOLLOWING PARAMETERS ARE USED FOR THE INITIAL OPERATION (IN
ADDITION TO THE 'MINIMUM' ADDRESS VALUES).

LOC	TAG	INITIAL VALUE	VALUE RANGE	FUNCTION
---	---	-----	-----	-----
1514	BEGPAT	10	1 - 15	THE CODE FOR THE STARTING PATTERN. (IF A WRITE ORDER OR A WRITE CHECK ORDER IS SPECIFIED IN 'BEGCOD')
1516	BEGCOD	5	0 - 5	THE INITIAL COMMAND FOR EACH DRIVE EXERCISED. 0 = WRITE CHECK DATA 1 = WRITE CHECK HEADER & DATA 2 = WRITE DATA 3 = WRITE HEADER & DATA 4 = READ DATA 5 = READ HEADER & DATA
1520	BEGSIZ	402	2 - SIZE	THE BUFFER SIZE FOR THE FIRST DATA TRANSFER OPERATION.

4.3 SWITCH REGISTER SETTINGS

SW <15> = 1 HALT ON ERROR
SW <13> = 1 INHIBIT ERROR TYPEOUT
SW <10> = 1 RING THE TELETYPE BELL IF ERROR
SW <7> = 1 DISPLAY ALL DATA COMPARE ERRORS
SW <6> = 1 DO NOT ALTER THE CURRENT OPERATION PARAMETERS
SW <5> = 1 PARTIAL REGISTER DISPLAY IF ERROR; DO NOT DISPLAY
ECC CORRECTION RESULTS
SW <4> = 1 INHIBIT MAXIMUM ERROR COUNT CHECK; DO NOT DEASSIGN
DRIVES WHEN NORMAL END OF TEST REACHED.
SW <3> = 1 DISPLAY THE SECTOR IN ERROR (BEFORE RETRY ATTEMPTS)
IF 'DCK', 'DTE', OR 'WCF' ERRORS OR AFTER THE 28TH
RETRY IF 'UNCORRECTABLE' 'DCK' ERROR.
IF DATA COMPARE ERRORS & SW<7> SET, DISPLAY REST
OF BUFFER
SW <2> = 1 INHIBIT SUBSYSTEM STATUS TYPEOUT DURING STARTUP.
INHIBIT PERFORMANCE SUMMARY TYPEOUTS.
SW <1> = 1 INHIBIT DATA COMPASION AFTER READ ORDERS
SW <0> = 1 READ ONLY MODE

IF THE PROGRAM IS BEING RUN ON A SWITCHLESS PROCESSOR (I.E. AN 11/34)
THE PROGRAM WILL DETERMINE THAT THE HARDWARE SWITCH REGISTER IS
NOT PRESENT AND WILL USE A 'SOFTWARE' SWITCH REGISTER. THE
'SOFTWARE' SWITCH REGISTER IS LOCATED AT LOCATION 176 (8). THE
SETTINGS OF THE 'SOFTWARE' SWITCHES ARE CONTROLLED THROUGH A KEYBOARD
ROUTINE WHICH IS CALLED BY TYPING A 'CONTROL G'. THE PROGRAM WILL
RECOGNIZE THE 'CONTROL G' AT ANY TIME EXCEPT WHEN THE PROGRAM IS
IN KEYBOARD ENTRY MODE, OR IS AT A HIGHER PRIORITY PROCESSING AN
RM03 INTERRUPT. THE 'SOFTWARE' SWITCH VALUES ARE ENTERED
AS AN OCTAL NUMBER IN RESPONSE TO THE PROMPT FROM THE SWITCH
ENTRY ROUTINE:

'SWR = NNNNNN NEW ='

EACH TIME SWITCH SETTINGS ARE ENTERED, THE ENTIRE SWITCH REGISTER

IMAGE MUST BE ENTERED. LEADING ZEROS ARE NOT REQUIRED. 'RUBOUT' AND 'CONTROL U' FUNCTIONS MAY BE USED TO CORRECT TYPING ERRORS DURING SWITCH ENTRY.

ON PROCESSORS WITH HARDWARE SWITCH REGISTERS, THE 'SOFTWARE' SWITCH REGISTER MAY BE USED. IF THE PROGRAM FINDS ALL 16 SWITCHES IN THE 'UP' POSITION, ALL SWITCH REGISTER REFERENCES WILL BE TO THE 'SOFTWARE' REGISTER AND THE PROCEDURES DESCRIBED ABOVE MUST BE FOLLOWED.

4.4 KEYBOARD COMMANDS

NOTE: ALL KEYBOARD COMMANDS WILL BE DISABLED IF DIAGNOSTIC IS RUNNING IN THE APT SCRIPT MODE.

THROUGH THE KEYBOARD COMMANDS, THE OPERATOR MAY ASSIGN DRIVES FOR TEST ('T' COMMAND), WRITE AND CHECK DATA PACKS ('W' COMMAND), PERFORM WRITE DATA AND FOLLOWED BY TEST ('WT' COMMAND), PERFORM A SEQUENTIAL READ OF A PACK ('R' COMMAND), REQUEST A DRIVE PERFORMANCE SUMMARY ('S' COMMAND), OR DEASSIGN A DRIVE WHICH IS BEING TESTED, READING, OR WRITING ('D' COMMAND).

AFTER THE PROGRAM HAS BEEN INITIALIZED, THE FOLLOWING MESSAGE WILL BE TYPED:

'PROGRAM INITIALIZATION COMPLETE
'TYPE A CONTROL C TO ENTER COMMANDS'

KEYBOARD ENTRIES WILL NOT BE RECOGNIZED UNTIL THE OPERATOR TYPES A 'CONTROL C'. WHEN THE PROGRAM SEES A 'CONTROL C' ENTRY, IT WILL SUSPEND THE SCHEDULING OF FURTHER DEVICE OPERATIONS AND WAIT UNTIL ALL OUTSTANDING ORDERS HAVE TERMINATED. THE PROGRAM WILL ENTER COMMAND ENTRY MODE AND TYPE THE FOLLOWING PROMPTING MESSAGE:

'HH:MM:SS
'ENTER COMMANDS:'

THE PROGRAM WILL THEN ACCEPT ANY OF THE VALID COMMANDS. AT THE COMPLETION OF A COMMAND, THE PROGRAM WILL EXIT COMMAND MODE; THE OPERATOR MUST TYPE ANOTHER 'CONTROL C' TO RETURN THE PROGRAM TO COMMAND MODE. IF THE COMMAND ENTERED SPECIFIED AN 'A' DRIVE NUMBER, THE PROGRAM WILL REMAIN IN COMMAND MODE UNTIL ALL AVAILABLE DRIVES HAVE BEEN PROCESSED.

THE 'WT', 'T', 'W', AND 'R' COMMANDS REQUIRE ADDRESS LIMITS, DRIVE I.D. AND BAD LOCATION ADDRESS ENTRIES FOR THE DRIVE BEING REFERENCED.

THE PROGRAM WILL FIRST ASK FOR ADDRESS LIMITS WITH THE FOLLOWING TYPEOUT:

'ENTER ADDRESS LIMITS FOR DRV #N / RM03

THE PROGRAM WILL REQUEST VALUES FOR THE FOLLOWING ADDRESS LIMIT PARAMETERS.

DEFAULT VALUE

NAME	BASE	VALUE	RANGE	FUNCTION
MINCYL	10	0	0 - 822	THE MINIMUM CYLINDER ADDRESS
MAXCYL	10	822	0 - 822	THE MAXIMUM CYLINDER ADDRESS
MINTRK	10	0	0 - 4	THE MINIMUM TRACK ADDRESS
MAXTRK	10	4	0 - 4	THE MAXIMUM TRACK ADDRESS
MINSEC	10	0	0 - 31	THE MINIMUM SECTOR ADDRESS
MAXSEC	10	31	0 - 31	THE MAXIMUM SECTOR ADDRESS

NOTE: 1. THE ADDRESS LIMITS ARE IN DECIMAL

2. THE MINIMUM CYLINDER, TRACK, OR SECTOR ADDRESS MAY BE SPECIFIED AS BEING LARGER THAN THE MAXIMUM ADDRESS. WHEN THESE VALUES ARE INVERTED, THE PROGRAM WILL SELECT ADDRESSES BETWEEN THE 'MIN' ADDRESS AND THE UPPER PHYSICAL LIMIT FOR THAT ADDRESS AND BETWEEN '0' AND THE VALUE IN 'MAX'.

EACH COMMAND, EXCEPT THE 'S' AND THE 'D' COMMANDS, WILL ASK FOR A DRIVE IDENTIFICATION ENTRY. THE DRIVE IDENTIFICATION ENTRY REQUEST IS MADE BY THE FOLLOWING MESSAGE:

'ENTER I.D. FOR DRV #N:'

THE OPERATOR MAY ENTER AN I.D. NUMBER FOR THE DRIVE OF UP TO 6 CHARACTERS IN LENGTH. THIS I.D. WILL BE DISPLAYED, ALONG WITH THE DATE AND OPERATOR I.D. ENTRIES (SEE SECTION 4.1), WHEN THE 'SA' COMMAND IS EXECUTED. THE OPERATOR MAY ENTER ANY CHARACTER STRING, TERMINATED BY A 'CARRIAGE RETURN', OR A 'PERIOD' ONLY (NULL ENTRY) IN RESPONSE TO THE I.D. REQUEST.

4.4.1

'T' COMMAND
USED TO ASSIGN A DRIVE(S) FOR TEST. THIS COMMAND IS REQUIRED TO PERFORM THE TEST OF THE DRIVE(S). THE OTHER COMMANDS ARE CONVIENCE COMMANDS OR SUPPORT COMMANDS.

FORMAT: TN<CR>

N = DRIVE NUMBER. MAY BE 0 TO 7 OR 'A'. ENTRY MUST BE TERMINIATED BY A CARRIAGE RETURN <CR>.

EXAMPLE: TO<CR> - ASSIGN DRIVE 0 FOR TEST
TA<CR> - ASSIGN ALL AVAILABLE DRIVES FOR TEST

NOTE: DRIVE OPERATION BEGINS IMMEDIATELY AFTER COMMAND IS ENTERED.

4.4.2 'D' COMMAND

USED TO DEASSIGN A DRIVE(S) BEING EXERCISED.

FORMAT: DN<CR>

705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760

N = DRIVE NUMBER. MAY BE 0 TO 7 OR 'A'. ENTRY MUST BE TERMINATED BY A CARRIAGE RETURN <CR>.

EXAMPLE: DO<CR> - DEASSIGN DRIVE 0
DA<CR> - DEASSIGN ALL DRIVES BEING TESTED.

- NOTES:
1. IF THE 'D' COMMAND REFERENCES A DRIVE NOT ASSIGNED THE PROGRAM WILL TYPEOUT '?DRIVE NOT ASSIGNED'
 2. THE DRIVES WILL BE DEASSIGNED AS THEIR OPERATIONS COMPLETE.
 3. IF 'DA' IS USED, ALL DRIVES BEING TESTED WILL BE DEASSIGNED; THE MESSAGE IN (1) WILL BE DISPLAYED FOR ALL DRIVES NOT BEING TESTED.

4.4.3 'S' COMMAND

USED TO REQUEST A PERFORMANCE SUMMARY TYPEOUT FOR THE REFERENCED DRIVE(S).

FORMAT: SN<CR>

N = DRIVE NUMBER. MAY BE 0 TO 7 OR 'A'. ENTRY MUST BE TERMINATED BY A CARRIAGE RETURN <CR>.

EXAMPLE: SO<CR> - TYPEOUT PERFORMANCE SUMMARY FOR DRIVE 0
SA<CR> - TYPEOUT PERFORMANCE SUMMARY FOR ALL DRIVES BEING TESTED.

- NOTES:
1. IF 'SA' IS USED ONLY DRIVES BEING TESTED WILL BE DISPLAYED.
 2. IF PARAMETER 'MINUTE' IS NOT ZERO, THE PROGRAM WILL AUTOMATICALLY DISPLAY A PERFORMANCE SUMMARY FOR EACH DRIVE BEING TESTED AT A RATE DETERMINED BY 'MINUTE'.
 3. IF THE 'SA' COMMAND IS USED, THE PROGRAM WILL TYPEOUT THE OPERATOR ENTERED DATE, OPERATOR I.D., AND THE DRIVE I.D. FOR EACH DRIVE BEING TESTED. THE DATE AND OPERATOR I.D. WILL NOT BE TYPED OUT IF NO DRIVES ARE BEING TESTED.

4.4.4 'W' COMMAND

USED TO WRITE A DATA PACK WITH DATA ACCEPTABLE TO THE RM03 PERFORMANCE EXERCISER PROGRAM.

FORMAT: WN<CR>

N = DRIVE NUMBER. MAY BE 0 TO 7 OR 'A'. ENTRY MUST BE TERMINATED BY A CARRIAGE RETURN <CR>.

761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816

EXAMPLE: WA<CR> - WRITE DATA PACKS ON ALL AVAILABLE DRIVES.
WO<CR> - GENERATE A DATA PACK ON DRIVE 0.

- NOTES:
1. DATA PATTERNS GENERATED BY THE RMO3 FORMATTER PROGRAM (CZRMACO) ARE COMPATIBLE.
 2. THE 'W' COMMAND MUST BE USED TO WRITE A NEW PACK UNDER TEST BEFORE OTHER COMMANDS ARE ISSUED; IF THE DISK PACK HAS BEEN WRITTEN BY ANY PROGRAM OTHER THAN THE FORMATTER.
 3. THE 'W' COMMAND PERFORMS A SEQUENTIAL WRITE OF THE PACK USING A 'WRITE DATA' COMMAND. THE DATA PATTERN USED FOR EACH WRITE IS SELECTED RANDOMLY. HOWEVER, THE OPERATION OF THE COMMAND IS SEQUENTIAL, BEGINNING AT 'MINCYL', 'MINTRK' AND CONTINUING TO 'MAXCYL', 'MAXTRK'.
 4. THE 'W' COMMAND DOES NOT WRITE HEADERS AND ASSUMES THAT THE FORMAT OF THE PACK IS GOOD.
 5. THE 'W' COMMAND CANNOT BE STARTED IF SWITCH 0 (READ ONLY MODE) IS SET. IF SWITCH 0 IS SET DURING THE OPERATION OF THE 'W' COMMAND, THE DRIVE PERFORMING THE 'W' COMMAND WILL IGNORE THE SWITCH.

4.4.5 'R' COMMAND

USED TO PERFORM A SEQUENTIAL READ OF THE PACK.

FORMAT: RN<CR>

N = DRIVE NUMBER. MAY BE 0 TO 7 OR 'A'. ENTRY MUST BE TERMINATED BY A CARRIAGE RETURN <CR>.

EXAMPLE: RA<CR> - READ THE PACKS ON ALL OF THE ONLINE DRIVES.
RO<CR> - READ THE PACK ON DRIVE 0.

- NOTES:
1. THE PROGRAM WILL PERFORM A NORMAL CHECK OF ALL DATA READ. HOWEVER, ALL OPERATIONS WILL BE SEQUENTIAL.
 2. THE PROGRAM WILL READ THE PACK STARTING AT THE ADDRESS SPECIFIED BY 'MINCYL', 'MINTRK' TO THE ADDRESS SPECIFIED BY 'MAXCYL', 'MAXTRK'. THE READ WILL BE SEQUENTIAL.

4.4.6 'WT' COMMAND

USED TO PERFORM A SEQUENTIAL WRITE PACK AND FOLLOWED BY A "TEST" COMMAND

FORMAT: WTN<CR>

817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872

873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909

N= DRIVE NUMBER 0 TO 7 OR "A". ENTRY MUST BE TERMINATED BY A
CARRIAGE RETURN <CR>.

EXAMPLE: WTA<CR> -WRITE ALL PACKS AND TEST ALL PACKS
WTO<CR>-WRITE PACK 0 AND TEST PACK 0

4.4.7 GENERAL COMMAND INFORMATION

- A. CONTROL-C MUST BE ENTERED BEFORE ISSUING ANY COMMAND.
- B. T,R,W AND WT COMMANDS ARE EXCLUSIVE TO ONE ANOTHER
ON THE SAME DRIVE UNDER TESTING.
D COMMAND MUST BE ENTERED IN ORDER TO SWITCH COMMANDS
AMONG T,R,W AND WT .
- C. S COMMAND CAN BE ENTERED ANY TIME DURING THE TEST.
- D. THE ERROR RESPONSES FROM THE PROGRAM ARE AS FOLLOWS

RESPONSE	COMMAND(S)
-----	-----
?UNIT N OFFLINE	T, W, R,WT
?UNIT N NOT ASSIGNED	D, S
?UNIT N ALREADY ASSIGNED	T, W, R,WT
?UNIT N NOT PRESENT	T, W, R,WT
?UNIT N UNSAFE	T, W, R,WT
?UNIT N NOT AN RM03	T, W, R,WT

5. PERFORMANCE SUMMARY TYPEOUT

5.1 THE PROGRAM WILL DISPLAY A PERFORMANCE SUMMARY FOR THE DRIVES
BEING EXERCISED. THIS SUMMARY WILL BE DISPLAYED AUTOMATICALLY
IF THE PARAMETER 'INTRVL' IS NOT ZERO OR CAN BE DISPLAYED ON
REQUEST BY THE OPERATOR THROUGH THE USE OF THE 'S' COMMAND.
THE PERFORMANCE SUMMARY TYPEOUT CONTAINS THE FOLLOWING FIELDS:

'DRV'	THE DRIVE NUMBER
'PASS'	THE PRESENT PASS COUNT FOR THE DRIVE
'ORDERS'	THE NUMBER OF ORDERS PERFORMED BY THE DRIVE
'SEEKS'	THE NUMBER OF SEEK OPERATIONS THE DRIVE PERFORMED
'WRDS XFER'	THE TOTAL NUMBER OF WORDS WRITTEN AND READ BY THE DRIVE
'WRDS READ'	THE TOTAL NUMBER OF WORDS READ BY THE DRIVE
'SOFT'	THE NUMBER OF SOFT DATA ERRORS
'HARD'	THE NUMBER OF HARD DATA ERRORS
'SKI'	THE NUMBER OF 'SKI' ERRORS
'MISP'	THE NUMBER OF POSITIONING ERRORS
'OTHER'	THE TOTAL ERRORS OF OTHER TYPES

NOTE: ERRORS EM1, EM2, EM3, EM4, EM5, & EM10 ARE NOT INCLUDED IN
THE 'OTHER' ERROR TOTAL.

5.2 SOFT/HARD ERROR DEFINITIONS

5.2.1 HARD ERRORS

A. A 'DTE' (DRIVE TIMING ERROR) OR A 'DCK' (DATA CHECK ERROR)
WHICH OCCURS DURING A READ DATA OR A READ HEADER & DATA OPERATION
AND IS NOT CORRECTABLE OR DOES NOT BECOME CORRECTABLE AFTER THE PROGRAM
HAS PERFORMED THE COMPLETE RETRY SEQUENCE ON THE BAD SECTOR.
THE RETRY SEQUENCE IS 16 RE-READS AT TRACK CENTER AND 2 ATTEMPTS
BOTH AT POSITIVE AND NEGATIVE OFFSETS:

5.2.2 SOFT ERRORS

A. ECC CORRECTABLE 'DCK' ERRORS.
B. 'DCK' & 'ECH' ERRORS WHICH BECOME ECC CORRECTABLE DURING
RETRY OR WHICH ARE READ CORRECTLY DURING RETRY.
C. HEADER READ ERRORS - READ DATA, READ HEADER & DATA, OR
WRITE DATA ORDERS.
D. 'DTE' ERRORS WHICH ARE CORRECTED OR WHICH BECOME ECC CORRECTABLE
'DCK' ERROR DURING THE RETRY SEQUENCE.

6. DATA CHECKING & ERROR RECOVERY

6.1 DATA COMPARISON

DATA COMPARISON OCCURS AFTER EACH 'RDDAT' (READ DATA) OR 'RDHD' (READ HEADER AND DATA) OPERATION UNDER THE FOLLOWING CONDITIONS:

- A. THE ORDER TERMINATED WITH NO ERROR.
- B. THE OPERATION TERMINATED WITH 'DCK' SET AND THE ERROR IS ECC CORRECTABLE OR THE SECTOR IN ERROR IS READ CORRECTLY AFTER RETRY ATTEMPTS.

6.2 VERIFICATION OF DATA WRITTEN

DATA VERIFICATION IS DONE EITHER THROUGH READING THE WRITTEN DATA BACK AND MATCHING THE DATA WITH ONE OF THE 15 PATTERNS OR THROUGH ISSUING WRITE CHECK COMMANDS.

6.3 SECTOR REFORMATTING

THE PROGRAM WILL REFORMAT AN UNCORRECTABLE ERROR SECTOR IN THE FOLLOWING CASES (PARAMETER 'FORMAT' MUST BE SET AND SW<00> = 0). THIS PREVENTS THE SAME ERROR FROM BEING CONTINUOUSLY REPORTED.

- A. DATA CHECK ERRORS - EM21
- B. HEADER READ ERRORS - EM20, EM24, EM25, EM26, EM27
- C. DRIVE TIMING ERRORS - EM31
- D. OPERATION INCOMPLETE ERRORS - EM32
- E. WRITE CHECK ERRORS - EM22, EM23

6.4 BAD TRACK/SECTOR FLAGGING

SINCE THE RM03 SUBSYSTEM HAS AN AUTOMATIC BAD TRACK HANDLING CAPABILITY, THE MULTIDRIVE EXERCISER ALLOWS TO IDENTIFY UP TO 16 BAD TRACK/SECTOR LOCATIONS WHEN THE DRIVE IS ASSIGNED FOR TEST. (SEE SECTION 4.4 FOR ADDITIONAL INFORMATION.)

IF ONE OF THE FOLLOWING ERRORS OCCURS AT A LOCATION IDENTIFIED BY THE OPERATOR, THE PROGRAM WILL INHIBIT THE ERROR REPORT FOR THAT ERROR.

DATA CHECK ERRORS ('DCK')
WRITE CHECK ERRORS ('WCE')
OPERATION INCOMPLETE ERRORS ('OPI')
DRIVE TIMING ERRORS ('DTE')
HEADER READ ERRORS ('FER' & 'HCRC', 'HCE' & 'HCRC', 'HCRC')

OPTIONALLY, THE OPERATOR MAY REQUEST AN ERROR REPORT FOR FLAGGED AREAS. (PARAMETER 'NOTPRT' MUST BE SET TO 0 AT STARTUP.) THE PROGRAM WILL IDENTIFY ERROR MESSAGES ASSOCIATED WITH THE PACK; THESE ERRORS WILL NOT BE ADDED TO THE ERROR TOTALS MAINTAINED BY THE PROGRAM.

7. ERROR MESSAGES

DRIVE ERRORS ARE REPORTED ON THE TELETYPE OR (IF AVAILABLE) A LINE PRINTER. ALL ERROR CONDITIONS ARE REPORTED IN ERROR MESSAGES; THE PROGRAM CONTAINS NO CODED ERROR HALTS. IF THE PROGRAM HALTS (ASSUMING, OF COURSE, THAT SW(15) IS NOT SET), AN UNRECOVERABLE PROGRAM CONDITION HAS OCCURRED OR A CENTRAL PROCESSOR FAILURE HAS OCCURRED.

ERROR MESSAGES ARE MADE UP OF SEVERAL LINES. EACH TYPE OF ERROR HAS SEVERAL OPTIONAL LINES WHICH MAY APPEAR WITH IT. ALL OF THE POSSIBLE ERROR MESSAGE LINES WHICH MAY APPEAR ARE GIVEN IN THE SECTION DESCRIBING THE PARTICULAR ERROR HEADER.

7.1 ERROR DESCRIPTION LINES

MESSAGES EM1, EM2, EM3, EM4, EM5, EM10, EM11, & EM12 ARE ALWAYS DISPLAYED ON THE TTY. THE OTHER MESSAGES ARE DISPLAYED ON EITHER THE LINE PRINTER (IF AVAILABLE) OR THE TTY.

(THE MESSAGE TAGS ARE GIVEN FOR REFERENCE.)

MESSAGE

TAG	TEXT
---	----
EM1	RH11 INTERRUPT OCCURRED (RMAS=0)
	THE RH11 INTERRUPTED AND THE ATTENTION SUMMARY REGISTER (RMAS) WAS CLEARED.
EM2	UNEXPECTED ATTENTION OCCURRED
	THE INDICATED DRIVE INTERRUPTED BUT THE DRIVE WAS NOT PERFORMING AN OPERATION.
EM3	MASSBUS PARITY ERROR (MCPE=1)
	THE RH11 DETECTED A CONTROL BUS PARITY ERROR WHEN READING THE INDICATED REGISTER FROM THE INDICATED DRIVE.
EM4	MASSBUS PARITY ERROR (PAR=1)
	THE INDICATED RM03 DETECTED A CONTROL BUS PARITY ERROR WHEN THE RH11 LOADED THE SPECIFIED REGISTER.
EM5	ADDRESS PLUG CHANGE BIT SET
	THE 'OPE' BIT WAS SET WHEN THE INDICATED DRIVE INTERRUPTED.
EM6	RH11 DIDN'T RESPOND TO ADDRESSING
	WHEN THE PROGRAM ADDRESSED THE RH11, NO RESPONSE WAS RECEIVED FROM THE INDICATED ADDRESS.

1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067

1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123

EM10 UNCORRECTABLE MASSBUS PARITY ERROR
THE PROGRAM HAS TRIED 3 TIMES TO READ OR WRITE THE INDICATED REGISTER.

EM11 FATAL MASSBUS PARITY ERROR
A CONTROL BUS PARITY ERROR OCCURRED WHEN THE RH11 ATTEMPTED TO PROCESS A PREVIOUS, DIFFERENT PARITY ERROR.

EM12 PERSISTENT DEVICE UNSAFE
THE DRIVE BECAME UNSAFE; DRIVE CLEAR TO THE DRIVE DID NOT CLEAR THE UNSAFE CONDITION. THE PROGRAM WILL AUTOMATICALLY DEASSIGN THE DRIVE. THE DRIVE CANNOT BE EXERCISED UNTIL THE UNSAFE CONDITION HAS BEEN CLEARED BY MANUAL INTERVENTION.

EM13 OPERATION NOT COMPLETED WITHIN TIME LIMIT
THE DRIVE DID NOT COMPLETE THE OPERATION WITHIN 1 SECOND AFTER THE OPERATION WAS INITIATED.

EM14 UNIT WENT OFFLINE
THE DRIVE WENT OFFLINE DURING THE INDICATED OPERATION. (THE 'MOL' BIT BECAME ZERO.) THE PROGRAM WILL AUTOMATICALLY DEASSIGN THE DRIVE. THE OPERATOR MUST REASSIGN THE DRIVE WITH THE 'T' COMMAND TO RE-INITIATE TESTING.

EM15 NO RESPONSE TO PORT REQUEST
THE PROGRAM IS TESTING A DUAL PORT DRIVE WHICH HAS NOT SWITCHED TO THE REQUESTING PORT WITHIN 10 SECONDS AFTER PORT REQUEST TO THE DRIVE FROM THE REPORTING PORT.

EM20 HEADER CRC ERROR
A HEADER CRC ERROR WAS DETECTED AT THE INDICATED DISK ADDRESS. THE CONTENTS OF THE HEADER ARE DISPLAYED. THE OPERATION WILL BE RETRIED 3 TIMES.

EM21 DATA CHECK ('DCK') ERROR
A DATA CHECK ERROR WAS DETECTED AT THE INDICATED SECTOR. THE FULL RETRY SEQUENCE (INCLUDING OFFSET) WILL BE INITIATED FOR THE SECTOR IN ERROR IF THE ECC HARD ERROR ('ECH') BIT IS SET.

EM22 WRITE CHECK ERROR - DATA CHECK ('DCK') SET
A WRITE CHECK ERROR OCCURRED AND THE DATA CHECK ('DCK') BIT WAS SET. IF 'ECH' IS NOT SET, THE OPERATION WILL BE RETRIED UP TO 3 TIMES; IF THE 'ECH' BIT IS SET, THE OPERATION WILL BE RETRIED UP TO 16 TIMES.

1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179

EM23 WRITE CHECK ERROR - DATA CHECK ('DCK') NOT SET
A WRITE CHECK ERROR OCCURRED AND 'DCK' WAS NOT SET. THE WORDS WHICH CAUSED THE ERROR ARE DISPLAYED IN THE ERROR MESSAGE. THE OPERATION WILL BE RETRIED 3 TIMES.

EM24 HEADER READ ERROR - 'FMT' BIT DROPPED
A WRITE DATA, WRITE CHECK DATA, OR A READ DATA WAS BEING PERFORMED AND A 'FMT' ERROR OCCURRED. THE PROGRAM RE-READ THE HEADER OF THE ERROR SECTOR AND THE 'HCRC' BIT WAS SET. THE CONTENTS OF THE HEADER ARE DISPLAYED. THE OPERATION WILL BE RETRIED 3 TIMES.

EM25 HEADER READ ERROR - HEADER COMPARE ('HCE') ERROR
SIMILAR TO EM24, EXCEPT THAT THE 'HCE' ERROR BIT WAS SET INITIALLY. THE OPERATION WILL BE RETRIED 3 TIMES.

EM26 FORMAT ERROR ('FER')
FORMAT ERROR OCCURRED. WHEN THE HEADER WAS RE-READ, THE 'HCRC' BIT WAS NOT SET. THE CONTENTS OF THE HEADER ARE DISPLAYED. THE OPERATION WILL BE RETRIED 3 TIMES.

EM27 HEADER COMPARE ('HCE') ERROR
SIMILAR TO EM26 EXCEPT THAT THE 'HCE' BIT WAS SET INITIALLY. THE OPERATION WILL BE RETRIED 3 TIMES.

EM30 MISCELLANEOUS DRIVE ERROR
THIS MESSAGE IS GIVEN FOR THE FOLLOWING ERROR BITS: 'AOE', 'RMR', 'ILF', OR 'ILR'

EM31 OPERATION INCOMPLETE ('OPI') ERROR
AN OPERATION INCOMPLETE ERROR OCCURRED AT THE INDICATED SECTOR.

EM32 DRIVE TIMING ('DTE') ERROR
DRIVE TIMING ERROR OCCURRED ON THE INDICATED SECTOR. THE OPERATION WILL BE RETRIED 3 TIMES.

EM33 PARITY ('PAR') ERROR AFTER OPERATION STARTED
THE 'PAR' BIT WAS SET WHEN THE OPERATION WAS COMPLETED. THE OPERATION WILL BE RETRIED 3 TIMES.

EM34 WRITE CLOCK FAILURE ('WCF')
A WRITE CLOCK FAILURE OCCURRED DURING THE OPERATION. THE OPERATION WILL BE RETRIED 3 TIMES.

EM35 INVALID ADDRESS ('IAE') ERROR

1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235

AN INVALID ADDRESS ERROR OCCURRED DURING THE OPERATION.

EM36 WRITE LOCK ('WLE') ERROR

A WRITE OPERATION WAS ATTEMPTED BUT THE DRIVE WAS WRITE LOCKED.

EM40 RH11 OR UNIBUS TRANSFER ERROR

'TRE' IS SET IN THE RH11 CONTROL REGISTER AND NO DRIVE ERROR HAS OCCURRED. THE OPERATION WILL BE RETRIED 3 TIMES IF THE ERROR WAS CAUSED BY 'DLT', 'UPE', 'MXF', OR 'MDPE'.

EM41 BUS ADDRESS OR WORD COUNT INCORRECT

NO DRIVE ERROR OCCURRED BUT EITHER THE BUS ADDRESS INDICATES THAT AN INCORRECT NUMBER OF WORDS WERE TRANSFERED OR THE WORD COUNT REGISTER IS NOT ZERO.

EM42 DATA COMPARE ERRORS - NO DRIVE ERROR DETECTED

NO SUBSYSTEM ERROR WAS SIGNALLED; HOWEVER, THE DATA DOES NOT COMPARE.

EM43 CAN'T MATCH DATA READ WITH A PATTERN

THE DATA IN THE BUFFER DOES NOT MATCH ANY OF THE STANDARD PATTERNS.

EM44 ERROR BIT(S) SET, BUT NO ERROR SIGNALLED BY THE RH11

THE OPERATION COMPLETED NORMALLY; HOWEVER, THE PROGRAM FOUND EITHER ERROR BITS IN THE RM03 SET OR ERROR BITS IN THE RH11 SET.

EM45 ECC LOGIC FAILURE

THE CONTENTS OF EITHER THE ECC POSITION REGISTER (RMEC1) OR THE CONTENTS OF ECC PATTERN REGISTER (RMEC2) ARE NOT VALID. THE POSITION REGISTER IS EITHER '0' OR > 40066 (8) OR THE PATTERN REGISTER CONTAINS ZEROS.

EM46 BUS ADDRESS OR WORD COUNT NOT CONSISTENT

THE PROGRAM WAS PROCESSING AN ERROR AND FOUND THAT THE NUMBER OF WORDS TRANSFERED AS INDICATED BY THE BUS ADDRESS REGISTER DOES NOT AGREE WITH THE TRANSFER COUNT FROM THE WORD COUNT REGISTER.

EM50 SEEK INCOMPLETE ERROR

THE DRIVE SIGNALLED EITHER 'SKI' OR 'OCYL' ERROR BITS.

EM51 NOT USED

EM60 DEVICE UNSAFE

THE INDICATED DRIVE UNSAFE ERROR OCCURRED; THE ERROR WAS
CLEARED BY A 'DRIVE CLEAR' INSTRUCTION.

7.2 DETAIL ERROR LINES

THE LINE NUMBERS GIVEN BELOW ARE FOR REFERENCE ONLY.

LINE 1

TT:TT:TT (DESCRIPTION OF ERROR)

'TT:TT:TT' IS THE TIME SINCE THE PROGRAM
WAS STARTED. TT:TT:TT IS GIVEN IN HOURS:
MINUTES: SECONDS.

LINE 2

'PRESENT ORDER = XXXX PREVIOUS ORDER = YYYY'

MNEMONICS USED FOR THE ORDERS ARE DEFINED BELOW:

UNLOAD - UNLOAD (OCTAL 3)
 SEEK - SEEK (OCTAL 5)
 RECAL - RECALIBRATE (OCTAL 7)
 DRVCLR - DRIVE CLEAR (OCTAL 11)
 RELSE - RELEASE (OCTAL 13)
 OFFSET - OFFSET (OCTAL 15)
 RTC - RETURN TO CENTERLINE (OCTAL 17)
 READIN - READIN PRESET (OCTAL 21)
 PACK - PACK ACKNOWLEDGE (OCTAL 23)
 SEARCH - SEARCH (OCTAL 31)
 GETREG - GET REGISTERS (OCTAL 41)
 SETFMT - SET FORMAT (ECI OR HCI) (OCTAL 43)
 SELDRV - SELECT DRIVE (OCTAL 45)
 WCKD - WRITE CHECK DATA (OCTAL 51)
 WCKHD - WRITE CHECK HEADER & DATA (OCTAL 53)
 WRDAT - WRITE DATA (OCTAL 61)
 WRTHD - WRITE CHECK HEADER & DATA (OCTAL 63)
 RDDAT - READ DATA (OCTAL 71)
 RDHD - READ HEADER & DATA (OCTAL 73)

(DISPLAY OF THE RH11/RM03 REGISTERS IN TWO GROUPS:
 RMCS1, RMCS2, RMDS1, RMER1, RMER2, RMER3, RMEC1, & RMEC2 FORM THE FIRST
 GROUP; ALL THE OTHER REGISTERS ARE IN THE SECOND GROUP.
 IF SW(05) IS SET, ONLY THE REGISTERS IN THE FIRST GROUP WILL BE
 DISPLAYED.)

THE ABOVE LINE WILL BE TYPED IF THE ERROR OCCURRED DURING

THE NON-DATA TRANSFER PART OF THE OPERATION.

'* ERROR AT BAD TRACK/SECTOR'

THE ABOVE LINE WILL BE PRINTED IF A DATA ERROR OCCURES AT AN ADDRESS ON THE PACK WHICH THE OPERATOR HAS IDENTIFIED AS BEING BAD. PARAMETER 'NOTPRT' MUST BE 0 FOR THE ERROR TO BE REPORTED.

A WORD CALLED 'STATUS' IS DISPLAYED WITH THE RMO3 REGISTERS. THE CONTENTS OF THIS WORD IDENTIFY HOW THE ERROR WAS PROCESSED BY THE RMO3 DRIVE HANDLER ROUTINE. THE BITS IN THIS WORD ARE ENCODED AS FOLLOWS:

BIT #	MEANING IF BIT IS '1'
-----	-----
15	ERROR OCCURRED DONE (BIT07=0), BITS 14-9, 2, 1 SPECIFY TYPE DONE (BIT07=1), BITS 6-3 SPECIFY TYPE
14	DRIVE IS OFFLINE
12	PERSISTENT UNSAFE CONDITION EXISTS
11	UNCORRECTABLE PARITY ERROR OCCURRED
10	FATAL PARITY ERROR OCCURRED. MASSBUS CLEAR WAS PERFORMED
9	OPERATION NOT COMPLETED WITHIN 1 SECOND MASSBUS CLEAR PERFORMED. ALL OTHER OUTSTANDING OPERATIONS WERE RESTARTED.
7	DONE - OPERATION COMPLETED
6	DATA ERROR OCCURRED DURING THE TRANSFER
5	ERROR OCCURRED WHILE SEARCHING FOR THE 'TRANSFER' SECTOR OR DURING RECALIBRATE OR OFFSET COMMANDS
4	CORRECTABLE UNSAFE CONDITION OCCURRED
3	DRIVE ERROR OCCURRED THAT CAUSED AN AUTOMATIC RECALIBRATE SEQUENCE
2	PORT REQUEST TIMEOUT
1	NON-EXISTENT DRIVE REQUESTED

LINE 3

ERROR AT CXXX TYY SZZ PREV ADDR = CUUU TVV SWW

THE ACTUAL ADDRESS OF THE ERROR SECTOR AND THE PREVIOUS
DISK ADDRESS ARE GIVEN IN THIS LINE. CYLINDER, TRACK, &

1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347

1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403

SECTOR ADDRESSES ARE IN DECIMAL.

LINE 4

PRESENT ADDR = CXXX TYY SZZ PREV ADDR = CUUU TVV SWW

THIS LINE IDENTIFIES THE ADDRESS WHEN THE ERROR WAS DETECTED;
THE PREVIOUS ADDRESS IS ALSO GIVEN. CYLINDER, TRACK, & SECTOR
ADDRESSES ARE GIVEN IN DECIMAL.

LINE 5

START CYL = XXX END CYL = YYY

THIS LINE IDENTIFIES THE STARTING CYLINDER OR A SEEK (IMPLIED)
AND THE DESTINATION CYLINDER. CYLINDER ADDRESSES ARE IN
DECIMAL.

LINE 6

START CYL = XXX END CYL = YYY ACTUAL CYL = ZZZ

THIS LINE IDENTIFIES THE STARTING CYLINDER OF AN IMPLIED SEEK,
THE DESTINATION CYLINDER, AND THE CYLINDER THE DISK ACTUALLY
STOPPED AT. CYLINDER ADDRESSES ARE IN DECIMAL.

LINE 7

RMBA = XXXX RMWC = YYYY

THIS LINE GIVES THE CONTENTS OF THE RH11 BUFFER ADDRESS
REGISTER AND THE RH11 WORD COUNT REGISTER. THIS LINE IS
NOT PRINTED IF SW(05) IS NOT SET.

LINE 8

START CYL = XXX START TRK = YY START SECTOR = ZZ

THIS LINE IDENTIFIES THE STARTING DISK ADDRESS OF THE PRESENT
OPERATION. CYLINDER, TRACK, AND SECTOR VALUES ARE DECIMAL.

LINE 9

RMDA = XXXX RMCA = YYYY

THIS LINE GIVES THE CONTENTS OF THE RM03 TRACK AND SECTOR
ADDRESS REGISTER AND THE CONTENTS OF THE DESIRED CYLINDER
ADDRESS REGISTER. THIS LINE IS NOT PRINTED IF SW(05) IS NOT
SET.

1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459

LINE 10

BUFFER ADDR = XXXX SIZE = YYYY ACTUAL NUMBR WRDS XFRD = ZZZZ

THIS LINE GIVES THE STARTING ADDRESS OF THE BUFFER USED FOR THE
CURRENT DATA TRANSFER OPERATION, ITS SIZE, AND THE ACTUAL NUMBER
OF WORD TRANSFERED. THE STARTING ADDRESS OF THE BUFFER IS IN
OCTAL, THE SIZE AND WORD TRANSFERED VALUE ARE IN DECIMAL.

LINE 11

GOOD DATA = XXXX BAD DATA = YYYY SECT POS = ZZZ

THIS LINE GIVES THE GOOD DATA, THE ACTUAL DATA FROM THE DISK,
AND THE LOCATION IN THE SECTOR OF THE ACTUAL DATA. THE SECTOR
POSITION IS IN DECIMAL.

LINE 12

HEADER CONTENTS OF ERROR SECTOR = XXXX XXXX XXXX XXXX

THIS LINE GIVES THE CONTENTS OF THE HEADER OF THE SECTOR WHICH
GAVE THE ERROR.

LINE 13

RMEC1 = XXXX RMEC2 = YYYY

THIS LINE WILL BE PRINTED AFTER A SUCESSFUL RETRY OF A SECTOR
WHICH BECAME ECC CORRECTABLE DURING RETRY.

LINE 14

ECC CORRECTABLE WITHOUT OFFSET

THE SECTOR IN ERROR IS ECC CORRECTABLE; NO RETRY ATTEMPTS ARE
NECESSARY.

LINE 15

READ CORRECTLY AT (NEG OR POS) OFFSET

THE SECTOR IN ERROR WAS READ WITHOUT ERROR AT THE INDICATED
OFFSET VALUE.

LINE 16

ECC CORRECTABLE AT (NEG OR POS) OFFSET

1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515

THE SECTOR IN ERROR BECAME ECC CORRECTABLE AT THE INDICATED
OFFSET.

LINE 17

CORRECTED ON X RETRY

THE OPERATION WAS PERFORMED ERROR FREE ON THE INDICATED RETRY
ATTEMPT.

LINE 18

UNCORRECTABLE AFTER X RETRIES

THE OPERATION COULD NOT BE PERFORMED CORRECTLY AFTER THE
INDICATED NUMBER OF RETRY ATTEMPTS.

LINE 19

DIFFERENT ERROR DURING RETRY

WHILE THE PROGRAM WAS RETRYING THE ERROR, A DIFFERENT OCCURRED.
IF THIS LINE IS PRINTED, THE RH11/RM03 REGISTERS WILL ALSO BE
PRINTED (SEE LINE 2).

LINE 20

DATA COMPARISON ERRORS

A PRINTOUT OF THE DATA COMPARISON ERRORS FOLLOW THIS LINE.

LINE 21

TOTAL COMPARE ERRORS = XXXX

THIS LINE GIVES THE TOTAL DATA COMPARISON ERROR COUNT. THE
VALUE GIVEN IS IN DECIMAL.

LINE 22

THE DATA COMPARED OK

THIS LINE INDICATES THE RESULTS OF THE DATA COMPARISON FOLLOWING
ECC CORRECTION.

LINE 23

ECC CORRECTION RESULTS

THE PROGRAM PERFORMED ECC CORRECTION AND THE RESULTS ARE REPORTED.
THE ADDRESS IN MEMORY OF THE WORD(S) IN ERROR ARE GIVEN, THE WORD(S)
BEFORE CORRECTION AND THE WORD(S) AFTER CORRECTION ARE PRINTED.

LINE 24

ERROR BURST BEGINS AT WORD XXX IN DATA FIELD OF ERROR SECTOR

THIS IS AN INFORMATIONAL LINE WHICH WILL BE PRINTED FOR 'DCK' ERRORS
WHICH ARE ECC CORRECTABLE OR WHICH BECOME ECC CORRECTABLE DURING
RETRY. 'XXX' IS THE WORD OFFSET VALUE FROM 'RMEC1' AND IS IN
DECIMAL.

LINE 25

ERROR WAS NOT IN THE DATA READ -
ECC CORRECTION CAN'T BE PERFORMED

THE DATA ERROR WAS NOT IN DATA TRANSFERED TO MEMORY.

LINE 26

CONTENTS OF THE ERROR SECTOR (REPORTED ABOVE)

IF SW(03) IS SET, THE SECTOR WHICH GAVE THE 'DCK', 'DTE' OR,
'WCF' ERROR OR 'HARD' DATA CHECK ERROR IS PRINTED. THE
CONTENTS OF THE SECTOR FOLLOW THIS LINE.

LINE 27

ORDERS: WWW ERRORS: X WRDS XFR: YYYY WRDS READ: ZZZZ

THIS IS THE LAST LINE PRINTED FOR ALL NON-POSITIONING
TYPE ERRORS.

'ORDERS' IS THE TOTAL NUMBER OF COMMANDS GIVEN TO THE DRIVE
WHICH REPORTED THE ERROR.

'ERRORS' IS THE TOTAL ERROR COUNT FOR THE DRIVE AND INCLUDES
EVERY ERROR DETECTED, REGARDLESS OF TYPE.

'WRDS XFR' IS THE TOTAL NUMBER OF WORDS WRITTEN AND READ BY
THE DRIVE.

'WRDS READ' IS THE TOTAL NUMBER OF WORD READ BY THE DRIVE.

LINE 28

ORDERS: WWW TOTAL SEEKS: XXX TOTAL POS ERR = YYY TOTAL SKI ERR = Z

THIS IS THE LAST LINE PRINTED FOR ALL POSITIONING TYPE ERRORS.

CZRM880 RM03/2 PERF EXER
CZRM88.P11 21-NOV-77 15:34

MACY11 30(1046) 22-NOV-77 10:06 PAGE 32

SEQ 0031

1572
1573
1574
1575
1576
1577
1578
1579
1580

'ORDERS' IS THE TOTAL NUMBER OF ORDERS GIVEN TO THE DRIVE WHICH
REPORTED THE ERROR.

'TOTAL SEEKS' IS THE TOTAL NUMBER OF SEEK OPERATIONS PERFORMED
BY THE DRIVE.

'TOTAL SKI ERR' IS THE TOTAL NUMBER OF 'SKI' ERRORS
SIGNALLED BY THE DRIVE.

8. PROGRAM DESCRIPTION

8.1 PROGRAM OPERATION

WHEN THE PROGRAM IS STARTED, PROVIDING APT TTY ENABLE BIT IS SET OR DIAGNOSTIC LOADED BY OTHER THAN APT SCRIPT MODE, ALL TABLES AND PARAMETERS ARE CLEARED OR INITIALIZED. THE PARAMETERS WHICH ARE UNDER OPERATOR TTY ENTRY CONTROL ARE CHECKED FOR VALIDITY AND CONSISTENCY. RH11 INTERRUPT ENABLE ('IE') IS SET, TTY KEYBOARD INTERRUPT ENABLE IS SET, AND THE KW11-L OR KW11-P CLOCK IS STARTED. WHEN THESE ACTIONS HAVE BEEN COMPLETED, THE PROGRAM TYPES OUT 'PROGRAM INITIALIZE COMPLETE'. COMMAND ENTRIES WILL NOW BE ACCEPTED BY THE PROGRAM

THE PROGRAM SCANS ITS INTERNAL ASSIGNMENT TABLES, LOOKING FOR:

- 1) DRIVES TO ASSIGN/DEASSIGN
- 2) PERFORMANCE SUMMARY TYPEOUT REQUESTS
- 3) DRIVES REQUIRING COMMAND INITIATION, BUFFER ASSIGNMENT, OR PARAMETER SELECTION.
- 4) DRIVES COMPLETING CURRENT OPERATIONS.

THE PROGRAM CONTINUES SCANNING ITS TABLES UNTIL AN ENTRY IS FOUND. IN THE CASE OF THE PROGRAM AT INITIAL START, THE FIRST ENTRY WILL BE MADE BY THE OPERATOR WHEN A DRIVE IS ASSIGNED ('T' COMMAND).

WHEN A DRIVE IS ASSIGNED, THE KEYBOARD ENTRY ROUTINE VERIFIES THAT THE DRIVE IS PRESENT, IS AN RM03, AND IS ONLINE. THE ASSIGNMENT ROUTINE THEN ISSUES A 'READIN PRESET' INSTRUCTION, SETS 'FMT22', AND ISSUES A 'RECALIBRATE' INSTRUCTION.

PARAMETERS FOR THE OPERATION ARE SELECTED AND A BUFFER IS ASSIGNED. IF THE OPERATION IS A WRITE OR WRITE CHECK ORDER, THE ASSIGNED BUFFER WILL BE FILLED WITH THE SELECTED PATTERN. (WRITE CHECK ORDERS ARE ISSUED AFTER EACH WRITE ORDER. THE WRITE CHECK ORDER USES THE PARAMETERS SELECTED FOR THE PRECEDING WRITE ORDER.) CONTROL IS THEN PASSED TO THE COMMAND INITIATION ROUTINE.

THE COMMAND INITIATION ROUTINE FIRST LOOKS AT THE CYLINDER ADDRESS OF THE REQUESTED OPERATION. IF THE DRIVE MUST SEEK TO ANOTHER CYLINDER TO PERFORM THE OPERATION, THE PROGRAM ISSUES A SEARCH INSTRUCTION TO THE DRIVE WITH A 'TARGET' SECTOR WHICH IS 8 SECTORS EARLIER THAN THE 'TRANSFER' SECTOR. (THIS ALLOWS THE PROGRAM TO INITIATE OPERATIONS ON ANOTHER DRIVE WHILE THE PRESENT DRIVE, OR OTHER DRIVES, ARE SEARCHING FOR 'TARGET' SECTORS. ALL SEEKS ISSUED BY THE PROGRAM ARE IMPLIED SEEK SEARCH OPERATIONS.) WHEN A SEARCHING DRIVE FINDS THE 'TARGET' SECTOR AND INTERRUPTS, THE PROGRAM READS THE LOOK AHEAD REGISTER (RLA) OF THE INTERRUPTING DRIVE AND COMPARES THE POSITION OF THE DISK WITH THAT OF THE DESIRED SECTOR.

IF OTHER DRIVES ARE WAITING ON CYLINDER, THEY ARE ALSO CHECKED. THE PROGRAM THEN ISSUES THE REQUESTED ORDER TO THE DRIVE NEAREST ITS TRANSFER SECTOR. THE DRIVES NOT SELECTED WILL HAVE ANOTHER SEARCH INITIATED. IF A DRIVE IS NOT SELECTED FOR TRANSFER AFTER THREE REVOLUTIONS OF ITS DISK, IT IS GIVEN PRIORITY OVER DRIVES WHICH HAVE

1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636

NOT BEEN ON CYLINDER AS LONG.

WHEN THE DATA TRANSFER OPERATION IS COMPLETE, THE DRIVE REGISTERS ARE STORED AND A DATA TRANSFER IS INITIATED FOR A WAITING DRIVE.

IF THE OPERATION HAS BEEN COMPLETED NORMALLY, THE SAVED DRIVE REGISTERS ARE CHECKED TO VERIFY THAT NO ERROR BITS ARE SET; THE RH11 BUS ADDRESS AND WORD COUNT ADDRESS REGISTERS ARE CHECKED TO VERIFY THAT THE CORRECT NUMBER OF WORDS HAVE BEEN TRANSFERED AND THAT THE TWO REGISTERS ARE CONSISTENT WITH EACH OTHER; AND IF THE ORDER WAS A READ ORDER, THE DATA BUFFER IS COMPARED. WHEN THIS SEQUENCE IS COMPLETED, THE DRIVE IS RETURNED TO THE ASSIGNED, INACTIVE LIST. THE PROGRAM THEN INITIATES A DATA TRANSFER ON A WAITING DRIVE AND RESELECTS AND REINITIATES ANOTHER OPERATION ON THE RELEASED DRIVE.

ERRORS WHICH OCCUR ARE PROCESSED IN THE FOLLOWING ORDER. MULTIPLE ERRORS WILL BE REPORTED AS THE FIRST ERROR TYPE CHECKED.

A. ERRORS REPORTED FOR OPERATIONS WHICH HAVE NOT COMPLETED NORMALLY.

PERSISTENT UNSAFE CONDITION - EM12
UNCORRECTABLE MASSBUS PARITY ERROR - EM10
FATAL MASSBUS PARITY ERROR - EM11
OPERATION NOT COMPLETED WITHIN TIME LIMIT - EM13
UNIT WENT OFFLINE - EM14

B. ERRORS REPORTED FOR OPERATIONS WHICH COMPLETE NORMALLY.

CORRECTABLE UNSAFE - EM60
DRIVE 'MING' ERROR - EM32
DATA CHECK ERROR - EM21
WRITE CHECK WITH DCK SET - EM22
HEADER CRC ERRORS - EM20
FORMAT ERRORS - EM24, EM26
HEADER COMPARE ERRORS - EM25, EM27
PROGRAM DETECTED POSITIONING ERROR - EM51
SEEK INCOMPLETE OR OFF CYLINDER ERROR - EM50
WRITE CHECK WITHOUT 'DCK' SET - EM23
RH11 OR UNIBUS TRANSFER ERROR - EM40
'OPI' ERROR - EM31
'PAR' ERROR - EM33
'WCF' ERROR - EM34
'IAE' ERROR - EM35
'WLE' ERROR - EM36
MISCELLANEOUS DRIVE ERROR - EM30

C. ERRORS NOT FLAGGED BY THE HARDWARE ERROR DETECTION LOGIC.

BUS ADDRESS OR WORD COUNT INCORRECT - EM41
DATA COMPARE ERRORS - NO DRIVE ERROR DETECTED - EM42
CAN'T MATCH DATA READ WITH A PATTERN - EM43
ERROR BIT(S) SET, BUT NO ERROR SIGNALLED BY THE RH11 - EM44
ECC LOGIC FAILURE - EM45
BUS ADDRESS OR WORD COUNT NOT CONSISTENT - EM46

B.2 DUAL PORT OPERATION

1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692

1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748

DUAL PORT OPERATION IS NEARLY IDENTICAL TO THE OPERATION DESCRIBED IN SECTION 8.1. THE DIFFERENCES ARE IN COMMAND SEQUENCE INITIATION AND ORDER TERMINATION.

WHEN THE DUAL PORT HANDLER ROUTINE IN THE EXERCISER PROGRAM RECEIVES A REQUEST FOR A DRIVE, THE PROGRAM VERIFIES THAT THE DRIVE IS ONLINE. THE DRIVE IS SELECTED AND READ THE RMCS1 REGISTER BY TEST THE "DVA" BIT.

IF THE DRIVE IS IN NEUTRAL, THIS WILL SEIZE THE DRIVE. IF THE DRIVE IS SEIZED BY THE OTHER PORT, READING 'RMDS1' WILL SET 'PORT REQUEST'. THE PROGRAM CHECKS 'DVA' IN 'RMCS1'. IF THE DRIVE IS AVAILABLE AS INDICATED BY THE 'DVA' BIT, THE COMMAND SEQUENCE WILL BE INITIATED IN THE NORMAL MANNER (SEE SECTION 8.1 ABOVE). IF 'DVA' WAS NOT SET, THE PROGRAM MAKES AN ENTRY FOR THE DRIVE IN AN INTERNAL 'PORT REQUEST PENDING' TABLE AND STARTS A 10 SECOND TIMER FOR THE DRIVE. IF THE DRIVE HAS NOT SWITCHED TO THE REQUESTING SYSTEM WITHIN THE 10 SECOND INTERVAL, THE PROGRAM REPORTS A 'NO RESPONSE TO PORT REQUEST' ERROR. NORMALLY THIS ERROR MESSAGE INDICATES A FAILURE IN THE DUAL PORT CONTROL LOGIC IN THE DRIVE BEING TESTED; HOWEVER, UNDER CERTAIN CONDITIONS (E.G. MASSBUS PARITY ERRORS BEING REPORTED ON THE OTHER SYSTEM ON A MOD33 TTY), THE OTHER PROCESSOR WAS UNABLE TO PROCESS THE DRIVE AFTER IT HAD REQUESTED THE DRIVE. THE OPERATOR MUST BE AWARE OF WHAT THE OTHER SYSTEM IS DOING AT ALL TIMES TO INTERPRET THE PORT RELATED ERROR MESSAGES PROPERLY.

AFTER A DRIVE HAS COMPLETED AN OPERATION, THE PROGRAM WILL STORE THE REGISTERS AND ISSUE A 'RELEASE' TO THE DRIVE. IF THE OPERATION TERMINATED WITH AN ERROR, THE DRIVE WILL NOT BE RELEASED UNTIL ERROR PROCESSING HAS BEEN COMPLETED.

SINGLE PORT DRIVES, DRIVES WHICH ARE IN NEUTRAL BUT NOT BEING EXERCISED BY THE OPPOSITE PORT ARE STILL TREATED AS DUAL PORT DRIVES IN THAT A RELEASE COMMAND IS ISSUED AT THE END OF NORMAL ORDER PROCESSING OR AT THE END OF ERROR PROCESSING. A RELEASE COMMAND ISSUED UNDER THESE CONDITIONS HAS NO FUNCTIONAL EFFECT ON THE OPERATION OF THE DRIVE.

8.3 SELECTION OF OPERATION VARIABLES

A. SECTOR ADDRESS SELECTION IS RANDOM BETWEEN THE VALUES IN 'MINSEC' AND 'MAXSEC'. TRACK ADDRESS SELECTION IS RANDOM BETWEEN THE VALUES IN 'MINTRK' AND 'MAXTRK'. CYLINDER ADDRESS SELECTION IS RANDOM BETWEEN 'MINCYL' AND 'MAXCYL'. IF A MINIMUM ADDRESS IS GREATER THAN THE CORRESPONDING MAXIMUM ADDRESS, THE PROGRAM WILL EXCLUDE ALL ADDRESSES BETWEEN 'MAX' AND 'MIN' FROM THE SELECTION. FOR EXAMPLE: IF 'MINTRK' IS 5 AND 'MAXTRK' IS 2, THEN TRACK ADDRESS SELECTION WILL EXCLUDE TRACKS 3 - 4 FROM THE SELECTION AND SELECT AN ADDRESS FROM AMONG ADDRESSES 5, 0, 1, 2.

B. THE BUFFER SIZE IS RANDOM SELECTED BETWEEN 4 - AND THE VALUE IN 'SIZE'. THE SIZE SELECTED IS WEIGHTED TO ENSURE THIS IS NECESSARY AS THE PROGRAM REQUIRES 4 LOCATIONS IN THE DATA PORTION OF THE SECTOR TO BE ABLE TO MATCH THE DATA

TO A PATTERN FOR DATA COMPARISON PURPOSES.

- C. THE DATA WRITTEN IS RANDOMLY SELECTED AMONG THE 15 STANDARD PATTERNS. THE PARAMETER "PATTERN" ENABLES THE RANDOM PATTERN SELECTION, IF THIS PARAMETER IS 0; OTHERWISE, THE DATA PATTERN INDEXED BY THE VALUE "PATTERN" IS SELECTED.
- D. THE ORDERS ARE SELECTED RANDOMLY. WRITE CHECK DATA AND WRITE CHECK HEADER & DATA ORDERS ARE PERFORMED ONLY IF THE PREVIOUS ORDER WAS THE APPROPRIATE DATA ORDER. IF THE 'FORMAT' PARAMETER IS ZERO, THE PROGRAM WILL NOT SELECT WRITE HEADER & DATA (AND WRITE CHECK HEADER & DATA) ORDERS. WHEN THE PROGRAM SELECTS A WRITE HEADER & DATA ORDER, THE BUFFER SIZE IS FORCED TO 258 (10); THE PROGRAM WILL NOT PERFORM A MULTI-SECTOR FORMAT WRITE OPERATION.
- E. THE FIRST ORDER PERFORMED AFTER A UNIT IS ASSIGNED WITH A 'W' OR 'R' COMMAND IS NOT RANDOMLY SELECTED. THE PARAMETERS FOR THE FIRST OPERATION ARE THE MINIMUM OR STARTING VALUES OF THE VARIABLES.

8.4 DATA PATTERNS

THE PROGRAM SELECTS ONE OF THE FOLLOWING DATA PATTERNS TO WRITE WHEN A WRITE ORDER IS SELECTED. THE ENTIRE BUFFER IS FILLED WITH THE SELECTED PATTERN. WHEN DATA IS READ FROM THE DISK, THE PROGRAM COMPARES DATA ON A SECTOR BASIS: FROM THE FIRST 4 DATA WORDS OF EACH SECTOR, THE PROGRAM MATCHES THE DATA TO ONE OF THE FOLLOWING PATTERNS.

PAT 1	PAT 2	PAT 3	PAT 4	PAT 5	PAT 6	PAT 7	PAT 8
-----	-----	-----	-----	-----	-----	-----	-----
000001	177776	000000	133331	052525	155554	026455	066666
000003	177774	000000	133331	052525	155554	026455	066666
000007	177770	000000	133331	052525	155554	026455	066666
000017	177760	177777	133331	125252	155554	151322	066666
000037	177740	177777	133331	125252	155554	151322	066666
000077	177700	177777	133331	125252	155554	151322	066666
000177	177600	000000	133331	052525	155554	026455	066666
000377	177400	000000	133331	052525	155554	026455	066666
000777	177000	177777	133331	125252	155554	151322	066666
001777	176000	177777	133331	125252	155554	151322	066666
003777	174000	000000	133331	052525	155554	026455	066666
007777	170000	177777	133331	125252	155554	151322	066666
017777	160000	000000	133331	052525	155554	026455	066666
037777	140000	177777	133331	125252	155554	151322	066666
077777	100000	000000	133331	052525	155554	026455	066666
177777	000000	177777	133331	125252	155554	151322	066666
PAT 9	PAT 10	PAT 11	PAT 12	PAT 13	PAT 14	PAT 15	
-----	-----	-----	-----	-----	-----	-----	
000001	177776	172666	077777	153333	000000	177777	
000002	177775	155555	137777	066667	177777	000000	
000004	177773	172666	157777	153333	177777	000000	
000010	177767	155555	167777	066667	177777	000000	

K03

CZRM880 RM03/2 PERF EXER
CZRM88.P11 21-NOV-77 15:34

MACY11 30(1046) 22-NOV-77 10:06 PAGE 37

SEQ 0036

1805	000020	177757	172666	173777	153333	177777	000000
1806	000040	177737	155555	175777	066667	177777	000000
1807	000100	177677	172666	176777	153333	177777	000000
1808	000200	177577	155555	177377	066667	177777	000000
1809	000400	177377	172666	177577	153333	177777	000000
1810	001000	176777	155555	177677	066667	177777	000000
1811	002000	175777	172666	177737	153333	177777	000000
1812	004000	173777	155555	177757	066667	177777	000000
1813	010000	167777	172666	177767	153333	177777	000000
1814	020000	157777	155555	177773	066667	177777	000000
1815	040000	137777	172666	177775	153333	177777	000000
1816	100000	077777	155555	177776	066667	177777	000000

9.1 RH70(11)/RM03 DRIVER

THIS DOCUMENT IS THE USER'S GUIDE FOR THE RH70/RM03 DRIVER.

9.2 TO INITIALIZE THE DRIVER:

```
JSR    PC,RMINIT
RETURN
```

UPON RETURN YOU MUST EXAMINE THE "DRVSTA" TABLE TO DETERMINE THE DRIVES THAT ARE ONLINE FOR TESTING. THE 'DRVSTA' TABLE IS EIGHT BYTES, ONE BYTE PER DRIVE. THE STATE OF EACH DRIVE WILL BE INDICATED AS FOLLOWS:

DRVSTA	DRIVE STATE
-----	-----
>0	ONLINE RM03
=0	OFFLINE RM03, DRIVE IS NOT AN RM03, OR NONEXISTENT DRIVE
<0	UNSAFE RM03

THE DRIVE TYPE IS DEFINED IN AN 8 BYTE LONG TABLE TAGGED 'DRVTYP'. THE TABLE CONTAINS ONE BYTE FOR EACH DRIVE AND IS INDEXED BY THE DRIVE NUMBER. ENTRIES ARE ENCODED AS FOLLOWS:

DRVTYP	CONDITION
-----	-----
0	NONEXISTENT DRIVE
4	RM03
-1	NOT AN RM03

THE 'RMINIT' ROUTINE WILL DO A READIN PRESET AND WILL SET FMT22.

9.3 AFTER THE DRIVER HAS BEEN INITIALIZED, IT IS CALLED USING THE FOLLOWING SEQUENCE.

```
CALL: JSR    R0,RM03      ;MAKE THE CALL
      PNTDPB             ;ADDRESS OF DPB*
      RETURN1            ;RETURN IF QUEUE IS FULL
      RETURN2            ;RETURN IF REQUEST IS IN
                        ;QUEUE OR THERE IS AN
                        ;ERROR CONDITION
```

*DPB (DATA PARAMETER BLOCK)

PNTDPB: .BYTE	0	;(0) DRIVE NUMBER
.BYTE	0	;(1) OFFSET VALUE OR FMT22, ECT. AND HCI
.BYTE	0	;(2) COMMAND
.BYTE	0	;(3) PSEL AND A17 AND A16
.WORD	0	;(4) WORD COUNT (MUST BE NEG.)


```
1873 .WORD 0 ;(6) BUFFER ADDRESS OR
1874 .BYTE 0 ;REGISTER TABLE POINTER
1875 .BYTE 0 ;(10) SECTOR ADDRESS OR
1876 .BYTE 0 ;FIRST REG. INDEX
1877 .WORD 0 ;(11) TRACK ADDRESS OR
1878 .WORD 0 ;LAST REG. INDEX
1879 .WORD 0 ;(12) CYLINDER ADDRESS
1880 .WORD 0 ;(14) ERROR TABLE POINTER
1881 ;POINTS TO THE FIRST OF TWENTY
1882 ;LOCATIONS OF WHERE THE DRIVER
1883 ;IS TO STORE THE RM70/RM03
1884 ;REGISTERS ON AN ERROR. IF LEFT
1885 ;ZERO REGISTERS ARE NOT SAVED.
1886 .WORD 0 ;(16) STATUS/ERROR INDICATOR
1887 ;BIT15=1=>ERROR OCCURRED
1888 ;BIT07=1=>DONE
1889 ;BIT14-BIT09 AND BIT06-BIT03
1890 ;INDICATE TYPE OF ERROR
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
```

9.4 THE DRIVER PROVIDES A SOFTWARE TIMEOUT CAPABILITY.
TO UTILIZE THIS CAPABILITY YOU MUST SUPPLY THE "RM TIMER" ROUTINE
WITH THE ELAPSED TIME IN THE FOLLOWING MANNER:

```
MOV #16.,-(SP) ;16 MILLISECONDS BETWEEN
JSR PC,RMTMR ;CLOCK TICKS
;CALL THE TIMER ROUTINE
```

IT SHOULD BE NOTED THAT YOU MUST PROVIDE THE CODE TO DRIVE THE
CLOCK. AND THE ELAPSED TIME MUST BE IN MILLISECONDS.
THE DRIVER WILL SET THE TIMEOUT TO 1 SECOND FOR ALL POSITIONING
AND DATA TRANSFER OPERATIONS AND WILL SET THE TIMEOUT TO 30
SECONDS FOR ERROR RECOVERY OPERATIONS.

9.4.1 EXAMPLE - WRITE 1000. WORDS

```
1$: JSR R0,RM03 ;CALL THE DRIVER
WRTDPB ;DPB ADDRESS
BR 1$ ;WAIT FOR QUEUE IF FULL
2$: TST WRTDPB+16 ;WAIT FOR COMMAND TO COMPLETE
BEQ 2$
BMI ERROR1 ;ERROR OCCURRED
.
.
.
WRTDPB: .BYTE 5 ;DRIVE #5
.BYTE 0 ;
.BYTE 161 ;WRITE COMMAND
.BYTE 0 ;
.WORD -1000. ;WORD COUNT
.WORD WRTBUF ;BUFFER ADDRESS
.BYTE 3 ;SECTOR
.BYTE 5 ;TRACK
.WORD 400 ;CYLINDER
.WORD EPRTBS ;ERROR TABLE
```

.WORD 0 ;STATUS/ERROR INDICATOR

ALTERNATE DPB SETUP

WRTPB: .WORD 5 ;THIS SETUP ACHIEVED
.WORD WRITE ;EVERYTHING THE
.WORD -1000. ;ABOVE TABLE DID, BUT
.WORD WRTBUF ;IN A CLEANER FORMAT
.BYTE 3,5
.WORD 400,ERRTB5,0

9.5 RH70/RM03 REGISTERS

MNEMONIC	INDEX
-----	-----
RMCS1	0
RMWC	2
RMBA	4
RMDA	6
RMCS2	10
RMD5	12
RMR1	14
RMA5	16
RMLA	20
RMOB	22
RMMR1	24
RMDT	26
RMSN	30
RMOF	32
RMDC	34
RMHR	36
RMMR2	40
RMR2	42
RMEC1	44
RMEC2	46

9.6 COMMANDS PERFORMED BY THE DRIVER

COMMAND	CODE	COMMAND TYPE
-----	-----	-----
NO OPERATION	101	N
UNLOAD	103	N
SEEK	105	P
RECALIRATE	107	P
DRIVE CLEAR	111	N
RELEASE	113	N
OFFSET	115	P

1985	RETURN TO CENTER	117	P
1986			
1987	READIN PRESET	121	N
1988			
1989	PACK ACKNOWLEDGE	123	N
1990			
1991	SEARCH	131	P
1992			
1993	GET REGISTER(S)	141	S
1994			
1995	SET FORMAT	143	S
1996			
1997	SELECT DRIVE	145	S
1998			
1999	WRITE CHECK DATA	151	D
2000			
2001	WRITE CHECK HEADER	153	D
2002	AND DATA		
2003			
2004	WRITE DATA	161	D
2005			
2006	WRITE HEADER & DATA	163	D
2007			
2008	READ DATA	171	D
2009			
2010	READ HEADER & DATA	173	D
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			
2037			
2038			
2039			
2040			

N = HOUSEKEEPING
P = POSITIONING
D = DATA TRANSFER
S = SPECIAL PROVIDED BY THE DRIVER

9.7 DPB STATUS/ERROR INDICATOR WORD

THIS INDICATOR WILL INFORM THE USER OF THE RESULTS OF THE REQUEST.
THIS IS ACCOMPLISHED BY SETTING VARIOUS BITS OF THE INDICATOR TO
A ONE.

<u>BIT NO.</u>	<u>MEANING IF ON A "1"</u>
15	ERROR OCCURRED DONE (BIT07=0); BITS 14-10 SPECIFIES TYPE DONE (BIT07=1); BITS 06-03 SPECIFIES TYPE
14(1)	USER MADE A REQUEST FOR A FUNCTION TO BE PERFORMED ON AN OFFLINE OR UNSAFE DRIVE
13(1)	USER MADE A REQUEST FOR A FUNCTION TO BE PERFORMED ON A DRIVE THAT HAS AN UNLOAD REQUEST IN QUEUE.
12(2)	PERSISTENT UNSAFE CONDITION EXIST.
11(2)	UNCORRECTABLE PARITY ERROR OCCURRED

2041	10(2)(4)	FATAL PARITY ERROR. A MASSBUS CLEAR WAS PERFORMED, ALL QUEUES WERE EMPTIED, AND ALL DRVACT'S SET TO THE IDLE STATE
2042		
2043		
2044		
2045	9(3)(4)	SOFTWARE TIMEOUT OCCURRED ON THIS DRIVE
2046	8(4)	SOFTWARE TIMEOUT OCCURRED ON ANOTHER DRIVE
2047	7	DONE
2048		
2049		
2050		
2051	6(2)	ERROR OCCURRED DURING AN I/O OPERATION
2052		
2053	5(2)	ERROR OCCURRED DURING AN OPERATION OTHER THAN I/O.
2054		
2055		
2056	4(2)	CORRECTABLE UNSAFE CONDITION OCCURRED
2057		
2058	3(2)	DRIVE ERROR OCCURRED THAT CAUSED AN AUTOMATIC "RECALIBRATE" SEQUENCE
2059		
2060	2	PORT REQUEST TIMEOUT. THE DRIVER REQUESTED THE DRIVE BUT THE OPPOSITE PORT DID NOT RELEASE THE DRIVE WITHIN 20 SECONDS.
2061		
2062	1	NON-EXISTENT DRIVE REQUESTED. USER MADE A REQUEST FOR A NON-EXISTENT DRIVE.
2063		
2064	(1) =>	REQUEST WASN'T PUT IN QUEUE. (RH70/RM03 REGISTERS WERE NOT SAVED)
2065		
2066	(2) =>	REQUEST QUEUE HAS BEEN EMPTIED. THE DRIVER ISSUED A "DRIVE CLEAR" TO THE DRIVE. NOTE: ALL RH70/RM03 REGISTERS ARE SAVED AS PER DPB+14 BEFORE THE "DRIVE CLEAR".
2067		
2068	(3) =>	REQUEST QUEUE HAS BEEN EMPTIED. THE DRIVER ISSUED A MASSBUS INIT. ALL RH70/RM03 REGISTERS FOR THE DRIVE WERE SAVED AS PER DPB+14 BEFORE THE INIT.
2069		
2070	(4) =>	A "RECALIBRATE" SHOULD BE ISSUED BEFORE ANY OTHER COMMAND.
2071		
2072		
2073		
2074		
2075		
2076		
2077		
2078		
2079		
2080		
2081		
2082		
2083		
2084		
2085	9.8	ERROR CALLS MADE BY THE DRIVER.
2086		THERE ARE A FEW ERRORS THAT CAN OCCUR THAT CAN NOT BE INDICATED IN A DPB.
2087		WHEN THIS TYPE OF ERROR IS DETECTED BY THE DRIVER IT WILL MAKE AN ERROR CALL OF THE FORM "ERROR N", WHERE "N" IS THE ERROR NUMBER AND THE ERROR WILL BE AN EMT INSTRUCTION.
2088		
2089		
2090		
2091		
2092		
2093		
2094		
2095		
2096		

N	TYPE	DATA AVAILABLE
-	----	-----

2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127

1	RM70 INTERRUPT OCCURRED (RMAS=0)	*R4= RMCS1'S ADDRESS
2	UNEXPECTED ATTENTION OCCURRED	R1= DRIVE NUMBER R3= ATA BIT *R4= RMCS1'S ADDRESS R5= (RMAS) RMERRS =RMD5 RMERRS+2=RMER1 RMERRS+4=RMER2 RMERRS+6=RMMR2
3	MASSBUS PARITY ERROR (MCPE=1)	RD.ADR= ADDRESS OF REG. READ RD.WRD= WORD READ
4	MASSBUS PARITY ERROR (PAR=1)	WRT.AD= ADDRESS OF REG. WRITTEN WRT.WD= WORD WRITTEN RD.WRD= WORD READ BACK
5	ADDRESS PLUG CHANGE BIT SET ('OPE' ERROR)	R1= DRIVE NUMBER R3= ATA BIT *R4= RMCS1'S ADDRESS R5= (RMAS) RMERRS =RMD5 RMERRS+2=RMER1 RMERRS+4=RMER2 RMERRS+6=RMMR2
	* THIS IS THE ACTUAL UNIBUS ADDRESS (176700)	
2		

```
2128
2129
2130 .TITLE CZRM880 RM03/2 PERF EXER
2131 .*COPYRIGHT (C) 1977
2132 .*DIGITAL EQUIPMENT CORP.
2133 .*MAYNARD, MASS. 01754
2134
2135 .*PROGRAM BY C. CHEN
2136
2137 .*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
2138 .*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
2139
2140 .SBTTL OPERATIONAL SWITCH SETTINGS
2141
2142 .*
2143 .*      SWITCH      USE
2144 .*      -----
2145 .*      15          HALT ON ERROR
2146 .*      13          INHIBIT ERROR TIMEOUTS
2147 .*      10          BELL ON ERROR
2148 .*      7           DISPLAY ALL DATA COMPARE ERRORS
2149 .*      6           DON'T CHANGE PARAMETERS (LOOP ON PRESENT VALUES)
2150 .*      5           A. PARTIAL REGISTER DISPLAY IF ERROR
2151 .*                   B. NO ECC CORRECTION RESULTS DISPLAYED IF ERROR
2152 .*      4           A. DO NOT CHECK FOR MAXIMUM ERROR COUNTS
2153 .*                   B. DO NOT DROP DRIVE WHEN NORMAL END OF TEST REACHED
2154 .*      3           A. DISPLAY ERROR SECTOR IF 'DCK', 'DTE', OR 'WCF' ERROR
2155 .*                   B. DISPLAY SECTOR IF 'DCK' ERR UNCORRECTABLE AFTER
2156 .*                     28TH RETRY
2157 .*      2           C. IF DATA COMPARE ERROR & SW7 SET, DISPLAY
2158 .*                     REMAINDER OF BUFFER
2159 .*      1           A. DON'T TYPE SUBSYSTEM STATUS WHEN PROGRAM STARTED
2160 .*                   B. DON'T TYPE PERFORMANCE SUMMARY
2161 .*      0           INHIBIT DATA COMPARISON AFTER READ ORDERS
2162 .*                   READ ONLY MODE
2163 .SBTTL BASIC DEFINITIONS
2164
2165 .*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
2166 STACK= 1100
2167
2168 .EQUIV EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
2169 .EQUIV IOT,SCOPE      ;;BASIC DEFINITION OF SCOPE CALL
2170
2171 .*MISCELLANEOUS DEFINITIONS
2172
2173 HT= 11              ;;CODE FOR HORIZONTAL TAB
2174 LF= 12              ;;CODE FOR LINE FEED
2175 CR= 15              ;;CODE FOR CARRIAGE RETURN
2176 CRLF= 200           ;;CODE FOR CARRIAGE RETURN-LINE FEED
2177 PS= 177776          ;;PROCESSOR STATUS WORD
2178 .EQUIV PS,PSW
2179 STKLMT= 177774       ;;STACK LIMIT REGISTER
2180 PIRQ= 177772         ;;PROGRAM INTERRUPT REQUEST REGISTER
2181 DSWR= 177570         ;;HARDWARE SWITCH REGISTER
2182 ODISP= 177570        ;;HARDWARE DISPLAY REGISTER
2183
2184 .*GENERAL PURPOSE REGISTER DEFINITIONS
2185 R0= %0              ;;GENERAL REGISTER
2186 R1= %1              ;;GENERAL REGISTER
2187 R2= %2              ;;GENERAL REGISTER
```


2184	000003	R3=	%3	:: GENERAL REGISTER
2185	000004	R4=	%4	:: GENERAL REGISTER
2186	000005	R5=	%5	:: GENERAL REGISTER
2187	000006	R6=	%6	:: GENERAL REGISTER
2188	000007	R7=	%7	:: GENERAL REGISTER
2189	000006	SP=	%6	:: STACK POINTER
2190	000007	PC=	%7	:: PROGRAM COUNTER
2191				
2192		: *PRIORITY LEVEL DEFINITIONS		
2193	000000	PR0=	0	:: PRIORITY LEVEL 0
2194	000040	PR1=	40	:: PRIORITY LEVEL 1
2195	000100	PR2=	100	:: PRIORITY LEVEL 2
2196	000140	PR3=	140	:: PRIORITY LEVEL 3
2197	000200	PR4=	200	:: PRIORITY LEVEL 4
2198	000240	PR5=	240	:: PRIORITY LEVEL 5
2199	000300	PR6=	300	:: PRIORITY LEVEL 6
2200	000340	PR7=	340	:: PRIORITY LEVEL 7
2201				
2202		: *"SWITCH REGISTER" SWITCH DEFINITIONS		
2203	100000	SW15=	100000	
2204	040000	SW14=	40000	
2205	020000	SW13=	20000	
2206	010000	SW12=	10000	
2207	004000	SW11=	4000	
2208	002000	SW10=	2000	
2209	001000	SW09=	1000	
2210	000400	SW08=	400	
2211	000200	SW07=	200	
2212	000100	SW06=	100	
2213	000040	SW05=	40	
2214	000020	SW04=	20	
2215	000010	SW03=	10	
2216	000004	SW02=	4	
2217	000002	SW01=	2	
2218	000001	SW00=	1	
2219		.EQUIV	SW09, SW9	
2220		.EQUIV	SW08, SW8	
2221		.EQUIV	SW07, SW7	
2222		.EQUIV	SW06, SW6	
2223		.EQUIV	SW05, SW5	
2224		.EQUIV	SW04, SW4	
2225		.EQUIV	SW03, SW3	
2226		.EQUIV	SW02, SW2	
2227		.EQUIV	SW01, SW1	
2228		.EQUIV	SW00, SW0	
2229				
2230		: *DATA BIT DEFINITIONS (BIT00 TO BIT15)		
2231	100000	BIT15=	100000	
2232	040000	BIT14=	40000	
2233	020000	BIT13=	20000	
2234	010000	BIT12=	10000	
2235	004000	BIT11=	4000	
2236	002000	BIT10=	2000	
2237	001000	BIT09=	1000	
2238	000400	BIT08=	400	
2239	000200	BIT07=	200	

```

2240      000100      BIT06= 100
2241      000040      BIT05= 40
2242      000020      BIT04= 20
2243      000010      BIT03= 10
2244      000004      BIT02= 4
2245      000002      BIT01= 2
2246      000001      BIT00= 1
2247      .EQUIV      BIT09,BIT9
2248      .EQUIV      BIT08,BIT8
2249      .EQUIV      BIT07,BIT7
2250      .EQUIV      BIT06,BIT6
2251      .EQUIV      BIT05,BIT5
2252      .EQUIV      BIT04,BIT4
2253      .EQUIV      BIT03,BIT3
2254      .EQUIV      BIT02,BIT2
2255      .EQUIV      BIT01,BIT1
2256      .EQUIV      BIT00,BIT0
2257
2258      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
2259      ERRVEC= 4      ;; TIME OUT AND OTHER ERRORS
2260      RESVEC= 10     ;; RESERVED AND ILLEGAL INSTRUCTIONS
2261      TBITVEC= 14    ;; "T" BIT
2262      TRTVEC= 14     ;; TRACE TRAP
2263      BPTVEC= 14     ;; BREAKPOINT TRAP (BPT)
2264      IOTVEC= 20     ;; INPUT/OUTPUT TRAP (IOT) **SCOPE**
2265      PWRVEC= 24     ;; POWER FAIL
2266      EMTVEC= 30     ;; EMULATOR TRAP (EMT) **ERROR**
2267      TRAPVEC= 34    ;; "TRAP" TRAP
2268      TKVEC= 60      ;; TTY KEYBOARD VECTOR
2269      TPVEC= 64      ;; TTY PRINTER VECTOR
2270      PIRQVEC= 240   ;; PROGRAM INTERRUPT REQUEST VECTOR
2271      176700      ABASE=176700
2272      000254      AVECT1=254
2273
2274      ;;*****
2275
2276      .SBTTL RM03 REGISTERS
2277
2278      ;;*****
2279
2280      ;CONTROL AND STATUS REGISTER 1 (RMCS1)
2281
2282      000100      IE= 100      ; INTERRUPT ENABLE (BIT #6)
2283      000200      RDY= 200     ; READY (BIT #7)
2284      000400      A16= 400     ; HIGH ORDER BUS ADDRESS BIT (BIT #8)
2285      001000      A17= 1000    ; HIGH ORDER BUS ADDRESS BIT (BIT #9)
2286      002000      PSEL= 2000   ; PORT SELECT (BIT #10)
2287      020000      MCPE= 20000  ; MASSBUSS PARITY ERROR (BIT #13)
2288      040000      TRE= 40000   ; TRANSFER ERROR (BIT #14)
2289      ;SC= 100000             ; SPECIAL CONDITION (BIT #15)
2290
2291      ;WORD COUNT REGISTER (RMWC)
2292      ;(EACH BIT IS CALLED BY BIT NUMBER)
2293
2294      ;BUS ADDRESS REGISTER (RMBA)
2295      ;(EACH BIT IS CALLED BY BIT NUMBER)

```



```

2296
2297
2298
2299      000001      US1=      1      ;UNIT SELECT (BIT #0)
2300      000002      US2=      2      ;UNIT SELECT (BIT #1)
2301      000004      US4=      4      ;UNIT SELECT (BIT #2)
2302      000010      BAI=     10      ;BUS ADDRESS INCREMENT INHIBIT (BIT #3)
2303      000020      PAT=     20      ;MASSBUS PARITY TEST (BIT #4)
2304      000040      CLR=     40      ;CLEAR (BIT #5)
2305      000100      IR=     100      ;INPUT READY (BIT #6)
2306      000200      OR=     200      ;OUTPUT READY (BIT #7)
2307      000400      MOPE=    400      ;MASS BUS PARITY ERROR (BIT #8)
2308      001000      MXF=    1000     ;MISSED TRANSFER ERROR (BIT #9)
2309      002000      PGE=    2000     ;PROGRAM ERROR (BIT #10)
2310      004000      NEM=    4000     ;NON EXISTENT MEMORY (BIT #11)
2311      010000      NED=    10000    ;NON EXISTENT DRIVE (BIT #12)
2312      020000      UPE=    20000    ;UNIBUS PARITY ERROR (BIT #13)
2313      040000      WCE=    40000    ;WRITE CHECK ERROR (BIT #14)
2314      100000      DLT=   100000    ;DATA LATE (BIT #15)
2315
2316      ;DATA BUFFER REGISTER (RMD8)
2317      ; (EACH BIT IS CALLED BY BIT NUMBER)
2318
2319
2320      ; ; *****
2321
2322      .SBTTL  RM03 REGISTERS
2323
2324      ; ; *****
2325
2326      ;CONTROL AND STATUS 1 REGISTER. (#00)
2327
2328      000001      GO=      1      ;GO BIT (BIT #0)
2329      000002      FO=      2      ;FUNCTION CODE BIT #1
2330      000004      F1=      4      ;FUNCTION CODE BIT #2
2331      000010      F2=     10      ;FUNCTION CODE BIT #3
2332      000020      F3=     20      ;FUNCTION CODE BIT #4
2333      000040      F4=     40      ;FUNCTION CODE BIT #5
2334      004000      DVA=    4000     ;DEVICE AVAILABLE (BIT #11)
2335
2336      ;DRIVE STATUS REGISTER (RMD51) (#01)
2337
2338      000001      OFFON= 1      ;OFFSET ON (BIT #0)
2339      000100      VV=     100      ;VOLUME VALID (BIT #6)
2340      000200      DRY=     200      ;DRIVE READY (BIT #7)
2341      000400      DPR=     400      ;DRIVE PRESENT (BIT #8)
2342      001000      PGM=    1000     ;PROGRAMABLE (BIT #9)
2343      002000      LBT=    2000     ;LAST SECTOR TRANSFERRED (BIT #10)
2344      004000      WRL=    4000     ;WRITE LOCK (BIT #11)
2345      010000      MOL=    10000    ;MEDIUM ON-LINE (BIT #12)
2346      020000      PIP=    20000    ;POSITIONING OPERATION IN PROGRESS (BIT #13)
2347      040000      ERR=    40000    ;COMPOSITE ERROR (BIT #14)
2348      100000      ATA=   100000    ;ATTENTION ACTIVE (BIT #15)
2349
2350      ;ERROR REGISTER #01 (RMER1) (#02)
2351

```

2352	000001	ILF=	1	; ILLEGAL FUNCTION (BIT #0)
2353	000002	ILR=	2	; ILLEGAL REGISTER (BIT #1)
2354	000004	RMR=	4	; REGISTER MODIFICATION REFUSED (BIT #2)
2355	000010	PAR=	10	; PARITY ERROR (BIT #3)
2356	000020	FER=	20	; FORMAT ERROR (BIT #4)
2357	000040	WCF=	40	; WRITE CLOCK FAIL (BIT #5)
2358	000100	ECH=	100	; ECC HARD ERROR (BIT #6)
2359	000200	HCE=	200	; HEADER COMPARE ERROR (BIT #7)
2360	000400	HCRC=	400	; HEADER CRC ERROR (BIT #8)
2361	001000	AOE=	1000	; ADDRESS OVERFLOW ERROR (BIT #9)
2362	002000	IAE=	2000	; INVALID ADDRESS ERROR (BIT #10)
2363	004000	WLE=	4000	; WRITE LOCK ERROR (BIT #11)
2364	010000	OTE=	10000	; DRIVE TIMING ERROR (BIT #12)
2365	020000	OPI=	20000	; OPERATION INCOMPLETE (BIT #13)
2366	040000	UNS=	40000	; DRIVE UNSAFE (BIT #14)
2367	100000	DCK=	100000	; DATA CHECK ERROR (BIT #15)
2368				
2369				; MAINTAINABILITY REGISTER (RMMR1) (#03)
2370				
2371				
2372				; ATTENTION SUMMARY PSEUDO-REGISTER (RMAS) (#04)
2373				
2374	000001	AT0=	1	; DEVICE 0 (BIT #0)
2375	000002	AT1=	2	; DEVICE 1 (BIT #1)
2376	000004	AT2=	4	; DEVICE 2 (BIT #2)
2377	000010	AT3=	10	; DEVICE 3 (BIT #3)
2378	000020	AT4=	20	; DEVICE 4 (BIT #4)
2379	000040	AT5=	40	; DEVICE 5 (BIT #5)
2380	000100	AT6=	100	; DEVICE 6 (BIT #6)
2381	000200	AT7=	200	; DEVICE 7 (BIT #7)
2382				
2383				; DESIRED SECTOR/TRACK ADDRESS REGISTER (RMDA) (#05)
2384				
2385				
2386				; DRIVE TYPE REGISTER (RMDT) (#06)
2387				
2388	000001	DT00=	1	; DRIVE TYPE NUMBER BIT 1
2389	000002	DT01=	2	; DRIVE TYPE NUMBER BIT 2
2390	000004	DT02=	4	; DRIVE TYPE NUMBER BIT 3
2391	000010	DT03=	10	; DRIVE TYPE NUMBER BIT 4
2392	000020	DT04=	20	; DRIVE TYPE NUMBER BIT 5
2393	000040	DT05=	40	; DRIVE TYPE NUMBER BIT 6
2394	000100	DT06=	100	; DRIVE TYPE NUMBER BIT 7
2395	000200	DT07=	200	; DRIVE TYPE NUMBER BIT 8
2396	000400	DT08=	400	; DRIVE TYPE NUMBER BIT 9
2397	004000	DRQ=	4000	; DRIVE REQUEST REQUIRED (BIT #11)
2398	020000	MOH=	20000	; MOVING HEAD (BIT #13)
2399	040000	TAP=	40000	; TAPE DRIVE (BIT #14)
2400	100000	NSA=	100000	; NOT SECTOR ADDRESSED (BIT #15)
2401				
2402				; LOOK-AHEAD REGISTER (RMLA) (#07)
2403				
2404	000100	SC1=	100	; SECTOR COUNT FIELD 0 (BIT #6)
2405	000200	SC2=	200	; SECTOR COUNT FIELD 1 (BIT #7)
2406	000400	SC04=	400	; SECTOR COUNT FIELD 2 (BIT #8)
2407	001000	SC10=	1000	; SECTOR COUNT FIELD 3 (BIT #9)

```

2408      002000      SC20= 2000      ;SECTOR COUNT FIELD 4 (BIT #10)
2409
2410      ;SERIAL NUMBER REGISTER (RMSN) (#10)
2411      ;(EACH IS CALLED BY BIT NUMBER)
2412      ;OFFSET REGISTER (RMOF) (#11)
2413
2414      000001      OFFDIR= 1      ;RM03 OFFSET DIRECTION
2415      002000      HCI= 2000      ;HEADER COMPARE INHIBIT (BIT #10)
2416      004000      ECI= 4000      ;ERROR CORRECTION CODE INHIBIT (BIT #11)
2417      010000      FMT16= 10000      ;FORMAT BIT (BIT #12)
2418
2419      ;DESIRED CYLINDER ADDRESS (RMDC) (#12)
2420      ;(EACH BIT IS CALLED BY BIT NUMBER)
2421
2422      ;CURRENT CYLINDER ADDRESS (RMCC) (#13)
2423      ;(REGISTER CURRENTLY NOT USED)
2424
2425      ;RM03 ERROR REGISTER #02 (RMER2) (#15)
2426
2427      000010      OPE= 10
2428      000200      DVC= 200
2429      002000      LBC= 2000
2430      004000      LSC= 4000
2431      010000      IVC= 10000
2432      020000      DPE= 20000
2433      040000      SKI= 40000      ;SEEK INCOMPLETE (BIT #14)
2434
2435      ;ECC POSITION REGISTER (RMEC1) (#16)
2436      ;(EACH BIT IS CALLED BY BIT NUMBER)
2437
2438      ;ECC PATTERN REGISTER (RMEC2) (#17)
2439      ;(EACH BIT IS CALLED BY BIT NUMBER)
2440
2441      ;*****
2442      .SBTTL RM03 DRIVER COMMANDS
2443      ;*****
2444
2445      000101      RNOP = 101      ;NO OPERATION
2446      000105      SEEK = 105      ;SEEK
2447      000107      RECAL = 107      ;RECALIBRATE
2448      000111      DRVCLR = 111      ;DRIVE CLEAR
2449      000113      RELSE = 113      ;RELEASE
2450      000115      OFFSET = 115      ;OFFSET
2451      000117      RTC = 117      ;RETURN TO CENTER LINE
2452      000121      READIN = 121      ;READ IN PRESET
2453      000123      ACK = 123      ;PACK ACKNOWLEDGE
2454      000131      SEARCH = 131      ;SEARCH
2455      000141      GETREG = 141      ;GET REGISTERS
2456      000143      SETFMT = 143      ;SET FORMAT (& ECI OR HCI)
2457      000145      SELDRV = 145      ;SELECT DRIVE
2458      000151      WCKD = 151      ;WRITE CHECK DATA
2459      000153      WCKHD = 153      ;WRITE CHECK HEADER & DATA
2460      000161      WRTOAT = 161      ;WRITE DATA
2461
2462
2463

```

```

2464      000163      WRTHD      =      163      ;WRITE HEADER & DATA
2465      000171      RDDAT      =      171      ;READ DATA
2466      000173      RDHD      =      173      ;READ HEADER & DATA
2467
2468      .SBTTL  TRAP CATCHER
2469
2470      000000      .=0
2471      ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
2472      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
2473      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
2474      000174      000174      .=174
2475      000174      000000      DISPRG: .WORD 0      ;;SOFTWARE DISPLAY REGISTER
2476      000176      000000      SWREG:  .WORD 0      ;;SOFTWARE SWITCH REGISTER
2477      .SBTTL  STARTING ADDRESS(ES)
2478      000200      000137      003632      JMP      @#START1      ;;JUMP TO STARTING ADDRESS OF PROGRAM
2479      000204      000137      003622      JMP      @#START      ;;CHANGE THE RH11 UNIBUS ADDRESS
2480      ;AFTER INITIAL START
2481
2482      .SBTTL  ACT11 HOOKS
2483
2484      ;*****
2485      ;HOOKS REQUIRED BY ACT11
2486      000210      $SVPC=.      ;SAVE PC
2487      000046      .=46
2488      000046      027450      $ENDAD      ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
2489      000052      000052      .=52
2490      000052      040000      .WORD 40000      ;;2)SET LOC.52 TO 40000
2491      000210      .=$SVPC      ;; RESTORE PC
2492      001100      .=1100
2493      .SBTTL  APT PARAMETER BLOCK
2494
2495      ;*****
2496      ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
2497      ;*****
2498      001100      .SX=.      ;SAVE CURRENT LOCATION
2499      000024      .=24      ;SET POWER FAIL TO POINT TO START OF PROGRAM
2500      000024      000200      200      ;FOR APT START UP
2501      000044      .=44      ;POINT TO APT INDIRECT ADDRESS PNTR.
2502      000044      001100      $APTHDR      ;POINT TO APT HEADER BLOCK
2503      001100      .=$X      ;RESET LOCATION COUNTER
2504
2505      ;*****
2506      ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
2507      ;INTERFACE SPEC.
2508      001100      $APTHD:
2509      001100      000000      $HIBTS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
2510      001102      001204      $MBADR: .WORD $MAIL      ;;ADDRESS OF APT MAILBOX (BITS 0-15)
2511      001104      000264      $TSTM:  .WORD 180.      ;;RUN TIM OF LONGEST TEST
2512      001106      000264      $PASTM: .WORD 180.      ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
2513      001110      000264      $UNITM: .WORD 180.      ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
2514      001112      000032      .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
2515      001114      TAB.XY=.

```

```

2516      .SBTTL  COMMON TAGS
2517
2518      ;*****
2519      ;THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
2520      ;USED IN THE PROGRAM.
2521
2522      001114      .=TAB.XY
2523      SCMTAG:      .WORD 0      ;; START OF COMMON TAGS
2524      001114      000000      ;; CONTAINS THE TEST NUMBER
2525      001116      000      ;; CONTAINS ERROR FLAG
2526      001117      000      ;; CONTAINS SUBTEST ITERATION COUNT
2527      001120      000000      ;; CONTAINS SCOPE LOOP ADDRESS
2528      001122      000000      ;; CONTAINS SCOPE RETURN FOR ERRORS
2529      001124      000000      ;; CONTAINS TOTAL ERRORS DETECTED
2530      001126      000000      ;; CONTAINS ITEM CONTROL BYTE
2531      001130      000      ;; CONTAINS MAX. ERRORS PER TEST
2532      001131      001      ;; CONTAINS PC OF LAST ERROR INSTRUCTION
2533      001132      000000      ;; CONTAINS ADDRESS OF 'GOOD' DATA
2534      001134      000000      ;; CONTAINS ADDRESS OF 'BAD' DATA
2535      001136      000000      ;; CONTAINS 'GOOD' DATA
2536      001140      000000      ;; CONTAINS 'BAD' DATA
2537      001142      000000      ;; RESERVED--NOT TO BE USED
2538      001144      000000      ;;
2539      001146      000000      ;;
2540      001150      000      ;; AUTOMATIC MODE INDICATOR
2541      001151      000      ;; INTERRUPT MODE INDICATOR
2542      001152      000000      ;;
2543      001154      177570      ;; ADDRESS OF SWITCH REGISTER
2544      001156      177570      ;; ADDRESS OF DISPLAY REGISTER
2545      001160      177560      ;; TTY KBD STATUS
2546      001162      177562      ;; TTY KBD BUFFER
2547      001164      177564      ;; TTY PRINTER STATUS REG. ADDRESS
2548      001166      177566      ;; TTY PRINTER BUFFER REG. ADDRESS
2549      001170      000      ;; CONTAINS NULL CHARACTER FOR FILLS
2550      001171      002      ;; CONTAINS # OF FILLER CHARACTERS REQUIRED
2551      001172      012      ;; INSERT FILL CHARS. AFTER A "LINE FEED"
2552      001173      000      ;; "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
2553      001174      177607      000377      ;; CODE FOR BELL
2554      001200      077      ;; QUESTION MARK
2555      001201      015      ;; CARRIAGE RETURN
2556      001202      000012      ;; LINE FEED
2557      ;*****
2558      .SBTTL  APT MAILBOX-ETABLE
2559
2560      ;*****
2561      .EVEN
2562      $MAIL:      ;; APT MAILBOX
2563      $MSGTY:      .WORD  AMSGTY  ;; MESSAGE TYPE CODE
2564      $FATAL:      .WORD  AFATAL   ;; FATAL ERROR NUMBER
2565      $TESTN:      .WORD  ATESTN   ;; TEST NUMBER
2566      $PASS:      .WORD  APASS     ;; PASS COUNT
2567      $DEVCT:      .WORD  ADEVCT   ;; DEVICE COUNT
2568      $UNIT:      .WORD  AUNIT     ;; I/O UNIT NUMBER
2569      $MSGAD:      .WORD  AMSGAD    ;; MESSAGE ADDRESS
2570      $MSGLG:      .WORD  AMSGLG    ;; MESSAGE LENGTH
2571      $ETABLE:      ;; APT ENVIRONMENT TABLE

```

M04

CZRM880 RM03/2 PERF EXER
CZRM88.P11 21-NOV-77 15:34

MACY11 30(1046) 22-NOV-77 10:06 PAGE 52
APT MAILBOX-ETABLE

SEQ 0051

2572	001224	000			\$ENV: .BYTE	AENV	:: ENVIRONMENT BYTE
2573	001225	000			\$ENVM: .BYTE	AENVM	:: ENVIRONMENT MODE BITS
2574	001226	000000			\$SWREG: .WORD	ASWREG	:: APT SWITCH REGISTER
2575	001230	000000			\$USWR: .WORD	AUSWR	:: USER SWITCHES
2576	001232	000000			\$CPUOP: .WORD	ACPUOP	:: CPU TYPE, OPTIONS
2577					.*		BITS 15-11=CPU TYPE
2578					.*		11/04=01, 11/05=02, 11/20=03, 11/40=04, 11/45=05
2579					.*		11/70=06, PDQ=07, Q=10
2580					.*		BIT 10=REAL TIME CLOCK
2581					.*		BIT 9=FLOATING POINT PROCESSOR
2582					.*		BIT 8=MEMORY MANAGEMENT
2583	001234	000			\$MAMS1: .BYTE	AMAMS1	:: HIGH ADDRESS, M.S. BYTE
2584	001235	000			\$MTYP1: .BYTE	AMTYP1	:: MEM. TYPE, BLK#1
2585					.*		MEM. TYPE BYTE -- (HIGH BYTE)
2586					.*		900 NSEC CORE=001
2587					.*		300 NSEC BIPOLAR=002
2588					.*		500 NSEC MOS=003
2589	001236	000000			\$MADR1: .WORD	AMADR1	:: HIGH ADDRESS, BLK#1
2590					.*		MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
2591	001240	000			\$MAMS2: .BYTE	AMAMS2	:: HIGH ADDRESS, M.S. BYTE
2592	001241	000			\$MTYP2: .BYTE	AMTYP2	:: MEM. TYPE, BLK#2
2593	001242	000000			\$MADR2: .WORD	AMADR2	:: MEM. LAST ADDRESS, BLK#2
2594	001244	000			\$MAMS3: .BYTE	AMAMS3	:: HIGH ADDRESS, M.S. BYTE
2595	001245	000			\$MTYP3: .BYTE	AMTYP3	:: MEM. TYPE, BLK#3
2596	001246	000000			\$MADR3: .WORD	AMADR3	:: MEM. LAST ADDRESS, BLK#3
2597	001250	000			\$MAMS4: .BYTE	AMAMS4	:: HIGH ADDRESS, M.S. BYTE
2598	001251	000			\$MTYP4: .BYTE	AMTYP4	:: MEM. TYPE, BLK#4
2599	001252	000000			\$MADR4: .WORD	AMADR4	:: MEM. LAST ADDRESS, BLK#4
2600	001254	000254			\$VECT1: .WORD	AVECT1	:: INTERRUPT VECTOR#1, BUS PRIORITY#1
2601	001256	000000			\$VECT2: .WORD	AVECT2	:: INTERRUPT VECTOR#2, BUS PRIORITY#2
2602	001260	176700			\$BASE: .WORD	ABASE	:: BASE ADDRESS OF EQUIPMENT UNDER TEST
2603	001262	000000			\$DEVN: .WORD	ADEVN	:: DEVICE MAP
2604	001264	000000			\$CDW1: .WORD	ACDW1	:: CONTROLLER DESCRIPTION WORD#1
2605	001266	000000			\$CDW2: .WORD	ACDW2	:: CONTROLLER DESCRIPTION WORD#2
2606	001270				\$ETEND:		
2607					.MEXIT		
2608		000015			CR	=	15
2609		000012			LF	=	12
2610	001270	176700			\$RMADR: .WORD	176700	:: FIRST ADDRESS OF RM11/RM03 REGISTERS
2611	001272	000254			\$RMVEC: .WORD	254	:: RM03 VECTOR ADDRESS
2612	001274	172540			\$LKCSR: .WORD	172540	:: ADDR OF KW11-P STATUS REGISTER
2613	001276	172542			\$LKCSB: .WORD	172542	:: ADDR OF KW11-P COUNTER BUFFER
2614	001300	000104			\$LPVEC: .WORD	104	:: ADDR OF KW11-P VECTOR
2615	001302	177546			\$LKS: .WORD	177546	:: ADDR OF KW11-L STATUS REGISTER
2616	001304	000100			\$LLVEC: .WORD	100	:: ADDR OF KW11-L VECTOR
2617	001306	177777			PCLOCK: .WORD	-1	:: '0' IF KW11-P IS ON SYSTEM
2618	001310	177777			CLKFLG: .WORD	-1	:: '0' IF A CLOCK IS AVAILABLE
2619	001312	000074			HZ: .WORD	74	:: 74 (8) IF 60 HZ SYSTEM; 62 (8) IF 50 HZ SYSTEM
2620	001314	000000			STATIN: .WORD	0	:: 'TYPE STATISTICS' INDICATOR
2621	001316	000000			PACK: .WORD	0	:: 'W' COMMAND INDICATOR
2622	001320	000000	000000	000000	DATE: .WORD	0,0,0,0,0	:: OPERATOR ENTERED DATE
2623	001326	000000	000000				
2624	001332	000000	000000	000000	OPERID: .WORD	0,0,0,0	:: OPERATOR ID
2625	001340	000000					
2626		001216			DRIVE	=	\$UNIT
2627	001342	000000			ATTN: .WORD	0	:: DRIVE # STORAGE: ERRORS 1-5 & 10 ATTN REG STORAGE: ERRORS 1-5 & 10

2628	001344	000000	UNIT:	.WORD	0	;DRIVE # STORAGE FOR PRINTOUT
2629	001346	000000	MASK:	.WORD	0	;ERROR RETRY REGISTER MASK
2630	001350	000	RETRY:	.BYTE	0,0	;ERROR RETRY LIMIT IN THE LOWER BYTE
2631						;RETRY COUNT IN THE UPPER BYTE
2632	001352	000003	FAIRNS:	.WORD	3	;MAXIMUM TIME IN QUEUE VALUE
2633	001354	000000	LSTAD:	.WORD	0	;STORE LAST MEMORY ADDRESS HERE
2634	001356	000000	CHGADR:	.WORD	0	;CHANGE RH11 UNIBUS ADDRESS FLAG
2635	001360	000000	CFLAG:	.WORD	0	;CONTROL 'C' FLAG
2636	001362	000000	BAC SEC:	.WORD	0	;BAD SECTOR/TRACK FLAG
2637	001364	000000	HOUR:	.WORD	0	;HOUR COUNT STORED HERE (MAXIMUM - 999.)
2638	001366	000000	MINUTE:	.WORD	0	;MINUTE'S COUNT STORED HERE
2639	001370	000000	SECOND:	.WORD	0	;SECOND'S COUNT STORED HERE
2640	001372	000000	SIXTEE:	.WORD	0	;TIMER ROUTINE COUNTER (FOR ONE SECOND)
2641	001374	177777	ZROIND:	.WORD	-1	;ZERO INDICATOR FOR THE DATA COMPARE ROUTINE
2642	001376	000	FRSTER:	.BYTE	0	;DATA COMPARE ERROR FLAG
2643						;IF > 0, PROCESSING 'DCKER' OR CAN'T MATCH PATTERN
2644						;IF < 0, MISCOMPARISON FOUND
2645	001377	000		.BYTE	0	;MISCOMPARISON OR CAN'T MATCH PATTERN FLAG
2646						;IF < 0, ERROR IN BUFFER
2647	001400	000000	SAVER1:	.WORD	0	;SAVE R1 HERE
2648	001402	000000	SAVER5:	.WORD	0	;SAVE R5 HERE
2649	001404	000000	ERCTR:	.WORD	0	;NUMBER OF ERRORS
2650	001406	000000	LIMIT:	.WORD	0	;DISPLAY LIMIT
2651	001410	000000	CMCNT:	.WORD	0	;WORD COUNT
2652	001412	000000	CMCYL:	.WORD	0	;CYLINDER ADDRESS
2653	001414	000	CMSEC:	.BYTE	0	;SECTOR ADDRESS
2654	001415	000	CMTRK:	.BYTE	0	;TRACK ADDRESS
2655	001416	000000	ECBIT:	.WORD	0	;ERROR BURST BIT OFFSET
2656	001420	000000	ECSEC:	.WORD	0	;ERROR BURST WORD OFFSET (RELATIVE TO SECTOR)
2657	001422	000000	ECMSKO:	.WORD	0	;CORRECTION MASK FOR FIRST ERROR WORD
2658	001424	000000	ECMSK1:	.WORD	0	;CORRECTION MASK FOR SECOND ERROR WORD
2659	001426	000000	ECWRD:	.WORD	0	;LOCATION OF FIRST ERROR WORD
2660	001430	000000	ECGD:	.WORD	0	;GOOD DATA, FIRST WORD
2661	001432	000000	ECBADO:	.WORD	0	;BAD DATA, FIRST WORD
2662	001434	000000	ECWRD1:	.WORD	0	;LOCATION OF SECOND ERROR WORD
2663	001436	000000	ECGD1:	.WORD	0	;GOOD DATA, SECOND WORD
2664	001440	000000	ECBAD1:	.WORD	0	;BAD DATA, SECOND WORD
2665	001442	000037	SECLMT:	.WORD	31.	;SECTOR ADDRESS LIMIT
2666	001444	000004	TRKLMT:	.WORD	4.	;TRACK ADDRESS LIMIT
2667	001446	001465	CYLMT:	.WORD	821.	;CYLINDER ADDRESS LIMIT FOR RMD3
2668						
2669						;*****
2670						
2671			.SBTTL	COMMON PARAMETERS		
2672						
2673						;*****
2674						
2675	001450	002740	ENDCON:	.WORD	002740	;1.875X10 ¹⁸ WORDS (10) [3X10 ¹⁹ BITS]
2676	001452	005455		.WORD	005455	;MSW
2677	001454	143300	ENDSEK:	.WORD	143300	;3 X 10 ¹⁶ SEKS (LSW)
2678	001456	000055		.WORD	55	;MSW
2679	001460	000001	PASCNT:	.WORD	1	;NUMBER OF PASSES TO END OF TEST
2680	001462	000000	MAXDL:	.WORD	0	;MAXIMUM DATA TRANSFER SIZE IN WORDS
2681						; (FILLED BY PROGRAM AT STARTUP OR BY OPERATOR
2682						; DURING PARAMETER ENTRY DIALOG.)
2683	001464	000144	MAXER:	.WORD	100.	;MAXIMUM ERRORS - 100(10)

```

2684 001466 000005 000000 INTRVL: .WORD 5,0
2685
2686
2687
2688 001472 000004 CMPLMT: .WORD 4
2689 001474 000001 FORMAT: .WORD 1
2690
2691 001476 000000 WCSEL: .WORD 0
2692
2693
2694
2695 001500 000003 RATIO: .WORD 3
2696
2697
2698
2699
2700
2701
2702
2703
2704 001502 000001 AUTOCK: .WORD 1
2705
2706
2707
2708 001504 000001 NOTPRT: .WORD 1
2709
2710
2711
2712
2713 001506 000001 ENDET: .WORD 1
2714
2715
2716
2717 001510 000000 PATTEN: .WORD 0
2718
2719
2720 001512 000000 HEADER: .WORD 0
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731 001514 000010 BEGPAT: .WORD 10
2732 001516 000004 BEGCOD: .WORD 4
2733
2734
2735
2736
2737
2738
2739 001520 000400 BEGSIZ: .WORD 400

```

```

: FIRST WORD IS THE PERFORMANCE TIMEOUT INTERVAL
: (IN MINUTES). SECOND WORD IS THE INTERVAL
: COUNTER. UPPER BYTE IS VALUE.
: NUMBER OF COMPARE ERRORS TYPED OUT
: IF NOT EQ 0, ALLOW WRITE HEADER & DATA ORDERS
: IF EQ 0, DO NOT ALLOW WRITE HEADER & DATA ORDERS
: IF EQ TO 0, GENERATE A RANDOM WORD COUNT
: FOR THE OPERATION.
: IF NOT EQ TO 0, USE THE VALUE IN 'MAXDL' FOR
: THE WORD COUNT
: READ/WRITE RATIO (RANGE 0 - 7)
: 0 - 0/8 (READ/WRITE)
: 1 - 7/1
: 2 - 6/2
: 3 - 5/3
: 4 - 4/4
: 5 - 3/5
: 6 - 2/6
: 7 - 1/7
: IF NOT EQ 0, DO AN APPROPRIATE WRITE
: CHECK AFTER EACH WRITE ORDER.
: IF EQ 0, SELECT WRITE CHECK ORDERS
: RANDOMLY.
: IF EQ 1, DO NOT PRINT DATA ERROR MESSAGES
: ASSOCIATED WITH OPERATOR SPECIFIED
: BAD PACK AREAS.
: IF NOT EQ 0, PRINT ERROR MESSAGES RELATING TO
: THESE AREAS.
: IF NOT EQ 0, END OF PASS DETERMINED
: BY THE 'WORDS READ' COUNT.
: IF EQ 0, END OF PASS DETERMINED
: BY THE SEEK COUNT.
: IF EQ 0, RANDOMLY SELECT DATA PATTERN
: IF NOT EQ 0, SELECT ONE SET OF PATTERN
: POINTED BY THE "PATTEN".
: IF EQ TO 0, RANDOMLY SELECT DATA BLOCK
: ADDRESS. IF NOT EQU 0, SEQUENTIALLY
: SELECT DATA BLOCK ADDRESS

```

```

;;*****

```

```

.SBTTL VALUES FOR FIRST OPERATION

```

```

;;*****

```

```

: STARTING PATTERN CODE (RANGE 1 - 17 (OCTAL))
: STARTING COMMAND CODE (RANGE 0 - 5)
: 0 = WRITE CHECK DATA ('WCKD')
: 1 = WRITE CHECK HEADER & DATA ('WCHKHD')
: 2 = WRITE DATA ('WRDAT')
: 3 = WRITE HEADER & DATA ('WRTHD')
: 4 = READ DATA ('RODAT')
: 5 = READ HEADER & DATA ('RDHD')
: STARTING RECORD SIZE (RANGE 4 - MAXMEM)

```



```

2740
2741      ;*****
2742
2743      .SBTTL  TABLES, CONSTANTS, AND VARIABLE LOCATIONS
2744
2745      ;*****
2746
2747      001522  000000  000000  000000  ORDERQ: .WORD  0,0,0,0,0,0,0,0,0,0 ;LIST OF DRIVES PERFORMING COMMANDS
2748      001530  000000  000000  000000
2749      001536  000000  000000  000000
2750
2751      001544  000000  ASNLST: .WORD  0 ;A BIT SET IS AN ASSIGNED DRIVE
2752
2753      001546  000000  000000  000000  DUNIT: .WORD  0,0,0,0,0,0,0,0,0,0 ;ADDRESSES OF DRIVES TO BE DEASSIGNED
2754      001554  000000  000000  000000
2755      001562  000000  000000  000000
2756
2757      001570  000000  000000  000000  NEWUNT: .WORD  0,0,0,0,0,0,0,0,0,0 ;ADDRESSES OF NEWLY ASSIGNED DRIVES
2758      001576  000000  000000  000000
2759      001604  000000  000000  000000
2760
2761      001612  000000  000000  000000  AVAIL: .WORD  0,0,0,0,0,0,0,0,0,0 ;LIST OF DRIVES WAITING FOR BUFFERS/PARAMETERS
2762      001620  000000  000000  000000
2763      001626  000000  000000  000000
2764
2765      001634  000000  000000  000000  WAIT: .WORD  0,0,0,0,0,0,0,0,0,0 ;LIST OF DRIVES WAITING FOR BUFFERS
2766      001642  000000  000000  000000
2767      001650  000000  000000  000000
2768
2769      001656  000000  000000  000000  PARQ: .WORD  0,0,0,0,0,0,0,0,0,0 ;LIST OF DRIVES WAITING FOR NEXT PARAMETERS
2770      001664  000000  000000  000000
2771      001672  000000  000000  000000
2772
2773      001700  000000  BUFTBL: .WORD  0 ;BUFFER ALLOCATION TABLE ENTRY COUNT
2774      001702  000000  .WORD  0,0
2775      001706  000000  .WORD  0,0
2776      001712  000000  .WORD  0,0
2777      001716  000000  .WORD  0,0
2778      001722  000000  .WORD  0,0
2779      001726  000000  .WORD  0,0
2780      001732  000000  .WORD  0,0
2781      001736  000000  .WORD  0,0
2782      001742  000000  .WORD  0,0
2783      001746  000000  .WORD  0,0
2784      001752  000000  .WORD  0,0
2785      001756  000000  .WORD  0,0
2786      001762  000000  .WORD  0,0
2787      001766  000000  .WORD  0,0
2788      001772  000000  .WORD  0,0
2789      001776  000000  .WORD  0,0
2790      002002  000000  .WORD  0,0
2791      002006  000000  .WORD  0,0
2792      002012  000000  .WORD  0,0
2793      002016  000000  .WORD  0,0
2794      002022  000000  .WORD  0,0
2795      002026  000000  .WORD  0,0

```

2796	002032	000000	000000	.WORD	0,0	
2797	002036	000000	000000	.WORD	0,0	
2798	002042	000000	000000	.WORD	0,0	
2799	002046	000000	000000	.WORD	0,0	
2800	002052	000000	000000	.WORD	0,0	
2801	002056	000000	000000	.WORD	0,0	
2802	002062	000000	000000	.WORD	0,0	
2803	002066	000000	000000	.WORD	0,0	
2804	002072	000000	000000	.WORD	0,0	
2805	002076	000000	000000	.WORD	0,0	
2806						
2807	002102	043022		BLKADR: .WORD	DRIVE0	: ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 0
2808	002104	043326		.WORD	DRIVE1	: ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 1
2809	002106	043632		.WORD	DRIVE2	: ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 2
2810	002110	044136		.WORD	DRIVE3	: ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 3
2811	002112	044442		.WORD	DRIVE4	: ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 4
2812	002114	044746		.WORD	DRIVE5	: ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 5
2813	002116	045252		.WORD	DRIVE6	: ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 6
2814	002120	045556		.WORD	DRIVE7	: ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 7
2815						
2816	002122	151		COMTBL: .BYTE	WCKD	: WRITE CHECK DATA
2817	002123	153		.BYTE	WCKHD	: WRITE CHECK HEADER AND DATA
2818	002124	161		.BYTE	WRTDAT	: WRITE DATA
2819	002125	163		.BYTE	WRTHD	: WRITE HEADER AND DATA
2820	002126	171		.BYTE	RDDAT	: READ DATA
2821	002127	173		.BYTE	RDDHD	: READ HEADER AND DATA
2822						
2823	002130	002		OPTBL: .BYTE	2	: UNLOAD
2824	002131	004		.BYTE	4	: SEEK
2825	002132	006		.BYTE	6	: RECAL
2826	002133	010		.BYTE	10	: DRIVE CLEAR
2827	002134	012		.BYTE	12	: RELEASE
2828	002135	014		.BYTE	14	: OFFSET
2829	002136	016		.BYTE	16	: RETURN TO CENTERLINE
2830	002137	020		.BYTE	20	: READIN PRESET
2831	002140	022		.BYTE	22	: PACK ACKNOWLEDGE
2832	002141	030		.BYTE	30	: SEARCH
2833	002142	050		.BYTE	50	: WRITE CHECK DATA
2834	002143	052		.BYTE	52	: WRITE CHECK HEADER AND DATA
2835	002144	060		.BYTE	60	: WRITE DATA
2836	002145	062		.BYTE	62	: WRITE HEADER AND DATA
2837	002146	070		.BYTE	70	: READ DATA
2838	002147	072		.BYTE	72	: READ HEADER AND DATA
2839	002150	377		.BYTE	-1	: TERMINATOR
2840						
2841		002152		.EVEN		
2842						
2843						
2844	002152	047125	047514	042101	MNTBL: .ASCIZ	/UNLOAD /
2845	002160	000040				
2846	002162	042523	045505	020040	.ASCIZ	/SEEK /
2847	002170	000040				
2848	002172	042522	040503	020114	.ASCIZ	/RECAL /
2849	002200	000040				
2850	002202	051104	041526	051114	.ASCIZ	/DRVCLR /
2851	002210	000040				

CZRM880 RM03/2 PERF EXER
CZRM88.P11 21-NOV-77 15.34

MACY11 30(1046) 22-NOV-77 10:06 PAGE 57
TABLES, CONSTANTS, AND VARIABLE LOCATIONS

```

2852 002212 042522 051514 020105 .ASCIZ /RE_SE /
2853 002220 000040
2854 002222 043117 051506 052105 .ASCIZ /OFFSET /
2855 002230 000040
2856 002232 052122 020103 020040 .ASCIZ /RTC /
2857 002240 000040
2858 002242 042522 042101 047111 .ASCIZ /READIN /
2859 002250 000040
2860 002252 040520 045503 020040 .ASCIZ /PACK /
2861 002260 000040
2862 002262 042523 051101 044103 .ASCIZ /SEARCH /
2863 002270 000040
2864 002272 041527 042113 020040 .ASCIZ /WCKD /
2865 002300 000040
2866 002302 041527 044113 020104 .ASCIZ /WCKHD /
2867 002310 000040
2868 002312 051127 042124 052101 .ASCIZ /WRTDAT /
2869 002320 000040
2870 002322 051127 044124 020104 .ASCIZ /WRTHD /
2871 002330 000040
2872 002332 042122 040504 020124 .ASCIZ /RDDAT /
2873 002340 000040
2874 002342 042122 042110 020040 .ASCIZ /RDHD /
2875 002350 000040
2876 002352 047516 042516 020040 .ASCIZ /NONE /
2877 002360 000040
2878
2879
2880 002362 043101 042524 020122 OFMSG0: .ASCIZ /AFTER RETRY WITHOUT OFFSET/
2881 002370 042522 051124 020131
2882 002376 044527 044124 052517
2883 002404 020124 043117 051506
2884 002412 052105 000
2885 002415 101 020124 042516 OFMSG1: .ASCIZ /AT NEGATIVE OFFSET/
2886 002422 040507 044524 042526
2887 002430 047440 043106 042523
2888 002436 000124
2889 002440 052101 050040 051517 OFMSG2: .ASCIZ /AT POSITIVE OFFSET/
2890 002446 052111 053111 020105
2891 002454 043117 051506 052105
2892 002462 000
2893
2894 002464 .EVEN
2895
2896 002464 000 OFFCOD: .BYTE 0 ;OFFSET CODE TABLE
2897 002465 001 .BYTE 1 ;NUMBER FOR NEGATIVE OFFSET (DIR = OU*)
2898 002466 000 .BYTE 0 ;NUMBER FOR POSITIVE OFFSET (DIR = IN)
2899
2900 002470 .EVEN
2901
2902 002470 002362 OFMTBL: .WORD OFMSG0 ;1ST OFFSET MESSAGE
2903 002472 002415 .WORD OFMSG1 ;2ND OFFSET MESSAGE
2904 002474 002440 .WORD OFMSG2 ;3RD OFFSET MESSAGE
2905
2906
2907

```

```

2908
2909      .SBTTL  DATA PATTERNS
2910
2911      ;*****
2912
2913      STNDAT: .WORD  DATA0          ;STANDARD DATA PATTERN POINTER TABLE
2914              .WORD  DATA1
2915              .WORD  DATA1+40
2916              .WORD  DATA1+100
2917              .WORD  DATA1+140
2918              .WORD  DATA1+200
2919              .WORD  DATA1+240
2920              .WORD  DATA1+300
2921              .WORD  DATA1+340
2922              .WORD  DATA1+400
2923              .WORD  DATA1+440
2924              .WORD  DATA1+500
2925              .WORD  DATA1+540
2926              .WORD  DATA1+600
2927              .WORD  DATA1+640
2928              .WORD  DATA1+700
2929              .WORD  DATA0          ;ZER0ES
2930              .WORD  DATA1+642     ;ONES
2931
2932      DATA0: .WORD  0              ;DUMMY DATA PATTERN
2933              .WORD  0
2934              .WORD  0
2935              .WORD  0
2936              .WORD  0
2937              .WORD  0
2938              .WORD  0
2939              .WORD  0
2940              .WORD  0
2941              .WORD  0
2942              .WORD  0
2943              .WORD  0
2944              .WORD  0
2945              .WORD  0
2946              .WORD  0
2947              .WORD  0
2948
2949      DATA1: .WORD  000001          ;STANDARD PATTERN 1
2950              .WORD  000003
2951              .WORD  000007
2952              .WORD  000017
2953              .WORD  000037
2954              .WORD  000077
2955              .WORD  000177
2956              .WORD  000377
2957              .WORD  000777
2958              .WORD  001777
2959              .WORD  003777
2960              .WORD  007777
2961              .WORD  017777
2962              .WORD  037777
2963              .WORD  077777
2908      002476  002542
2909      002500  002602
2910      002502  002642
2911      002504  002702
2912      002506  002742
2913      002510  003002
2914      002512  003042
2915      002514  003102
2916      002516  003142
2917      002520  003202
2918      002522  003242
2919      002524  003302
2920      002526  003342
2921      002530  003402
2922      002532  003442
2923      002534  003502
2924      002536  002542
2925      002540  003444
2926      002542  000000
2927      002544  000000
2928      002546  000000
2929      002550  000000
2930      002552  000000
2931      002554  000000
2932      002556  000000
2933      002560  000000
2934      002562  000000
2935      002564  000000
2936      002566  000000
2937      002570  000000
2938      002572  000000
2939      002574  000000
2940      002576  000000
2941      002600  000000
2942      002602  000001
2943      002604  000003
2944      002606  000007
2945      002610  000017
2946      002612  000037
2947      002614  000077
2948      002616  000177
2949      002620  000377
2950      002622  000777
2951      002624  001777
2952      002626  003777
2953      002630  007777
2954      002632  017777
2955      002634  037777
2956      002636  077777

```

2964	002640	177777	.WORD	177777	
2965					
2966	002642	177776	.WORD	177776	:STANDARD PATTERN 2
2967	002644	177774	.WORD	177774	
2968	002646	177770	.WORD	177770	
2969	002650	177760	.WORD	177760	
2970	002652	177740	.WORD	177740	
2971	002654	177700	.WORD	177700	
2972	002656	177600	.WORD	177600	
2973	002660	177400	.WORD	177400	
2974	002662	177000	.WORD	177000	
2975	002664	176000	.WORD	176000	
2976	002666	174000	.WORD	174000	
2977	002670	170000	.WORD	170000	
2978	002672	160000	.WORD	160000	
2979	002674	140000	.WORD	140000	
2980	002676	100000	.WORD	100000	
2981	002700	000000	.WORD	000000	
2982					
2983	002702	000000	.WORD	000000	:STANDARD PATTERN 3
2984	002704	000000	.WORD	000000	
2985	002706	000000	.WORD	000000	
2986	002710	177777	.WORD	177777	
2987	002712	177777	.WORD	177777	
2988	002714	177777	.WORD	177777	
2989	002716	000000	.WORD	000000	
2990	002720	000000	.WORD	000000	
2991	002722	177777	.WORD	177777	
2992	002724	177777	.WORD	177777	
2993	002726	000000	.WORD	000000	
2994	002730	177777	.WORD	177777	
2995	002732	000000	.WORD	000000	
2996	002734	177777	.WORD	177777	
2997	002736	000000	.WORD	000000	
2998	002740	177777	.WORD	177777	
2999					
3000	002742	133331	.WORD	133331	:STANDARD PATTERN 4
3001	002744	133331	.WORD	133331	
3002	002746	133331	.WORD	133331	
3003	002750	133331	.WORD	133331	
3004	002752	133331	.WORD	133331	
3005	002754	133331	.WORD	133331	
3006	002756	133331	.WORD	133331	
3007	002760	133331	.WORD	133331	
3008	002762	133331	.WORD	133331	
3009	002764	133331	.WORD	133331	
3010	002766	133331	.WORD	133331	
3011	002770	133331	.WORD	133331	
3012	002772	133331	.WORD	133331	
3013	002774	133331	.WORD	133331	
3014	002776	133331	.WORD	133331	
3015	003000	133331	.WORD	133331	
3016					
3017	003002	052525	.WORD	052525	:STANDARD PATTERN 5
3018	003004	052525	.WORD	052525	
3019	003006	052525	.WORD	052525	

3020	003010	125252	.WORD	125252	
3021	003012	125252	.WORD	125252	
3022	003014	125252	.WORD	125252	
3023	003016	052525	.WORD	052525	
3024	003020	052525	.WORD	052525	
3025	003022	125252	.WORD	125252	
3026	003024	125252	.WORD	125252	
3027	003026	052525	.WORD	052525	
3028	003030	125252	.WORD	125252	
3029	003032	052525	.WORD	052525	
3030	003034	125252	.WORD	125252	
3031	003036	052525	.WORD	052525	
3032	003040	125252	.WORD	125252	
3033					
3034	003042	155554	.WORD	155554	; STANDARD PATTERN 6
3035	003044	155554	.WORD	155554	
3036	003046	155554	.WORD	155554	
3037	003050	155554	.WORD	155554	
3038	003052	155554	.WORD	155554	
3039	003054	155554	.WORD	155554	
3040	003056	155554	.WORD	155554	
3041	003060	155554	.WORD	155554	
3042	003062	155554	.WORD	155554	
3043	003064	155554	.WORD	155554	
3044	003066	155554	.WORD	155554	
3045	003070	155554	.WORD	155554	
3046	003072	155554	.WORD	155554	
3047	003074	155554	.WORD	155554	
3048	003076	155554	.WORD	155554	
3049	003100	155554	.WORD	155554	
3050					
3051	003102	026455	.WORD	026455	; STANDARD PATTERN 7
3052	003104	026455	.WORD	026455	
3053	003106	026455	.WORD	026455	
3054	003110	151322	.WORD	151322	
3055	003112	151322	.WORD	151322	
3056	003114	151322	.WORD	151322	
3057	003116	026455	.WORD	026455	
3058	003120	026455	.WORD	026455	
3059	003122	151322	.WORD	151322	
3060	003124	151322	.WORD	151322	
3061	003126	026455	.WORD	026455	
3062	003130	151322	.WORD	151322	
3063	003132	026455	.WORD	026455	
3064	003134	151322	.WORD	151322	
3065	003136	026455	.WORD	026455	
3066	003140	151322	.WORD	151322	
3067					
3068	003142	066666	.WORD	066666	; STANDARD PATTERN 8
3069	003144	066666	.WORD	066666	
3070	003146	066666	.WORD	066666	
3071	003150	066666	.WORD	066666	
3072	003152	066666	.WORD	066666	
3073	003154	066666	.WORD	066666	
3074	003156	066666	.WORD	066666	
3075	003160	066666	.WORD	066666	

CZRM880 RM03/2 PERF EXER
CZRM88.P11 21-NOV-77 15:34

MACY11 30(1046) 22-NOV-77 10:06 PAGE 61
DATA PATTERNS

SEQ 0060

3076	003162	066666	.WORD	066666	
3077	003164	066666	.WORD	066666	
3078	003166	066666	.WORD	066666	
3079	003170	066666	.WORD	066666	
3080	003172	066666	.WORD	066666	
3081	003174	066666	.WORD	066666	
3082	003176	066666	.WORD	066666	
3083	003200	066666	.WORD	066666	
3084					
3085	003202	000001	.WORD	000001	; STANDARD PATTERN 9
3086	003204	000002	.WORD	000002	
3087	003206	000004	.WORD	000004	
3088	003210	000010	.WORD	000010	
3089	003212	000020	.WORD	000020	
3090	003214	000040	.WORD	000040	
3091	003216	000100	.WORD	000100	
3092	003220	000200	.WORD	000200	
3093	003222	000400	.WORD	000400	
3094	003224	001000	.WORD	001000	
3095	003226	002000	.WORD	002000	
3096	003230	004000	.WORD	004000	
3097	003232	010000	.WORD	010000	
3098	003234	020000	.WORD	020000	
3099	003236	040000	.WORD	040000	
3100	003240	100000	.WORD	100000	
3101					
3102	003242	177776	.WORD	177776	; STANDARD PATTERN 10
3103	003244	177775	.WORD	177775	
3104	003246	177773	.WORD	177773	
3105	003250	177767	.WORD	177767	
3106	003252	177757	.WORD	177757	
3107	003254	177737	.WORD	177737	
3108	003256	177677	.WORD	177677	
3109	003260	177577	.WORD	177577	
3110	003262	177377	.WORD	177377	
3111	003264	176777	.WORD	176777	
3112	003266	175777	.WORD	175777	
3113	003270	173777	.WORD	173777	
3114	003272	167777	.WORD	167777	
3115	003274	157777	.WORD	157777	
3116	003276	137777	.WORD	137777	
3117	003300	077777	.WORD	077777	
3118					
3119	003302	172666	.WORD	172666	; STANDARD PATTERN 11
3120	003304	155555	.WORD	155555	
3121	003306	172666	.WORD	172666	
3122	003310	155555	.WORD	155555	
3123	003312	172666	.WORD	172666	
3124	003314	155555	.WORD	155555	
3125	003316	172666	.WORD	172666	
3126	003320	155555	.WORD	155555	
3127	003322	172666	.WORD	172666	
3128	003324	155555	.WORD	155555	
3129	003326	172666	.WORD	172666	
3130	003330	155555	.WORD	155555	
3131	003332	172666	.WORD	172666	

3132	003334	155555	.WORD	155555	
3133	003336	172666	.WORD	172666	
3134	003340	155555	.WORD	155555	
3135					
3136	003342	077777	.WORD	077777	; STANDARD PATTERN 12
3137	003344	137777	.WORD	137777	
3138	003346	157777	.WORD	157777	
3139	003350	167777	.WORD	167777	
3140	003352	173777	.WORD	173777	
3141	003354	175777	.WORD	175777	
3142	003356	176777	.WORD	176777	
3143	003360	177377	.WORD	177377	
3144	003362	177577	.WORD	177577	
3145	003364	177677	.WORD	177677	
3146	003366	177737	.WORD	177737	
3147	003370	177757	.WORD	177757	
3148	003372	177767	.WORD	177767	
3149	003374	177773	.WORD	177773	
3150	003376	177775	.WORD	177775	
3151	003400	177776	.WORD	177776	
3152					
3153	003402	153333	.WORD	153333	; STANDARD PATTERN 13
3154	003404	066667	.WORD	066667	
3155	003406	153333	.WORD	153333	
3156	003410	066667	.WORD	066667	
3157	003412	153333	.WORD	153333	
3158	003414	066667	.WORD	066667	
3159	003416	153333	.WORD	153333	
3160	003420	066667	.WORD	066667	
3161	003422	153333	.WORD	153333	
3162	003424	066667	.WORD	066667	
3163	003426	153333	.WORD	153333	
3164	003430	066667	.WORD	066667	
3165	003432	153333	.WORD	153333	
3166	003434	066667	.WORD	066667	
3167	003436	153333	.WORD	153333	
3168	003440	066667	.WORD	066667	
3169					
3170	003442	000000	.WORD	000000	; STANDARD PATTERN 14
3171	003444	177777	.WORD	177777	
3172	003446	177777	.WORD	177777	
3173	003450	177777	.WORD	177777	
3174	003452	177777	.WORD	177777	
3175	003454	177777	.WORD	177777	
3176	003456	177777	.WORD	177777	
3177	003460	177777	.WORD	177777	
3178	003462	177777	.WORD	177777	
3179	003464	177777	.WORD	177777	
3180	003466	177777	.WORD	177777	
3181	003470	177777	.WORD	177777	
3182	003472	177777	.WORD	177777	
3183	003474	177777	.WORD	177777	
3184	003476	177777	.WORD	177777	
3185	003500	177777	.WORD	177777	
3186					
3187	003502	177777	.WORD	177777	; STANDARD PATTERN 15

K05

CZRM880 RM03/2 PERF EXER
CZRM88.P11 21-NOV-77 15.34

MACY11 30(1046) 22-NOV-77 10:06 PAGE 63
DATA PATTERNS

SEQ 0062

3188	003504	000000	.WORD	000000
3189	003506	000000	.WORD	000000
3190	003510	000000	.WORD	000000
3191	003512	000000	.WORD	000000
3192	003514	000000	.WORD	000000
3193	003516	000000	.WORD	000000
3194	003520	000000	.WORD	000000
3195	003522	000000	.WORD	000000
3196	003524	000000	.WORD	000000
3197	003526	000000	.WORD	000000
3198	003530	000000	.WORD	000000
3199	003532	000000	.WORD	000000
3200	003534	000000	.WORD	000000
3201	003536	000000	.WORD	000000
3202	003540	000000	.WORD	000000
3203				

```

3204 .SBTTL ERROR POINTER TABLE
3205
3206 ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
3207 ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
3208 ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
3209 ;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
3210 ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
3211
3212 ;*      EM      ;;POINTS TO THE ERROR MESSAGE
3213 ;*      DH      ;;POINTS TO THE DATA HEADER
3214 ;*      DT      ;;POINTS TO THE DATA
3215 ;*      DF      ;;POINTS TO THE DATA FORMAT
3216
3217
3218 003542 $ERRTB:
3219 ;ERROR 1
3220
3221 ;ERROR 1
3222
3223 003542 046152 EM1 ;RH70 INTERRUPT OCCURRED (RMAS = 0)
3224 003544 050571 DH1
3225 003546 051222 DT1
3226 003550 000000 0
3227
3228 ;ERROR 2
3229
3230 003552 046221 EM2 ;UNEXPECTED ATTENTION OCCURRED
3231 003554 050576 DH2
3232 003556 051226 DT2
3233 003560 000000 0
3234
3235 ;ERROR 3
3236
3237 003562 046257 EM3 ;MASSBUS PARITY ERROR (MCPE=1)
3238 003564 050652 DH3
3239 003566 051244 DT3
3240 003570 000000 0
3241
3242 ;ERROR 4
3243
3244 003572 046315 EM4 ;MASSBUS PARITY ERROR (PAR=1)
3245 003574 050700 DH4
3246 003576 051254 DT4
3247 003600 000000 0
3248
3249 ;ERROR 5
3250
3251 003602 046352 EM5 ;ADDRESS PLUG BIT CHANGED
3252 003604 050576 DH2
3253 003606 051226 DT2
3254 003610 000000 0
3255
3256 ;ERROR 6
3257
3258 003612 046406 EM6
3259 003614 050737 DH6

```

3260	003616	051266			DT6	
3261	003620	000000			0	
3262						
3263						;;*****
3264						
3265						.SBTTL SETUP AND INITIALIZATION ROUTINE
3266						
3267						; START ADDRESS = 200
3268						; ADDRESS TO CHANGE RH11 UNIBUS ADDRESS = 204
3269						
3270						;;*****
3271						
3272	003622	012737	177777	001356	START:	MOV #-1,CHGADR ;SET RH11 ADDRESS CHANGE FLAG
3273	003630	000407			BR	START2 ;START THE PROGRAM
3274	003632	012737	000400	001356	START1:	MOV #400,CHGADR ;CLEAR THE RH11 ADDRESS CHANGE FLAG
3275						;;*****
3276	003640	000240			IST1:	NOP
3277	003642	012737	000001	001210	MOV	#1,\$TESTN ;SET TEST NUMBER IN APT MAIL BOX
3278	003650	000005			START2:	RESET ;CLEAR THE BUS
3279					.SBTTL	INITIALIZE THE COMMON TAGS
3280					;;CLEAR	THE COMMON TAGS (\$CMTAG) AREA
3281	003652	012706	001114		MOV	#SCMTAG,R6 ;FIRST LOCATION TO BE CLEARED
3282	003656	005026			CLR	(R6)+ ;CLEAR MEMORY LOCATION
3283	003660	022706	001154		CMP	#SWR,R6 ;;DONE?
3284	003664	001374			BNE	.-6 ;LOOP BACK IF NO
3285	003666	012706	001100		MOV	#STACK,SP ;SETUP THE STACK POINTER
3286					;;INITIALIZE	A FEW VECTORS
3287	003672	012737	031176	000030	MOV	#ERROR,\$EMTVEC ;EMT VECTOR FOR ERROR ROUTINE
3288	003700	012737	000340	000032	MOV	#340,\$EMTVEC+2 ;LEVEL 7
3289	003706	012737	034364	000034	MOV	#STRAP,\$TRAPVEC ;TRAP VECTOR FOR TRAP CALLS
3290	003714	012737	000340	000036	MOV	#340,\$TRAPVEC+2 ;LEVEL 7
3291	003722	012737	032246	000024	MOV	#SPWRON,\$PWAVEC ;POWER FAILURE VECTOR
3292	003730	012737	000340	000026	MOV	#340,\$PWAVEC+2 ;LEVEL 7
3293	003736	012737	176543	033750	MOV	#176543,\$RNUM ;PRIME THE RANDOM NUMBER GENERATOR
3294	003744	012737	123456	033752	MOV	#123456,\$LNUM ;BOTH HIGH AND LOW WORDS
3295					;;SIZE FOR A	HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
3296					;;EQUAL	TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
3297	003752	013746	000004		MOV	\$ERRVEC, -(SP) ;SAVE ERROR VECTOR
3298	003756	012737	004012	000004	MOV	#64,\$ERRVEC ;SET UP ERROR VECTOR
3299	003764	012737	177570	001154	MOV	#DSWR,SWR ;SETUP FOR A HARDWARE SWICH REGISTER
3300	003772	012737	177570	001156	MOV	#DISP,DISPLAY ;AND A HARDWARE DISPLAY REGISTER
3301	004000	022777	177777	175146	CMP	#-1,\$SWR ;TRY TO REFERENCE HARDWARE SWR
3302	004006	001012			BNE	66\$;BRANCH IF NO TIMEOUT TRAP OCCURRED
3303						AND THE HARDWARE SWR IS NOT = -1
3304	004010	000403			BR	65\$;BRANCH IF NO TIMEOUT
3305	004012	012716	004020		MOV	#65\$,(SP) ;SET UP FOR TRAP RETURN
3306	004016	000002			RTI	
3307	004020	012737	000176	001154	65\$:	MOV #SWREG,SWR ;POINT TO SOFTWARE SWR
3308	004026	012737	000174	001156	MOV	#DISPREG,DISPLAY
3309	004034	012637	000004		66\$:	MOV (SP)+,\$ERRVEC ;RESTORE ERROR VECTOR
3310						
3311	004040	005037	001212		CLR	\$PASS ;CLEAR PASS COUNT
3312	004044	132737	000200	001225	BITB	#APTSIZE,\$ENVM ;TEST USER SIZE UNDER APT
3313	004052	001403			BEQ	67\$;YES,USE NON-APT SWITCH
3314	004054	012737	001226	001154	MOV	#SSWREG,SWR ;NO,USE APT SWITCH REGISTER
3315	004062				67\$:	

3316	004062	012737	000240	000032	MOV	#240,2#EMTVEC+2	;CHANGE EMT PRIORITY TO 4
3317	004070	012737	000240	000036	MOV	#240,2#TRAPVEC+2	;CHANGE TRAP PRIORITY TO 4
3318	004076	005227	177777		INC	#-1	;FIRST START ?
3319	004102	001014			BNE	1\$	;BR IF NOT
3320	004104	104401	057512		TYPE	TITLE	;NAME AND MANDEC NUMBER
3321	004110	005737	000042		TST	42	;AUTO ACCEPT OR CHAIN MODE ?
3322	004114	001007			BNE	1\$	;BR IF EITHER
3323	004116	122737	000077	000041	CMPB	#77,41	;LOADED FROM AN RM03 ?
3324	004124	001003			BNE	1\$	;BR IF NOT
3325	004126	104401	057571		TYPE	,LOADRV	;INSTRUCT THE OPERATOR ON HOW TO TEST DRIVE 0
3326	004132	000000			HALT		
3327	004134	004737	030516		1\$: JSR	PC,STKINT	;TURN ON THE KEYBOARD INTERRUPT
3328					.SBTTL GET	VALUE FOR SOFTWARE SWITCH REGISTER	
3329	004140	005737	000042		TST	2#42	;ARE WE RUNNING UNDER XXDP/ACT?
3330	004144	001012			BNE	68\$	;BRANCH IF YES
3331	004146	123727	001224	000001	CMPB	\$ENV, #1	;ARE WE RUNNING UNDER APT?
3332	004154	001406			BEQ	68\$	;BRANCH IF YES
3333	004156	023727	001154	000176	CMP	\$SW, \$SWREG	;SOFTWARE SWITCH REG SELECTED?
3334	004164	001005			BNE	69\$	;BRANCH IF NO
3335	004166	104406			GTSWR		;GET SOFT-SWR SETTINGS
3336	004170	000403			BR	69\$	
3337	004172	112737	000001	001150	68\$: MOV	#1, \$AUTOB	;SET AUTO-MODE INDICATOR
3338	004200				69\$:		
3339	004200	105737	001224		TSTB	\$ENV	;RUN UNDER APT MODE
3340	004204	001423			BEQ	4\$	;NO, DONOT BOTHER
3341	004206	105737	001254		TSTB	\$VECT1	;NEW VECTOR ?
3342	004212	001403			BEQ	.+10	;NOT LOAD IF = 0
3343	004214	113737	001254	001272	MOV	\$VECT1, \$RMVEC	;NEW VECTOR
3344	004222	005737	001260		TST	\$BASE	;NEW BASE ADDRESS ?
3345	004226	001403			BEQ	.+10	;NO
3346	004230	013737	001260	001270	MOV	\$BASE, \$RMADR	;NEW BASE ADDRESS
3347	004236	013737	001270	034620	MOV	\$RMADR, \$RMADR	;LOAD ADDRESS INTO DRIVER
3348	004244	013737	001272	034622	MOV	\$RMVEC, \$RMVEC	;LOAD VECTOR INTO DRIVER
3349	004252	000420			BR	2\$	
3350	004254	105737	001150		4\$: TSTB	\$AUTOB	;AUTO MODE ?
3351	004260	001015			BNE	2\$	;YES
3352	004262	004737	056240		JSR	PC, \$BUSADR	;CHECK RH11 BUS ADDRESS
3353	004266	013737	001270	034620	MOV	\$RMADR, \$RMADR	;RH11 ADDRESS
3354	004274	013737	001272	034622	MOV	\$RMVEC, \$RMVEC	;RH11 VECTOR ADDRESS
3355	004302	005227	177777		INC	#-1	;FIRST START ?
3356	004306	001002			BNE	2\$	;BRANCH IF NOT
3357	004310	004737	055752		JSR	PC, \$OPRDAT	;GET THE DATE AND OPERATOR ID
3358	004314	005037	001314		2\$: CLR	\$STATIN	;CLEAR PERFORMANCE SUMMARY TYPEOUT FLAG
3359	004320	012705	001522		3\$: MOV	\$ORDERQ, \$R5	;START OF AREA TO CLEAR
3360	004324	005025			CLR	(\$R5)+	
3361	004326	022705	002102		CMP	\$BLKADR, \$R5	;LOOK FOR END OF CLEAR AREA
3362	004332	001374			BNE	3\$	;BR IF NOT FINISHED
3363	004334	012706	001100		MOV	\$STACK, \$P	;SETUP THE STACK POINTER
3364	004340	005037	177776		CLR	\$P	;CLEAR THE PROCESSOR STATUS WORD
3365	004344	013737	001312	001372	MOV	\$HZ, \$SIXTEE	;1/60 TH OR 1/50 TH SECOND COUNTER VALLE
3366	004352	005037	001364		CLR	\$HOUR	;CLEAR THE HOUR'S COUNTER
3367	004356	005037	001366		CLR	\$MINUTE	;CLEAR THE MINUTE'S COUNTER
3368	004362	005037	001370		CLR	\$SECOND	;CLEAR THE SECOND'S COUNTER
3369	004366	005037	001470		CLR	\$INTRVL+2	;CLEAR INTERVAL COUNTER
3370	004372	005037	001316		CLR	\$PACK	;CLEAR THE 'R' OR 'W' COMMAND FLAG
3371	004376	005037	001360		CLR	\$CFLAG	;CLEAR THE 'CONTROL C' FLAG

GET VALUE FOR SOFTWARE SWITCH REGISTER

```

3372 004402 042737 170000 001464 BIC #170000,MAXER ;MAKE SURE ERROR LIMITS ARE NOT TOO HIGH
3373
3374 ;ROUTINE TO DETERMINE BUFFER AREA SIZE
3375
3376 004410 005227 177777 SIZMEM: INC #1 ;SEE IF TIME TO SIZE MEMORY
3377 004414 001005 BNE 1$ ;BR IF NOT
3378 004416 004737 056142 JSR PC,$SIZE ;SEE HOW MUCH MEMORY ON SYSTEM
3379 004422 013737 056236 001354 SLSTAD,LSTAD ;SAVE THE LAST ADDRESS
3380 004430 012737 000001 001700 1$: MOV #1,BUFTBL ;LOAD NUMBER OF BUFFERS
3381 004436 012737 057512 001702 MOV #ENDPGM,BUFTBL+2 ;STARTING ADDRESS OF BUFFER
3382 004444 013737 001354 001704 MOV LSTAD,BUFTBL+4 ;LAST ADDR TO BUFFER ALLOCATION TABLE
3383 004452 023727 001354 160000 CMP LSTAD,#160000 ;OVER 28K ?
3384 004460 101403 BLOS 2$ ;NO
3385 004462 012737 160000 001704 MOV #160000,BUFTBL+4 ;XXDP MAX MEMORY 28K
3386 004470 162737 057512 001704 2$: SUB #ENDPGM,BUFTBL+4 ;SUBTRACT PROGRAM SPACE
3387 004476 000241 CLC ;CLEAR THE 'C' BIT
3388 004500 006037 001704 ROR BUFTBL+4 ;CONVERT TO WORD COUNT
3389 004504 162737 000144 001704 SUB #100.,BUFTBL+4 ;SAVE ROOM FOR THE 'ABS' LOADER
3390 004512 005737 000042 TST 42 ;LOAD FROM XXDP OR OTHER MONITOR ?
3391 004516 001403 BEQ 3$ ;BR IF LOADED BY PAPER TAPE
3392 004520 162737 002570 001704 SUB #1400.,BUFTBL+4 ;SUBTRACT 'XXDP' LOADER SIZE
3393 004526 005737 001462 3$: TST MAXDL ;VALUE IN 'MAXDL' ?
3394 004532 001003 BNE 4$ ;BR IF VALUE IS
3395 004534 012737 020000 001462 MOV #8192.,MAXDL ;ASSUME FULL TRACK MAXIMUM
3396 004542 023737 001462 001704 4$: CMP MAXDL,BUFTBL+4 ;IS THAT TOO LARGE ?
3397 004550 103403 BLO 5$ ;BR IF NOT
3398 004552 013737 001704 001462 MOV BUFTBL+4,MAXDL ;USE MAX AVAIL MEMORY AS MAX BUFFER SIZE
3399 004560 013737 001704 054716 5$: MOV BUFTBL+4,PARLST+2 ;VALUE FOR THE PARAMETER TABLE
3400
3401 ;SEE IF THE OPERATOR WANTS TO CHANGE ANY PARAMETERS
3402
3403 004566 105737 001150 LKPAR: TSTB $AUTOB ;'XXDP' CHAIN MODE OR 'ACT11' OPERATION ?
3404 004572 001033 BNE SETVEC ;BR IF YES
3405 004574 022737 007070 001264 CMP #7070,$CDW1 ;FROM THE POWER FAIL RT. ?
3406 004602 001427 BEQ SETVEC ;BRANCH IF SO
3407 004604 105737 001224 TSTB $ENV ;APT STAND ALONE MODE ?
3408 004610 001024 BNE SETVEC ;NO
3409 004612 104401 001201 TYPE , $CRLF
3410 004616 104401 055012 TYPE ,ASKPAR ;ASK FOR PARAMETERS
3411 004622 104411 RDLIN ;READ THE ENTRY
3412 004624 012605 MOV (SP)+,R5 ;ADDRESS OF ENTRY TO R5
3413 004626 122715 000131 CMPB #Y,(R5) ;WAS ENTRY A 'Y' (YES)
3414 004632 001013 BNE SETVEC ;BR IF NOT 'Y'
3415 004634 012703 054714 ENTPR: MOV #PARLST,R3 ;PARAMETER TABLE ADDRESS
3416 004640 004737 026610 JSR PC,PARENT ;GET THE PARAMETER ENTRY
3417 004644 023727 001462 000004 CMP MAXDL,#4 ;IS THE 'MAXDL' VALUE OK ?
3418 004652 103003 BHIS SETVEC ;BR IF IT IS
3419 004654 012737 000004 001462 MOV #4,MAXDL ;SET 'MAXDL' TO THE MINIMUM VALUE
3420
3421 ;DISPLAY DRIVE STATUS AND SET UP THE OTHER SYSTEM DEVICES THAT
3422 ; THE PROGRAM WILL USE
3423
3424 004662 004737 022632 SETVEC: JSR PC,CKCLK ;START THE CLOCK
3425 004666 004737 034636 JSR PC,RMINIT ;INITIALIZE THE RM03 DRIVER
3426 004672 012737 177777 034560 MOV #1,SAVEFG ;SET THE SAVE REGISTERS FLAG
3427 004700 062727 177777 000000 ADD #1,#0 ;CHECK FOR FIRST START

```

3428	004706	103004			BCC	11\$	;BR IF FIRST START
3429	004710	032777	000004	174236	BIT	#SW02,2SWR	;TYPEOUT THE DRIVE STATUS TABLE ?
3430	004716	001071			BNE	10\$	;BR IF NOT
3431	004720	005004			11\$: CLR	R4	;DRIVE TABLE POINTER
3432	004722	104401	001201		TYPE	,\$CRLF	;CR-LF
3433	004726	104401	053472		TYPE	,SYSTAT	;TYPE STATUS HEADING
3434	004732				1\$:		
3435	004732	010446			MOV	R4,-(SP)	;SAVE R4 FOR TYPEOUT
3436							;TYPE DRIVE NUMBER
3437	004734	104403			TYPOS		;GO TYPE--OCTAL ASCII
3438	004736	002			.BYTE	2	;TYPE 2 DIGIT(S)
3439	004737	000			.BYTE	0	;SUPPRESS LEADING ZEROS
3440	004740	104401	053244		TYPE	,LIN4SP	;SPACES
3441	004744	105764	034472		TSTB	DRVSTA(R4)	;CHECK DRIVE'S STATUS
3442	004750	100416			BMI	4\$	;BR IF UNSAFE
3443	004752	001020			BNE	5\$	;BR IF ONLINE
3444	004754	105764	034502		TSTB	DRVTYP(R4)	;SEE IF OFFLINE OR NONEXISTENT
3445	004760	001404			BEQ	2\$	;BR IF NONEXISTENT
3446	004762	100006			BPL	3\$	;BR IF OFFLINE
3447	004764	104401	053404		12\$: TYPE	,NOTRM	;DRIVE NOT AN RM03
3448	004770	000436			BR	4\$	;CHECK NEXT DRIVE
3449	004772	104401	053426		2\$: TYPE	,NOTPRS	;DRIVE NOT PRESENT
3450	004776	000433			BR	4\$	;CHECK NEXT DRIVE
3451	005000	104401	053313		3\$: TYPE	,UNTOFF	;DRIVE OFFLINE
3452	005004	000405			BR	6\$	;PRINT DRIVE TYPE
3453	005006	104401	053462		4\$: TYPE	,NOTSAF	;DRIVE UNSAFE
3454	005012	000402			BR	6\$	;PRINT DRIVE TYPE
3455	005014	104401	053324		5\$: TYPE	,UNTON	;DRIVE ONLINE
3456	005020	104401	053246		6\$: TYPE	,LINSP	;SPACES
3457	005024	012737	053512	005064	MOV	,RM03A,8\$	;ADDRESS OF RM03 MESSAGE
3458	005032	122764	000004	034502	CMPB	,#4,DRVTYP(R4)	;RM03 ?
3459	005040	001410			BEQ	7\$	;BRANCH IF SO
3460	005042	012737	053517	005064	MOV	,RM03B,8\$	;RM02 MESSAGE
3461	005050	122764	000005	034502	CMPB	,#5,DRVTYP(R4)	;RM02 DRIVE ?
3462	005056	001401			BEQ	7\$	;BRANCH IF SO
3463	005060	000741			BR	12\$	;OTHERWISE EXIT
3464	005062	104401			7\$: TYPE		;TYPE THE DRIVE TYPE MESSAGE
3465	005064	000000			8\$: .WORD	0	;MESSAGE ADDRESS HERE
3466	005066	104401	001201		9\$: TYPE	, \$CRLF	;CR-LF
3467	005072	005204			INC	R4	;INCREMENT DRIVE NUMBER/TABLE POINTER
3468	005074	020427	000010		CMP	R4,#8.	;FINISHED ?
3469	005100	001314			BNE	1\$	;BR IF NOT
3470	005102	104401	001201		10\$: TYPE	, \$CRLF	;CR-LF
3471	005106	005037	177776		CLR	PS	;SET PRIORITY BACK TO '0'
3472	005112	012706	001100		MOV	,STACK,SP	;SET UP STACK POINT
3473							
3474							;SETUP IF 'XXDP' OR 'ACT11' OPERATION
3475							
3476	005116	005737	000042		MONTR: TST	42	; 'XXDP' CHAIN MODE OR 'ACT11' AUTO ACCEPT
3477	005122	001405			BEQ	1\$	;BR IF NEITHER
3478	005124	005737	001356		TST	CHGADR	;200 START ?
3479	005130	003002			BGT	1\$	;YES
3480	005132	004737	024544		JSR	PC,ASGN2	;ASSIGN DRIVES
3481	005136	005227	177777		1\$: INC	,#-1	;FIRST START ?
3482	005142	001011			BNE	2\$	;BR IF NOT
3483	005144	105737	000041		TSTB	,#41	;LOADED FROM PAPER TAPE ?


```

3484 005150 001406      BEQ      2$      ;BR IF YES
3485 005152 023727 001354 100000    CMP      LSTAD,#100000 ;MORE THAN 16K ON THE SYSTEM ?
3486 005160 103002      BHS      2$      ;BR IF YES
3487 005162 104401 057770    TYPE      ,NOLOAD ;TELL THE OPERATOR THAT THE 'XXDP' LOADER
3488                                     ;WILL BE OVERWRITTEN
3489 005166 004737 030516      2$:      JSR      PC,STKINT ;INITIALIZE THE KEYBOARD INTERRUPT HANDLER
3490                                     ;200 START
3491 005172 105737 001150  FOWT1:  TSTB     $AUTOB ;AUTO MODE ?
3492 005176 001412      BEQ      1$      ;NO
3493 005200 012737 000001 001500    MOV      #1,RATIO ;SPEED UP TEST
3494 005206 012737 077777 001450    MOV      #77777,ENDCON ;SET LSW OF ENDING VALUE
3495 005214 012737 000027 001452    MOV      #27,ENDCON+2 ;SET MSW OF ENDING VALUE
3496 005222 000403      BR      FOWT2 ;START TO WRITE AND TEST
3497 005224 005737 001356      1$:      TST      CHGADR ;START AT 200 ?
3498 005230 003467      BLE      FOWT3 ;NO
3499 005232 005001      FOWT2:  CLR      R1 ;DRIVE #
3500 005234 005002      CLR      R2 ;AVAIL TABLE INDEX
3501 005236 005003      CLR      R3 ;DRIVE# X 2
3502 005240 122737 000024 000041    CMPB     #24,41 ;LOAD FROM DRIVE RM03
3503 005246 001002      BNE      1$      ;NO
3504 005250 005201      INC      R1 ;START FROM DRIVE 1
3505 005252 005723      TST      (R3)+ ;ADJUST INDEX
3506 005254 105761 034472      1$:      TSTB     DRVSTA(R1) ;DRIVE ON LINE ?
3507 005260 003442      BLE      2$      ;NO
3508 005262 016300      MOV      BLKADR(R3),R0 ;LOAD DPB ADDRESS
3509 005266 004737 025432      JSR      PC,CLRDPB ;CLEAR DPB BLOCK
3510 005272 004537 026154      JSR      R5,GETADR ;RETRIEVE BAD SPOT FILE
3511 005276 000433      BR      2$      ;ERROR RET FROM GETADR RT.
3512 005300 010062 001570      MOV      R0,NEWUNT(R2) ;LOAD DPB ADDRESS TO ABAIL QUEJE
3513 005304 105060 000026      CLRB     $PACK(R0) ;TEST PACK FLAG
3514 005310 105737 001150      TSTB     $AUTOB ;IN AUTO MODE ?
3515 005314 001403      BEQ      .+8. ;BRANCH IF SO
3516 005316 112760 177776 000026  MOVB     #-2,$PACK(R0) ;WRITE PATTERN THEN TEST
3517 005324 013760 001446 000106    MOV      CYLMT,MAXCYL(R0) ;UP CYLINDER LIMIT
3518 005332 013760 001444 000112    MOV      TRKMT,MAXTRK(R0) ;UPPER TRACK LIMIT
3519 005340 013760 001442 000116    MOV      SECLMT,MAXSEC(R0) ;UPPER SECTOR LIMIT
3520 005346 005060 000110      CLR      MINCYL(R0) ;CLEAR LOWER LIMIT
3521 005352 005060 000114      CLR      MINTRK(R0) ;CLEAR LOWER LIMIT
3522 005356 005060 000120      CLR      MINSEC(R0) ;CLEAR LOWER LIMIT
3523 005362 004737 017704      JSR      PC,WRTPK ;SET UP PARAMETERS
3524 005366 022322      2$:      CMP      (R3)+,(R2)+ ;INCREMENT INDEX
3525 005370 005201      INC      R1 ;NEXT DRIVE
3526 005372 022701 000007      CMP      #7,R1 ;ALL DRIVE ASSIGN ?
3527 005376 002326      BGE      1$      ;NO
3528 005400 005037 001316      CLR      PACK ;TEST PACK FLAG
3529 005404 000137 005574      JMP      MAIN1 ;JUMP TO WAIT PARAMETER AND BUFFER LOOP
3530 005410 104401 054462      FOWT3:  TYPE     ,INTDON ;TYPE 'INITIALIZE COMPLETE'
3531 005414 000137 005574      JMP      MAIN1 ;START THE PROGRAM
3532                                     ;$DOAGN:  JMP      START ;START AGAIN
3533
3534                                     ;*****
3535
3536 .SBTTL MAIN PROGRAM
3537
3538                                     ;*****
3539

```

```

3540 005420 005737 001360      MAIN:  TST      CFLAG      ;KEYBOARD INTERRUPTED ?
3541 005424 001407              BEQ      3$          ;BRANCH IF NOT
3542 005426 005737 001522      1$:  TST      ORDERQ     ;DON'T DEASSIGN ANY DRIVE, IF SYSTEM NOT IDLE
3543 005432 001402              BEQ      2$          ;
3544 005434 000137 006206              JMP      IDLE      ;LET ALL DRIVE FINISH ODDER
3545 005440 004737 024130      2$:  JSR      PC,KSR     ;SERVICE THE KEYBOARD
3546 005444 000240      3$:  NOP              ;EXIT
3547 005446 012703 000010      MAINDA: MOV      #8.,R3     ;DRIVE COUNTER
3548 005452 012705 001546              MOV      #DUNIT,R5   ;ADDRESS OF 'DROP DRIVE' TABLE
3549 005456 005715      1$:  TST      (R5)          ;SEE IF ENTRY AT PRESENT POSITION
3550 005460 001011              BNE      3$          ;BR IF THERE IS ONE
3551 005462 062705 000002      2$:  ADD      #2,R5       ;INCREMENT TO NEXT TABLE POSITION
3552 005466 005303              DEC      R3           ;DECREMENT DRIVE COUNTER
3553 005470 001372              BNE      1$          ;BR IF MORE TO CHECK
3554 005472 105737 001544              TSTB     ASNLST     ;ANY DRIVES ACTIVE ?
3555 005476 001036              BNE      MAIN1        ;BR IF YES
3556 005500 000137 027440              JMP      $GET42    ;CHECK FOR MONITOR RETURN
3557 005504 012701 001612      3$:  MOV      #AVAIL,R1    ;ADDRESS OF 'AVAILABLE DRIVES' TABLE
3558 005510 005711      4$:  TST      (R1)          ;SEE IF AT END OF TABLE
3559 005512 001405              BEQ      5$          ;BR IF AT END: GO CHECK 'WAIT' TABLE
3560 005514 021115              CMP      (R1),(R5)       ;IS DRIVE IN 'AVAIL' THE ONE TO BE DROPPED
3561 005516 001414              BEQ      7$          ;BR IF YES
3562 005520 062701 000002              ADD      #2,R1       ;INCREMENT 'AVAIL' TABLE ADDRESS
3563 005524 000771              BR      4$          ;CONTINUE LOOKING
3564 005526 012701 001634      5$:  MOV      #WAIT,R1    ;MOVE THE ADDRESS OF THE BUFFER WAIT TABLE
3565 005532 005711      6$:  TST      (R1)          ;AT THE END OF THE 'WAIT' TABLE ?
3566 005534 001752              BEQ      2$          ;BR IF YES: SEE IF ANY MORE 'DROP' REQUESTS
3567 005536 021115              CMP      (R1),(R5)       ;DRIVE IN THE 'WAIT' TABLE ?
3568 005540 001403              BEQ      7$          ;BR IF IT IS
3569 005542 062701 000002              ADD      #2,R1       ;INCREMENT 'WAIT' TABLE ADDRESS
3570 005546 000771      6$:  BR      6$          ;CONTINUE LOOK THROUGH THE 'WAIT' TABLE
3571 005550 011100      7$:  MOV      (R1),R0        ;PUT THE DRIVE'S BLOCK ADDRESS IN R0
3572 005552 104401 053771              TYPE     DEASSG    ;TYPE 'DRIVE DEASSIGNED'
3573 005556 004737 023122              JSR      PC,TYPEST ;TYPE THE DRIVE'S PERFORMANCE SUMMARY
3574 005562 005015              CLR      (R5)        ;CLEAR THE 'DROP DRIVE' TABLE ENTRY
3575 005564 005011              CLR      (R1)        ;REMOVE THE DRIVE FROM THE 'AVAIL' OR 'WAIT' TABLE
3576 005566 004737 017666              JSR      PC,COMPRES ;COMPRESS THE RESPECTIVE TABLE
3577 005572 000733              BR      2$          ;SEE IF ANY MORE DRIVES
3578
3579      ;LOOK FOR DRIVES TO BE ASSIGNED
3580
3581 005574 012703 000010      MAIN1:  MOV      #8.,R3     ;DRIVE COUNT
3582 005600 005002              CLR      R2          ;'AVAIL' INDEX
3583 005602 005004              CLR      R4          ;ASSIGN LIST INDEX
3584 005604 005005              CLR      R5          ;NEW DRIVE INDEX
3585 005606 005765 001570      1$:  TST      NEWUNT(R5) ;NEW DRIVE IN THIS POSITION
3586 005612 001006              BNE      3$          ;BR IF THERE IS
3587 005614 062705 000002      2$:  ADD      #2,R5       ;INCREMENT R5
3588 005620 005204              INC      R4          ;INCREMENT ASSIGN INDEX
3589 005622 005303              DEC      R3          ;DECREMENT DRIVE COUNT
3590 005624 001370              BNE      1$          ;BR IF MORE DRIVES
3591 005626 000430              BR      MAIN2    ;START OPERATIONS FOR THE AVAILABLE DRIVES
3592 005630 104401 053305      3$:  TYPE     UNTMSG    ;'DRIVE'
3593 005634 010446              MOV      R4,-(SP)   ;SAVE R4 FOR TYPEOUT
3594
3595 005636 104403              TYPEOS  ;TYPE DRIVE NUMBER
;GO TYPE--OCTAL ASCII

```


3596	005640	002		.BYTE	2		::TYPE 2 DIGIT(S)
3597	005641	000		.BYTE	0		::SUPPRESS LEADING ZEROS
3598	005642	104401	054041	TYPE	ASGND		::STARTED
3599	005646	005762	001612	4\$:	TST	AVAIL(R2)	::AT END OF AVAILABLE TABLE
3600	005652	001403			BEQ	5\$	::BR IF YES
3601	005654	062702	000002		ADD	#2,R2	::INCREMENT AVAILABLE TABLE INDEX
3602	005660	000772			BR	4\$	::CONTINUE LOOKING FOR END OF TABLE
3603	005662	016562	001570	001612	5\$:	MOV	NEWUNT(R5),AVAIL(R2) ;MOVE ADDR OF DRIVE INTO AVAIL LST
3604	005670	005065	001570		CLR	NEWUNT(R5)	::TAKE DRIVE OUT OF NEW DRIVE TABLE
3605	005674	156437	034606	001544	BISB	ATABIT(R4),ASNLS	::SET DRIVE ASSIGNED INDICATOR
3606	005702	062702	000002		ADD	#2,R2	::INCREMENT AVAILABLE TABLE POINTER
3607	005706	000742			BR	2\$	::LOOK FOR MORE DRIVES
3608							
3609							
3610							::GET PARAMETERS, BUFFER SPACE, AND START ORDERS FOR DRIVES IN
3611							::THE 'AVAILABLE' QUEUE
3612	005710						
3613							
3614							
3615							
3616							
3617	005710	005002					
3618	005712	005762	001634	1\$:	CLR	R2	
3619	005716	001437			TST	WAIT(R2)	
3620	005720	016200	001634		BEQ	MAIN3	
3621	005724	005046			MOV	WAIT(R2),R0	
3622	005726	004737	015630		CLR	-(SP)	
3623	005732	012660	000006		JSR	PC,GETBUF	
3624	005736	001421			MOV	(SP)+,\$BUF(R0)	
3625	005740	005060	000072		BEQ	2\$	
3626	005744	004737	016214		CLR	\$FAIR(R0)	
3627	005750	004737	016342		JSR	PC,FILBUF	
3628	005754	012705	001522		JSR	PC,GODRIV	
3629	005760	005725			MOV	ORDERQ,R5	
3630	005762	001376			TST	(R5)+	
3631	005764	010045			BNE	-2	
3632	005766	012701	001634		MOV	R0-(R5)	
3633	005772	060201			MOV	#WAIT,R1	
3634	005774	004737	017666		ADD	R2,R1	
3635	006000	000744			JSR	PC,COMPRES	
3636	006002	062702	000002	2\$:	BR	1\$	
3637	006006	000741			ADD	#2,R2	
3638	006010	000240			BR	1\$	
3639	006012	000240			NOP		
3640	006014	000240			NOP		
3641	006016				NOP		
3642							
3643							
3644	006016	005002					
3645	006020	005762	001612	1\$:	TST	ORDERQ	
3646	006024	001002			BNE	IDLE	
3647	006026	000137	006206		CLR	R2	
3648	006032	016200	001612		TST	AVAIL(R2)	
3649	006036	005760	000104		BNE	+6	
3650	006042	001010			JMP	IDLE	
3651	006044	105760	000026		MOV	AVAIL(R2),R0	
					TST	\$NEXT(R0)	
					BNE	6\$	
					TSTB	\$PACK(R0)	

```

SET UP THE WAIT QUEUE FOR
ALL THE DRIVES THAT WAITING
FOR BUFFER IN THREE TIMES OF
SEARCHING
; START FROM THE FIRST LOCATION
; WAIT FOR THE BUFFER ?
; EXIT IF NONE
; LOAD R0 WITH THE DPB ADDRESS
; CLEAR THE STACK FOR BUFFER REQ
; CALL TO GET THE BUFFER RT.
; IF 0 BUFFER IS STILL NOT AVAILABLE
; BRANCH IF NO BUFFER AVAILABLE
; CLEAR THE FAIR FLAG
; FILL THE BUFFER
; SET COMMAND AND GO
; PUT THE WAIT QUEUE INTO ORDER QUEUE
; QUEUE AVAILABLE ?
; BRANCH IF NOT
; LOAD THE DPB ADDRESS INTO THE ORDER QUEUE
; REMOVE THE QUEUE FROM THE WAITING QUEUE
; OFFSET THE QUEUE POSITION
; COMPRESS THE QUEUE
; BRANCH IF DONE
; CHECK THE NEXT QUEUE
; LOOPING BACK
; OUTSTANDING BUFFER REQUESTS
; BR IF THERE ARE
; CLEAR DRIVE TABLE POINTER
; ANY DRIVES WAITING FOR PARAMETERS
; BRANCH IF ANY
; BRANCH IF NONE
; CONTROL BLOCK ADDR IN R0
; PARAMETERS BEEN SELECTED ?
; BR IF THEY HAVE
; 'R' OR 'W' COMMAND FOR THE DRIVE ?

```

3652	006050	001403		BEQ	5\$	: BR IF NOT
3653	006052	004737	017704	JSR	PC, WATPK	: GET DATA PACK PARAMETERS
3654	006056	000404		BR	7\$	: GET THE BUFFER
3655	006060	004737	016420	5\$: JSR	PC, SELPAR	: SELECT THE PARAMETERS
3656	006064	004737	017374	6\$: JSR	PC, GETPAR	: LOAD NEW PARAMETERS
3657	006070	005046		7\$: CLR	-(SP)	: MAKE ROOM ON THE STACK FOR THE BUFFER ADDR
3658	006072	004737	015630	JSR	PC, GETBUF	: GET BUFFER
3659	006076	012660	000006	MOV	(SP)+, \$BUF(R0)	: MOVE BUFFER ADDR TO DPB
3660	006102	001414		BEQ	8\$	: BR IF '0' ADDR (NO BUFFER)
3661	006104	004737	016214	JSR	PC, FILBUF	: FILL THE BUFFER
3662	006110	005060	000072	CLR	\$FAIR(R0)	: CLEAR THE 'FAIRNESS' COUNT
3663	006114	004737	016342	JSR	PC, GODRIV	: PUT CURRENT DPB IN DRIVER
3664	006120	012705	001522	MOV	#ORDERQ, R5	: ADDRESS OF ORDER QUEUE IN R5
3665	006124	005725		TST	(R5)+	: END OF QUEUE ?
3666	006126	001376		BNE	.-2	: BR IF NOT
3667	006130	010045		MOV	R0, -(R5)	: PUT BLOCK ADDRESS INTO QUEUE
3668	006132	000417		BR	10\$	: CONTINUE LOOKING
3669	006134					
3670	006134	005260	000072	8\$: INC	\$FAIR(R0)	: INCREMENT THE FAIR COUNT
3671	006140	022760	000003 000072	CMP	#3, \$FAIR(R0)	: THREE TIMES BUFFER IS NOT AVAILABLE ?
3672	006146	101006		BHI	9\$	: BRANCH IF NOT OVER THREE TIMES
3673	006150	012705	001634	MOV	#WAIT, R5	: LOAD INTO THE WAIT QUEUE
3674	006154	005725		TST	(R5)+	: AN AVAILABLE LOCATION ?
3675	006156	001376		BNE	.-2	: BRANCH IF NOT
3676	006160	010045		MOV	R0, -(R5)	: LOAD INTO WAIT QUE
3677	006162	000403		BR	10\$	: REMOVE THE DPB FROM AVAILABLE QUE
3678	006164					
3679	006164	062702	000002	9\$: ADD	#2, R2	: INCREMENT INDEX
3680	006170	000713		BR	1\$	: BRANCH BACK TO FIRE NEXT DRIVE
3681	006172	012701	001612	10\$: MOV	#AVAIL, R1	: 'AVAILABLE' TABLE ADDRESS
3682	006176	060201		ADD	R2, R1	: FORM ADDRESS OF LAST ENTRY
3683	006200	004737	017666	JSR	PC, CMPRES	: COMPRESS THE TABLE
3684	006204	000705		BR	1\$	: CONTINUE LOOKING
3685						
3686						: GET BUFFER ASSIGNMENTS FOR DRIVES IN THE 'BUFFER WAIT' QUEUE
3687						
3688						
3689						: WAIT FOR AN ORDER TO FINISH
3690						
3691	006206	012701	001522	IDLE: MOV	#ORDERQ, R1	: ADDRESS OF THE ORDER QUEUE IN R1
3692	006212	012100		1\$: MOV	(R1)+, R0	: PUT BLOCK ADDRESS INTO R0
3693	006214	001433		BEQ	IDLE1	: BR IF END OF QUEUE
3694	006216	005760	000016	TST	\$STATUS(R0)	: SEE IF DRIVE FINISHED
3695	006222	001773		BEQ	1\$	: BR IF DRIVE NOT FINISHED
3696	006224	162701	000002	SUB	#2, R1	: CORRECT THE QUEUE POINTER
3697	006230	010146		MOV	R1, -(SP)	: SAVE THE QUEUE ADDRESS
3698	006232	004737	015472	JSR	PC, STATIS	: ACCUMULATE STATISTICS FOR DRIVE IN R0
3699	006236	000240		NOP		: DEBUGGING AID
3700	006240	004737	006460	JSR	PC, PROCES	: PROCESS END OF ORDER
3701	006244	005037	001362	CLR	BADSEC	: CLEAR THE BAD TRK/SEC ERROR INDICATOR
3702	006250	004737	027064	JSR	PC, ABNML	: SEE IF ANY DRIVES HAVE TOO MANY ERRORS
3703	006254	004737	027112	JSR	PC, EOP	: SEE IF ANY DRIVE HAS XFERED 3X10 ¹⁹ BITS
3704	006260	012601		MOV	(SP)+, R1	: RESTORE THE ORDER TABLE INDEX
3705	006262	012705	001612	MOV	#AVAIL, R5	: FIND THE END OF THE 'AVAILABLE' TABLE
3706	006266	005725		2\$: TST	(R5)+	: END OF THE TABLE ?
3707	006270	001376		BNE	2\$	: BR IF NOT AT END OF LIST

```

3708 006272 011145      MOV      (R1),-(R5)      ;MOVE THE BLOCK ADDRESS INTO THE TABLE
3709 006274 004737 017666 JSR      PC,CMPRES      ;COMPRESS THE ORDER QUEUE
3710 006300 004737 015764 JSR      PC,RELBUF      ;RESTORE BUFFER
3711 006304      IDLE1:
3712 006304 032777 000004 172642 1$: BIT      #SW02,ASWR      ;TYPE PERFORMANCE SUMMARY
3713 006312 001007      BNE      2$      ;BR IF NOT
3714 006314 005737 001314      TST      STATIN      ;TIME TO TYPE THE PERFORMANCE SUMMARY ?
3715 006320 001404      BEQ      2$      ;BR IF NOT
3716 006322 005037 001314      CLR      STATIN      ;CLEAR THE INDICATOR
3717 006326 004737 023034      JSR      PC,STATPR      ;TYPE THE SUMMARY
3718 006332 000137 005420      2$: JMP      MAIN      ;CONTINUE THE LOOP
3719
3720      ;SETUP TO REFORMAT AN ERROR SECTOR
3721
3722 006336 032777 000001 172610 REFM1: BIT      #SW0,ASWR      ;READ ONLY SWITCH SET ?
3723 006344 001044      BNE      REFM1X      ;BR IF IT IS
3724 006346 032777 000200 172600      BIT      #SW7,ASWR      ;SWITCH 7 SET ?
3725 006354 001040      BNE      REFM1X      ;BR IF IT IS
3726 006356 005737 001474      TST      FORMAT      ;WRITE HEADER & DATA ORDERS ALLOWED ?
3727 006362 001435      BEQ      REFM1X      ;BR IF NOT
3728 006364 022760 001465 000370      CMP      #821,$RMDCL(R0) ;LEGAL VALUE ?
3729 006372 101431      BLOS      REFM1X      ;NO, NOT FORMAT
3730 006374 004737 022502      JSR      PC,READDR      ;GET CORRECTED SECTOR-TRACK ADDRESSES
3731 006400 012660 000100      MOV      (SP)+,$NCRYL(R0) ;CYLINDER
3732 006404 112660 000077      MOV      (SP)+,$NTRK(R0) ;TRACK ADDR TO DPB
3733 006410 112660 000076      MOV      (SP)+,$NSEC(R0) ;SECTOR ADDR TO DPB
3734 006414 012760 000402 000102      MOV      #258,$NWRDL(R0) ;WORD COUNT FOR FORMAT
3735 006422 112760 000003 000074 1$: MOV      #3,$NCODE(R0) ;COMMAND CODE
3736 006430 004537 017172      JSR      RS,CHKADR      ;AVOID REFORMAT BAD SPOT
3737 006434 000401      BR      2$      ;BRANCH IF NOT ON BAD SPOT
3738 006436 000407      BR      REFM1X      ;BRANCH IF ON BAD SPOT
3739 006440 004737 017332      2$: JSR      PC,GETPAT      ;GET A PATTERN
3740 006444 110560 000075      MOV      RS,$NPATC(R0) ;PATTERN CODE TO CONTROL BLOCK
3741 006450 012760 177777 000104      MOV      #-1,$NEXT(R0) ;SET PARAMETERS SELECTED INDICATOR
3742 006456 000207      REFM1X: RTS      PC      ;RETURN
3743
3744      ;PROCESS THE ORDER TERMINATION
3745
3746 006460 111037 001344      PROCES: MOV      (R0),UNIT      ;DRIVE NUMBER FOR ANY ERROR MESSAGES
3747 006464 005760 000016      TST      $STATUS(R0) ;SEE IF DRIVER SIGNALLED AN ERROR
3748 006470 100427      BMI      ERPROC      ;BR IF ERROR
3749 006472 032760 100000 000234      BIT      #BIT15,$RMCS1(R0) ;SEE IF 'SC' SET
3750 006500 001410      BEQ      1$      ;BR IF NOT SET
3751 006502 032760 040000 000234      BIT      #BIT14,$RMCS1(R0) ;SEE IF 'TRE' SET
3752 006510 001017      BNE      ERPROC      ;BR IF SET
3753 006512 032760 040000 000246      BIT      #BIT14,$RMD5(R0) ;SEE IF 'ERR' SET
3754 006520 001013      BNE      ERPROC      ;BR IF SET
3755 006522 004737 012640      1$: JSR      PC,CKERR      ;NO ERROR, CHECK ERROR BITS ANYWAY
3756 006526 004737 012732      JSR      PC,CKBUS      ;NO ERROR, CHECK BUS ADDR & WC
3757 006532 032777 000002 172414      BIT      #SW01,ASWR      ;DATA COMPARE ?
3758 006540 001002      BNE      2$      ;BR IF NOT
3759 006542 004737 013016      JSR      PC,CMPAR      ;NO ERROR, COMPARE DATA
3760 006546 000207      2$: RTS      PC      ;RETURN
3761
3762      ;ORDER TERMINATED WITH AN ERROR - PROCESS THE ERROR
3763

```

```

3764 006550 032760 000200 000016 ERPROC: BIT      #BIT07,STATUS(RO) :DONE BIT SET ?
3765 006556 001402          BEQ      ERPRC1      ;BR IF ORDER DIDN'T COMPLETE NORMALLY
3766 006560 000137 007204          JMP      DONE      ;PROCESS ERROR WITH 'DONE' BIT SET
3767
3768          ;PROCESS ORDER COMPLETION WITH 'ERROR' & 'DONE NOT' BITS
3769
3770 006564 032760 010000 000016 ERPRC1: BIT      #BIT12,STATUS(RO) :SEE IF DRIVE WAS UNSAFE
3771 006572 001025          BNE      PUNSAF      ;BR IF YES
3772 006574 032760 004000 000016          BIT      #BIT11,STATUS(RO) :PARITY ERROR OCCURRED
3773 006602 001055          BNE      UCPAR      ;BR IF IT DID
3774 006604 032760 002000 000016          BIT      #BIT10,STATUS(RO) :FATAL PARITY ERROR?
3775 006612 001056          BNE      FALPAR      ;BR IF THERE IS ONE
3776 006614 032760 001000 000016          BIT      #BIT09,STATUS(RO) :TIMEOUT?
3777 006622 001076          BNE      SWTIM      ;BR IF YES
3778 006624 032760 040002 000016          BIT      #BIT14:BIT01,STATUS(RO) :DRIVE WENT OFFLINE ?
3779 006632 001111          BNE      OFLIN      ;BR IF IT DID
3780 006634 032760 000004 000016          BIT      #BIT2,STATUS(RO) :PORT REQUEST TIME OUT ?
3781 006642 001141          BNE      PRTIM      ;BR IF IT DID
3782 006644 000207          RTS      PC      ;ERROR. RETURN
3783
3784          ;DRIVE IS PERSISTENTLY UNSAFE
3785
3786 006646 104401 001201          PUNSAF: TYPE      , $CRLF      ;CR-LF
3787 006652 104401 046552          TYPE      , EM12      ;'DRIVE UNSAFE' MESSAGE
3788 006656 104401 054014          TYPE      , DRNUM      ;DRIVE NUMBER
3789 006662 013746 001344          MOV      UNIT,-(SP) ;SAVE UNIT FOR TYPEOUT
3790          ;TYPE DRIVE NUMBER
3791 006666 104403          TYPOS      ;GO TYPE--OCTAL ASCII
3792 006670 002          .BYTE      2      ;TYPE 2 DIGIT(S)
3793 006671 000          .BYTE      0      ;SUPPRESS LEADING ZEROS
3794 006672 104401 001201          TYPE      , $CRLF      ;CR-LF
3795 006676 004737 020406          JSR      PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
3796 006702 104414 046552          DISPLY    EM12      ;PERSISTENT DEVICE UNSAFE MESSAGE
3797 006706 004737 020462          JSR      PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
3798 006712 004737 021070          JSR      PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
3799 006716 004737 021540          JSR      PC,LINE4 ;PRINT LINE 4 OF THE ERROR MESSAGE
3800 006722 004737 023644          JSR      PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
3801 006726 004737 022150          JSR      PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
3802 006732 000137 026764          JMP      DROP      ;DROP THE DRIVE
3803
3804          ;UNCORRECTABLE MASSBUS PARITY ERROR OCCURRED
3805
3806 006736 104401 001201          UCPAR: TYPE      , $CRLF      ;CR-LF
3807 006742 104401 046454          TYPE      , EM10      ;'UNCORRECTABLE PARITY ERROR' MESSAGE
3808 006746 000404          BR      FALPRI      ;FINISH PROCESSING THE ERROR
3809
3810          ;'FATAL' MASSBUS PARITY ERROR OCCURRED
3811
3812 006750 104401 001201          FALPAR: TYPE      , $CRLF      ;CR-LF
3813 006754 104401 046517          TYPE      , EM11      ;'FATAL PARITY ERROR' MESSAGE
3814 006760 104401 054014          FALPRI: TYPE      , DRNUM      ;DRIVE NUMBER
3815 006764 013746 001344          MOV      UNIT,-(SP) ;SAVE UNIT FOR TYPEOUT
3816          ;TYPE DRIVE NUMBER
3817 006770 104403          TYPOS      ;GO TYPE--OCTAL ASCII
3818 006772 002          .BYTE      2      ;TYPE 2 DIGIT(S)
3819 006773 000          .BYTE      0      ;SUPPRESS LEADING ZEROS

```

```

3820 006774 104401 001201          TYPE      $CRLF          ;CR-LF
3821 007000 004737 023644          JSR      PC,INCTOT      ;INCREMENT TOTAL ERROR COUNT
3822 007004 032777 100000 172142  BIT      #SW15,$SWR      ;HALT ON ERROR ?
3823 007012 001401                BEQ      1$             ;BR IF NOT
3824 007014 000000                HALT                      ;ERROR HALT
3825 007016 000207          1$:      RTS      PC
3826
3827          ;SOFTWARE TIMEOUT OCCURRED
3828
3829 007020 004737 020406          SWTIM:  JSR      PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
3830 007024 104414 046603          DISPLY  EM13             ;PRINT THE TIME OUT MESSAGE
3831 007030 004737 020462          JSR      PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
3832 007034 004737 021070          JSR      PC,LINE3      ;PRINT LINE 3 OF ERROR MESSAGE
3833 007040 004737 021540          JSR      PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
3834 007044 004737 023644          JSR      PC,INCTOT      ;INCREMENT TOTAL ERROR COUNT
3835 007050 004737 022150          JSR      PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
3836 007054 000207          RTS      PC                     ;RETURN
3837
3838          ;DRIVE WENT OFFLINE
3839
3840 007056 104401 001201          OFLIN:  TYPE      $CRLF          ;CR-LF
3841 007062 104401 046655          TYPE      EM14             ;'DRIVE WENT OFFLINE' MESSAGE
3842 007066 104401 054014          TYPE      DRNUM          ;DRIVE NUMBER
3843 007072 013746 001344          MOV      UNIT,-(SP)        ;SAVE UNIT FOR TYPEOUT
3844
3845 007076 104403                TYP0S                     ;TYPE DRIVE NUMBER
3846 007100          .BYTE      2                ;GO TYPE--OCTAL ASCII
3847 007101          .BYTE      0                ;TYPE 2 DIGIT(S)
3848 007102 104401 001201          TYPE      $CRLF          ;CR-LF
3849 007106 004737 020406          JSR      PC,LINE1      ;PRINT LINE 1 OF THE ERROR MESSAGE
3850 007112 104414 046655          DISPLY  EM14             ;PRINT OFFLINE MESSAGE
3851 007116 004737 020462          JSR      PC,LINE2      ;PRINT LINE 2 OF THE ERROR MESSAGE
3852 007122 004737 021070          JSR      PC,LINE3      ;PRINT LINE 3 OF THE ERROR MESSAGE
3853 007126 004737 021540          JSR      PC,LINE4      ;PRINT LINE 4 OF THE ERROR MESSAGE
3854 007132 004737 023644          JSR      PC,INCTOT      ;INCREMENT TOTAL ERROR COUNT
3855 007136 004737 022150          JSR      PC,LINE7      ;PRINT LINE 7 OF THE ERROR MESSAGE
3856 007142 000137 026764          JMP      DROP          ;DROP THE DRIVE
3857
3858          ;PORT REQUEST TIMEOUT ERROR
3859
3860 007146 004737 020406          PRTIM:  JSR      PC,LINE1      ;TYPE LINE 1 OF THE ERROR MESSAGE
3861 007152 104414 046700          DISPLY  EM15             ;PRINT PORT TIME OUT MESSAGE
3862 007156 004737 020462          JSR      PC,LINE2      ;TYPE LINE 2 OF THE ERROR MESSAGE
3863 007162 004737 021070          JSR      PC,LINE3      ;TYPE LINE 3 OF THE ERROR MESSAGE
3864 007166 004737 021540          JSR      PC,LINE4      ;TYPE LINE 4 OF THE ERROR MESSAGE
3865 007172 004737 023644          JSR      PC,INCTOT      ;INCREMENT TOTAL ERROR COUNT
3866 007176 004737 022150          JSR      PC,LINE7      ;TYPE LINE 7 OF THE ERROR MESSAGE
3867 007202 000207          RTS      PC                     ;RETURN
3868
3869          ;PROCESS ORDER COMPLETION WITH 'ERROR' & 'DONE' BITS SET
3870
3871 007204 032760 000030 000016  DONE:  BIT      #BIT04!BIT03,$STATUS(R0) ;UNSAFE OCCURRED
3872 007212 001402                BEQ      .+6              ;BR IF NOT
3873 007214 000137 012306          JMP      UNSAF          ;REPORT UNSAFE
3874 007220 032760 040000 000244  BIT      #BIT14,$RMCS2(R0) ;IS 'WCE' SET ?
3875 007226 001402                BEQ      .+6              ;BRANCH IF NOT SET

```

```

3876 007230 000137 010166      JMP      WCKER      ;WRITE CHECK ERROR
3877 007234 032760 040000 000246  BIT      #BIT14,$RMDS(RO) ;CHECK 'ERR'
3878 007242 001002          BNE      1$      ;BR IF SET
3879 007244 000137 012052      JMP      TRFER      ;PROCESS 'TRE'
3880 007250 032760 000400 000250 1$:  BIT      #BIT08,$RMER1(RO) ;'HCRC' SET?
3881 007256 001402          BEQ      .+6      ;BR IF NOT
3882 007260 000137 010514      JMP      HRCER      ;PROCESS 'HCRC'
3883 007264 032760 000020 000250  BIT      #BIT04,$RMER1(RO) ;'FMT' SET?
3884 007272 001402          BEQ      .+6      ;BR IF NOT SET
3885 007274 000137 010712      JMP      CKFMT      ;CHECK FORMAT ERROR
3886 007300 032760 000200 000250  BIT      #BIT07,$RMER1(RO) ;'HCE' SET?
3887 007306 001402          BEQ      .+6      ;BR IF NOT SET
3888 007310 000137 011106      JMP      CKHCE      ;CHECK 'HCE' ERROR
3889 007314 032760 020000 000250  BIT      #BIT13,$RMER1(RO) ;'OPI' SET?
3890 007322 001402          BEQ      .+6      ;BR IF NOT SET
3891 007324 000137 011406      JMP      OPTER      ;REPORT 'OPI'
3892 007330 032760 000010 000250  BIT      #BIT3,$RMER1(RO) ;'PAR' SET?
3893 007336 001402          BEQ      .+6      ;BR IF NOT SET
3894 007340 000137 011540      JMP      PARER      ;REPORT 'PAR'
3895 007344 032760 000040 000250  BIT      #BIT5,$RMER1(RO) ;'WCF' SET?
3896 007352 001402          BEQ      .+6      ;BR IF NOT SET
3897 007354 000137 012210      JMP      WCFER      ;REPORT 'WCF'
3898 007360 032760 000020 000250  BIT      #BIT10,$RMER1(RO) ;'IAE' SET?
3899 007366 001402          BEQ      .+6      ;BR IF NOT SET
3900 007370 000137 011632      JMP      IAEER      ;REPORT 'IAE'
3901 007374 032760 004000 000250  BIT      #BIT11,$RMER1(RO) ;'WLE' SET?
3902 007402 001402          BEQ      .+6      ;BR IF NOT SET
3903 007404 000137 011664      JMP      WLEER      ;REPORT 'WLE'
3904 007410 032760 001000 000250  BIT      #BIT9,$RMER1(RO) ;'AOE' SET?
3905 007416 001405          BEQ      2$      ;BR IF NOT SET
3906 007420 032760 002000 000246  BIT      #BIT10,$RMDS(RO) ;'LST' SET?
3907 007426 001401          BEQ      2$      ;BR IF NOT SET
3908 007430 000207          RTS      PC      ;'AOE' & 'LST' SET, EXIT
3909 007432 032760 010000 000250 2$:  BIT      #BIT12,$RMER1(RO) ;SEE IF 'DTE' SET
3910 007440 001402          BEQ      .+6      ;BR IF NOT
3911 007442 000137 011516      JMP      DTEER      ;REPORT 'DTE' ERROR
3912 007446 005760 000250      TST      $RMER1(RO) ;SEE IF 'DCK' SET
3913 007452 100002          BPL      .+6      ;BR IF NOT
3914 007454 000137 007512      JMP      DCKER      ;PROCESS 'DCK'
3915 007460 032760 060000 000276  BIT      #BIT14!BIT13,$RMER2(RO) ;'SKI' OR 'OCYL' SET
3916 007466 001006          BNE      3$      ;BRANCH IF SKI, OCYL SET
3917 007470 032760 100000 000276  BIT      #BIT15,$RMER2(RO) ;BAD SPOT ?
3918 007476 001004          BNE      4$      ;BRANCH IF SO (NO, OTHER ERROR)
3919 007500 000137 010660      JMP      DRVER      ;REPORT ERROR
3920 007504 000137 012152      JMP      SKIER      ;REPORT SKI ERROR
3921 007510 000207          RTS      PC      ;EXIT FROM ERROR ANALYSIS ROUT.
3922
3923      ;PROCESS DATA ('DCK') CHECK ERROR
3924
3925 007512 022760 010041 000300 DCKER: CMP      #10041,$RMEC1(RO) ;VALID POSITION COUNT ?
3926 007520 101407          BLOS      1$      ;BR IF NOT VALID
3927 007522 005760 000300      TST      $RMEC1(RO) ;POSITION COUNT 0 ?
3928 007526 001404          BEQ      1$      ;BR IF 0'S
3929 007530 005760 000302      TST      $RMEC2(RO) ;VALUE IN PATTERN REGISTER ?
3930 007534 001027          BNE      4$      ;BR IF YES
3931 007536 000431          BR       6$      ;DATA CHECK ERROR

```


3932	007540	004737	020406		15:	JSR	PC,LINE1	:TYPE FIRST LINE OF ERROR MESSAGE
3933	007544	104414	050305			DISPLY	EM45	:TYPE 'ECC LOGIC ERROR'
3934	007550	004737	020462			JSR	PC,LINE2	:TYPE LINE 2 OF ERROR MESSAGE
3935	007554	004737	023644			JSR	PC,INCTOT	:INCREMENT TOTAL ERROR COUNT
3936	007560	012737	000003	001350		MOV	#3,RETRY	:RETRY COUNT
3937	007566	004737	015344			JSR	PC,\$RETRY	:RETRY THE ORDER
3938	007572	000403				BR	25	:RETRY WAS NOT SUCCESSFUL
3939	007574	004737	022066			JSR	PC,LINE6C	:TYPE LINE 6C OF ERROR MESSAGE
3940	007600	000402				BR	35	:FINISH THE ERROR REPORT
3941	007602	004737	022074		25:	JSR	PC,LINE6D	:TYPE LINE 6D OF ERROR MESSAGE
3942	007606	004737	022150		35:	JSR	PC,LINE7	:TYPE LINE 7 OF ERROR MESSAGE
3943	007612	000402				BR	55	:EXIT
3944	007614	004737	020270		45:	JSR	PC,SPOTCK	:SEE IF ERROR AT A BAD SPOT ON THE PACK
3945	007620	000207			55:	RTS	PC	:IT IS, DON'T REPORT IT
3946	007622	004737	020406		65:	JSR	PC,LINE1	:PRINT LINE 1 OF ERROR MESSAGE
3947	007626	104414	046755			DISPLY	EM21	:DATA CHECK ERROR
3948	007632	004737	020462		DCKER1:	JSR	PC,LINE2	:PRINT LINE 2 OF ERROR MESSAGE
3949	007636	004737	021070			JSR	PC,LINE3	:PRINT LINE 3 OF ERROR MESSAGE
3950	007642	004737	021540			JSR	PC,LINE4	:PRINT LINE 4 OF ERROR MESSAGE
3951	007646	004737	014776			JSR	PC,PRTBAD	:SEE IF BAD SECTOR TO BE PRINTED
3952	007652	012737	110100	001346		MOV	#BIT15!BIT12!BIT06,MASK	:LOAD ERROR MASK
3953	007660	032760	010100	000250		BIT	#BIT12!BIT06,\$RMER1(R0)	:CHECK 'DTE' & 'ECH'
3954	007666	001003				BNE	15	:BR IF SET
3955	007670	004737	022040			JSR	PC,LINE6	:PRINT LINE 6 OF ERROR MESSAGE
3956	007674	000477				BR	85	:FINISH THE ERROR REPORT
3957	007676	012737	000020	001350	15:	MOV	#16.,RETRY	:RETRY COUNT
3958	007704	005001				CLR	R1	:R1 IS OFFSET CODE POINTER
3959								
3960								
3961	007706	004737	016342		25:	JSR	PC,GODRIV	:RETRY
3962	007712	005760	000016		35:	TST	\$STATUS(R0)	:TEST FOR DONE
3963	007716	001775				BEQ	35	:BR IF NOT DONE
3964	007720	100075				BPL	105	:BR IF NOT ERROR
3965	007722	032760	000200	000016		BIT	#BIT7,\$STATUS(R0)	:SEE IF ORDER TERMINATED NORMALLY
3966	007730	001006				BNE	145	:BR IF NOT
3967	007732	004737	023644			JSR	PC,INCTOT	:INCREMENT TOTAL ERROR COUNT
3968	007736	104414	052433			DISPLY	LINBM	:DIFFERENT ERROR DURING RETRY
3969	007742	000137	006564			JMP	ERPRC1	:SEE WHICH ERROR
3970	007746	033760	001346	000250	145:	BIT	MASK,\$RMER1(R0)	:LOOK AT CURRENT ERROR
3971	007754	001430				BEQ	55	:BR IF DIFFERENT ERROR
3972	007756	032760	010100	000250		BIT	#BIT12!BIT6,\$RMER1(R0)	: 'ECH' OR 'DTE' STILL SET ?
3973	007764	001437				BEQ	75	:BR IF NEITHER SET
3974	007766	105207	001351			INCB	RETRY+1	:INCREMENT RETRY COUNT
3975	007772	123737	001350	001351		CMPB	RETRY,RETRY+1	:DONE ?
3976	010000	001342				BNE	25	:BR IF NOT
3977	010002	005201				INC	R1	:INCREMENT TABLE INDEX
3978	010004	116137	002464	046063		MOVB	OFFCOD(R1),GENDPB+\$FMT	:OFFSET CODE
3979	010012	001435				BEQ	95	:BR IF END OF OFFSET TABLE
3980	010014	062737	000002	001350		ADD	#2,RETRY	:NEW RETRY LIMIT
3981	010022	004737	015226			JSR	PC,OFFST	:OFFSET
3982	010026	005737	046100		45:	TST	GENDPB+\$STATUS	:SEE IF FINISHED WITH OFFSET
3983	010032	001775				BEQ	45	:BR IF NOT
3984	010034	100324				BPL	25	:BR IF NO ERROR PERFORMING OFFSET
3985	010036	004737	022416		55:	JSR	PC,LINE8	:PRINT LINE 8 OF ERROR MESSAGE
3986	010042	004737	023550		65:	JSR	PC,INCHRD	:INCREMENT 'HARD' ERROR COUNT
3987	010046	004737	023644			JSR	PC,INCTOT	:INCREMENT TOTAL ERROR COUNT

```

3988 010052 004737 022150      JSR    PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
3989 010056 004737 014776      JSR    PC,PATBAD     ;PRINT THE BAD SECTOR
3990 010062 000436              BR     13$           ;CLEAN UP AND RETURN
3991 010064 004737 022060      7$: JSR    PC,LINE6B     ;PRINT LINE 6B OF ERROR MESSAGE
3992 010070 004737 021776      JSR    PC,LINE5B     ;PRINT LINE 5B OF THE ERROR MESSAGE
3993 010074 004737 023524      8$: JSR    PC,INCSOF    ;INCREMENT 'SOFT' ERROR COUNT
3994 010100 004737 014236      JSR    PC,ECC        ;CORRECT THE ERROR USING ECC AND CHECK IT
3995 010104 000407              BR     11$           ;COMPARE THE BUFFER
3996 010106 004737 022074      9$: JSR    PC,LINE6D     ;PRINT LINE 6D OF ERROR MESSAGE
3997 010112 000753              BR     6$           ;INCREMENT ERROR COUNT
3998 010114 004737 022052      10$: JSR   PC,LINE6A    ;PRINT LINE 6A OF ERROR MESSAGE
3999 010120 004737 023524      JSR    PC,INCSOF    ;INCREMENT 'SOFT' ERROR COUNT
4000 010124 012737 000001      11$: MOV    #1,FRSTER  ;SET PROCESSING 'DCKER' INDICATOR
4001 010132 004737 013034      JSR    PC,CMPCRD    ;COMPARE THE BUFFER
4002 010136 105737 001377      TSTB   FRSTER+1    ;ERROR IN COMPARE ?
4003 010142 100406              BMI    13$           ;BRANCH IF ERROR
4004 010144 004737 023644      JSR    PC,INCTOT    ;INCREMENT TOTAL ERROR COUNT
4005 010150 104414 052633      DISPLY  LIN9C      ;'DATA COMPARE OK' MESSAGE
4006 010154 004737 022150      12$: JSR    PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
4007 010160 004737 006336      13$: JSR    PC,REFMT    ;REFORMAT THE ERROR SECTOR
4008 010164 000207              RTS     PC            ;RETURN
4009
4010 ;WRITE CHECK ERROR PROCESSING
4011
4012 010166 032760 100000 000250 WCKER: BIT    #BIT15,$RMER1(R0) ;SEE IF 'DCK' SET ALSO
4013 010174 001034              BNE    2$           ;BR IF IT IS
4014 010176 004737 020406      JSR    PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
4015 010202 104414 047061      DISPLY  EM23      ;PRINT WCE & DCK NOT
4016 010206 005037 001346      CLR     MASK      ;CLEAR ERROR MASK
4017 010212 004737 020462      JSR    PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
4018 010216 004737 021070      JSR    PC,LINE3      ;PRINT LINE 3 OF ERROR MESSAGE
4019 010222 004737 021540      JSR    PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
4020 010226 004737 021630      JSR    PC,LINE5      ;PRINT LINE 5 OF ERROR MESSAGE
4021 010232 004737 023644      JSR    PC,INCTOT    ;INCREMENT TOTAL ERROR COUNT
4022 010236 012737 000003 001350 MOV    #3,RETRY    ;RETRY LIMIT
4023 010244 004737 015344      JSR    PC,$RETRY    ;RETRY THE OPERATION
4024 010250 000403              BR     1$           ;RETRY UNSUCCESSFUL
4025 010252 004737 022066      JSR    PC,LINE6C     ;PRINT LINE 6C OF ERROR MESSAGE
4026 010256 000502              BR     8$           ;FINISH PROCESSING THE ERROR
4027 010260 004737 022074      1$: JSR    PC,LINE6D     ;PRINT LINE 6D OF ERROR MESSAGE
4028 010264 000506              BR     10$          ;FINISH PROCESSING THE ERROR
4029 010266 004737 020270      2$: JSR    PC,SPOTCK    ;SEE IF ERROR AT BAD SPOT ON THE PACK
4030 010272 000507              BR     11$          ;EXIT IF AT BAD SPOT ON PACK
4031 010274 004737 020406      JSR    PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
4032 010300 012737 047006 010326 MOV    #EM22,13$    ;ASSUME THAT EM22 WILL BE PRINTED
4033 010306 032760 040000 000244 BIT    #BIT14,$RMCS2(R0) ;DID 'WCK' ALSO SET ?
4034 010314 001003              BNE    12$          ;BR IF IT DID
4035 010316 012737 047707 010326 MOV    #EM37,13$    ;MESSAGE FOR 'DCK' AND 'WCK' NOT DURING
4036 ;WRITE CHECK
4037 010324 104414      12$: DISPLY  0            ;TYPE THE ERROR MESSAGE
4038 010326 000000      13$: .WORD 0            ;MESSAGE ADDRESS GOES HERE
4039 010330 004737 020462      JSR    PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
4040 010334 004737 021070      JSR    PC,LINE3      ;PRINT LINE 3 OF ERROR MESSAGE
4041 010340 004737 021540      JSR    PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
4042 010344 004737 021630      JSR    PC,LINE5      ;PRINT LINE 5 OF ERROR MESSAGE
4043 010350 032760 000100 000250 BIT    #BIT06,$RMER1(R0) ;ECH SET ALSO ?

```


4044	010356	001442			BEQ	8\$	FINISH PROCESSING THE ERROR
4045	010360	012737	000020	001350	3\$:	MOV	RETRY LIMIT - 16 (10)
4046	010366	004737	016342		4\$:	JSR	PC, GODRIV
4047	010372	005760	000016		5\$:	TST	STATUS(RO)
4048	010376	001775				BEQ	5\$
4049	010400	100405				BMI	6\$
4050	010402	05237	001351			INCB	RETRY+1
4051	010406	004737	022066			JSR	PC, LINE6C
4052	010412	000431				BR	9\$
4053	010414	105237	001351		6\$:	INCB	RETRY+1
4054	010420	123737	001350	001351		CMPB	RETRY, RETRY+1
4055	010426	001714				BEQ	1\$
4056	010430	032760	100000	000250		BIT	#BIT15, \$RMER1(RO)
4057	010436	001407				BEQ	7\$
4058	010440	032760	000100	000250		BIT	#BIT06, \$RMER1(RO)
4059	010446	001347				BNE	4\$
4060	010450	004737	022066			JSR	PC, LINE6C
4061	010454	000403				BR	8\$
4062	010456	004737	02416		7\$:	JSR	PC, LINE8
4063	010462	000405				BR	9\$
4064	010464	004737	023644		8\$:	JSR	PC, INCTOT
4065	010470	004737	022150			JSR	PC, LINE7
4066	010474	000406				BR	11\$
4067	010476	004737	023644		9\$:	JSR	PC, INCTOT
4068	010502	004737	022150		10\$:	JSR	PC, LINE7
4069	010506	004737	006336			JSR	PC, REFORMAT
4070	010512	000207			11\$:	RTS	PC
4071							RETURN
4072							
4073							;REPORT 'HCRC' ERROR
4074	010514	004737	020270		HRCRCR:	JSR	PC, SPOTCK
4075	010520	000456				BR	3\$
4076	010522	004737	022502			JSR	PC, READDR
4077	010526	004737	015252			JSR	PC, READHD
4078	010532	004737	020406			JSR	PC, LINE1
4079	010536	104414	046734			DISPLY	EM20
4080	010542	004737	020462			JSR	PC, LINE2
4081	010546	004737	021070			JSR	PC, LINE3
4082	010552	004737	021540			JSR	PC, LINE4
4083	010556	004737	021730			JSR	PC, LINE5A
4084	010562	032760	040000	000244		BIT	#BIT14, \$RMCS2(RO)
4085	010570	001402				BEQ	1\$
4086	010572	004737	021630			JSR	PC, LINE5
4087	010576	004737	023524		1\$:	JSR	PC, INCSOF
4088	010602	004737	023644			JSR	PC, INCTOT
4089	010606	012737	000400	001346		MOV	#BIT8, MASK
4090	010614	012737	000003	001350		MOV	#3, RETRY
4091	010622	004737	015344			JSR	PC, \$RETRY
4092	010626	000405				BR	2\$
4093	010630	004737	022066			JSR	PC, LINE6C
4094	010634	004737	022150			JSR	PC, LINE7
4095	010640	000406				BR	3\$
4096	010642	004737	022074		2\$:	JSR	PC, LINE6D
4097	010646	004737	022150			JSR	PC, LINE7
4098	010652	004737	006336			JSR	PC, REFORMAT
4099	010656	000207			3\$:	RTS	PC

```
4100
4101 ;REPORT DRIVE ERROR
4102
4103 010660 004737 020406 DRIVER: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
4104 010664 104414 047351 DISPLY EM30 ;REPORT DRIVE ERROR
4105 010670 004737 020462 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
4106 010674 004737 021070 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
4107 010700 004737 023644 JSR PC,INCTCT ;INCREMENT TOTAL ERROR COUNT
4108 010704 004737 022150 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
4109 010710 000207 RTS PC ;RETURN
4110
4111 ;PROCESS FORMAT ('FER') ERROR
4112
4113 010712 032760 000400 000250 CKFMT: BIT #BIT8,$RMER1(RO) ;'HCRC' SET ON ORIGINAL ERROR ?
4114 010720 001402 1$ BEQ 1$ ;BR IF NOT SET
4115 010722 000137 010514 JMP HCRCER ;REPORT HCRC ERROR
4116 010726 004737 022502 1$: JSR PC,READDR ;GET CORRECTED TRACK & SECTOR ADDRSSES
4117 010732 004737 015252 JSR PC,READHD ;READ HEADER
4118 010736 032737 000400 046116 BIT #BIT8,GENREG+RMER1 ;'HCRC' SET WHEN HEADER READ?
4119 010744 001002 2$ BNE 2$ ;BR IF 'HCRC' SET
4120 010746 000137 011712 JMP FMTER ;NO ERROR IS 'FMT' ONLY
4121 010752 004737 020270 2$: JSR PC,SPOTCK ;SEE IF ERROR AT BAD SPOT ON THE PACK
4122 010756 000452 5$ BR 5$ ;EXIT IF IT IS
4123 010760 004737 020406 JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
4124 010764 104414 047140 DISPLY EM24 ;HEADER READ ERROR - FMT BIT DROPPED UP
4125 010770 004737 020462 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
4126 010774 004737 021070 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
4127 011000 004737 021540 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
4128 011004 032760 040000 000244 BIT #BIT14,$RMCS2(RO) ;'WCE' ERROR ALSO ?
4129 011012 001402 3$ BEQ 3$ ;BR IF NOT
4130 011014 004737 021630 JSR PC,LINES ;DISPLAY WORDS WHICH CAUSED 'WCE'
4131 011020 004737 021730 3$: JSR PC,LINESA ;DISPLAY HEADER
4132 011024 004737 023524 JSR PC,INCSOF ;INCREMENT SOFT ERROR COUNT
4133 011030 004737 023644 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
4134 011034 012737 000020 001346 MOV #BIT4,MASK ;SET ERROR MASK
4135 011042 012737 000003 001350 MOV #3,RETRY ;RETRY LIMIT
4136 011050 004737 015344 JSR PC,$RETRY ;RETRY THE ORDER
4137 011054 000405 4$ BR 4$ ;RETRY NOT SUCCESSFUL
4138 011056 004737 022066 JSR PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
4139 011062 004737 022150 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
4140 011066 000406 5$ BR 5$ ;EXIT
4141 011070 004737 022074 4$: JSR PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
4142 011074 004737 022150 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
4143 011100 004737 006336 JSR PC,REFMT ;REFORMAT THE ERROR SECTOR
4144 011104 000207 5$: RTS PC ;RETURN
4145
4146 ;PROCESS HEADER COMPARE ('HCE') ERROR
4147
4148 011106 032760 000400 000250 CKHCE: BIT #BIT8,$RMER1(RO) ;HCRC SET ON ORIGINAL ERROR ?
4149 011114 001402 1$ BEQ 1$ ;BR IF NOT SET
4150 011116 000137 010514 JMP HCRCER ;REPORT HEADER CRC ERROR
4151 011122 004737 022502 1$: JSR PC,READDR ;GET CURRENT SECTOR & TRACK ADDRS
4152 011126 004737 015252 JSR PC,READHD ;READ HEADER OF CURRENT SECTOR
4153 011132 032737 000400 046116 BIT #BIT8,GENREG+RMER1 ;'HCRC' SET ?
4154 011140 001016 3$ BNE 3$ ;BR IF SET
4155 011142 042737 150000 056512 BIC #150000,CYLDER ;CLEAR FORMAT,MFG,USER BITS FROM HEADER
```

```

4156 011150 026037 000270 056512 CMP $RMDC(RO),CYLDER ;CORRECT CYLINDER ?
4157 011156 001402 BEQ 2$ ;BR IF IT IS
4158 011160 000137 011332 JMP POSER ;REPORT POSITIONING ERROR
4159 011164 052737 150000 056512 2$: BIS #150000,CYLDER ;RESTORE THE FORMAT,MFG,USER BITS
4160 011172 000137 011770 JMP HCEER ;REPORT 'HCE' ERROR
4161 011176 004737 020270 3$: JSR PC,SPOTCK ;SEE IF ERROR AT BAD SPOT
4162 011202 000452 BR 6$ ;EXIT IF IT IS
4163 011204 004737 020406 JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
4164 011210 104414 047206 DISPLY EM25 ;HEADER READ ERROR - 'HCE' SET
4165 011214 004737 020462 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
4166 011220 004737 021070 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
4167 011224 004737 021540 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
4168 011230 032760 040000 000244 BIT #BIT14,$RMCS2(RO) ;'HCE' ERROR ALSO ?
4169 011236 001402 BEQ 4$ ;BR IF NOT
4170 011240 004737 021630 JSR PC,LINES ;DISPLAY WORDS WHICH CAUSED 'HCE'
4171 011244 004737 021730 4$: JSR PC,LINESA ;PRINT LINE 5 OF ERROR MESSAGE
4172 011250 004737 023524 JSR PC,INCSOF ;INCREMENT SOFT ERROR COUNT
4173 011254 004737 023644 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
4174 011260 012737 000200 001346 MOV #BIT7,MASK ;SET ERROR MASK
4175 011266 012737 000003 001350 MOV #3,RETRY ;RETRY LIMIT
4176 011274 004737 015344 JSR PC,$RETRY ;RETRY THE ORDER
4177 011300 000405 BR 5$ ;RETRY NOT SUCCESSFUL
4178 011302 004737 022066 JSR PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
4179 011306 004737 022150 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
4180 011312 000406 BR 6$ ;EXIT
4181 011314 004737 022074 5$: JSR PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
4182 011320 004737 022150 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
4183 011324 004737 006336 JSR PC,REFMT ;REFORMAT THE ERROR SECTOR
4184 011330 000207 6$: RTS PC ;RETURN
4185
4186 ;REPORT POSSIBLE POSITIONING ERROR
4187
4188 011332 004737 015174 POSER: JSR PC,RECALT ;RECALIBRATE
4189 011336 004737 020406 JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
4190 011342 104414 050503 DISPLY EM51 ;PROGRAM DETECTED POSITIONING ERROR
4191 011346 004737 020462 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
4192 011352 004737 021116 JSR PC,LINE3C ;PRINT LINE 3C OF ERROR MESSAGE
4193 011356 052737 150000 056512 BIS #150000,CYLDER ;RESTORE THE FORMAT BIT
4194 011364 004737 021730 JSR PC,LINESA ;PRINT LINE 5A OF THE ERROR MESSAGE
4195 011370 004737 023620 JSR PC,INCMIS ;INCREMENT MISPOSITIONING COUNT
4196 011374 004737 023644 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
4197 011400 004737 022276 JSR PC,LINE7A ;PRINT LINE 7A OF ERROR MESSAGE
4198 011404 000207 RTS PC ;EXIT
4199
4200 ;REPORT 'OPI' ERROR
4201
4202 011406 004737 020270 OPIER: JSR PC,SPOTCK ;SEE IF ERROR AT BAD SPOT
4203 011412 000207 RTS PC ;RETURN IF IT IS
4204 011414 004737 020406 JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
4205 011420 104414 047403 DISPLY EM31 ;'OPI' ERROR
4206 011424 004737 020462 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
4207 011430 004737 021070 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
4208 011434 004737 021540 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
4209 011440 004737 023644 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
4210 011444 012737 020000 001346 MOV #BIT13,MASK ;ERROR MASK
4211 011452 012737 000003 001350 OPIER1: MOV #3,RETRY ;RETRY LIMIT

```

4212	011460	004737	015344
4213	011464	000405	
4214	011466	004737	022066
4215	011472	004737	022150
4216	011476	000207	
4217	011500	004737	022074
4218	011504	004737	022150
4219	011510	004737	006336
4220	011514	000207	
4221			
4222			
4223			
4224	011516	004737	020270
4225	011522	000207	
4226	011524	004737	020406
4227	011530	104414	047446
4228	011534	000137	007632
4229			
4230			
4231			
4232	011540	004737	020406
4233	011544	104414	047501
4234	011550	004737	020462
4235	011554	004737	021174
4236	011560	004737	021540
4237	011564	004737	023644
4238	011570	012737	000010
4239	011576	012737	000003
4240	011604	004737	015344
4241	011610	000405	
4242	011612	004737	022066
4243	011616	004737	022150
4244	011622	000207	
4245	011624	004737	022074
4246	011630	000772	
4247			
4248			
4249			
4250	011632	004737	020406
4251	011636	104414	047620
4252	011642	004737	020462
4253	011646	004737	021262
4254	011652	004737	023644
4255	011656	004737	022150
4256	011662	000207	
4257			
4258			
4259			
4260	011664	004737	020406
4261	011670	104414	047656
4262	011674	004737	020462
4263	011700	004737	023644
4264	011704	004737	022150
4265	011710	000207	
4266			
4267			

```

1$:      JSR      PC,$RETRY      ;RETRY THE ORDER
        BR       1$             ;RETRY UNSUCCESSFUL
        JSR      PC,LINE6C      ;PRINT LINE 6C OF ERROR MESSAGE
        JSR      PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
        RTS      PC             ;EXIT
1$:      JSR      PC,LINE6D      ;PRINT LINE 6D OF ERROR MESSAGE
        JSR      PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
        JSR      PC,REFMT       ;REFORMAT THE ERROR SECTOR
        RTS      PC             ;RETURN

;REPORT 'DTE' ERROR

DTEER:   JSR      PC,SPOTCK      ;SEE IF ERROR AT BAD SPOT
        RTS      PC             ;RETURN IF IT IS
        JSR      PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
        DISPLY   EM32          ;'DTE' ERROR
        JMP      DCKER1        ;FINISH PROCESSING THE 'DTE' ERROR

;REPORT 'PAR' ERROR

PARER:   JSR      PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
        DISPLY   EM33          ;REPORT 'PAR'
        JSR      PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
        JSR      PC,LINE3E      ;PRINT LINE 3E OF ERROR MESSAGE
        JSR      PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
        JSR      PC,INCTOT      ;INCREMENT TOTAL ERROR COUNT
        MOV      #BIT03 MASK    ;ERROR MASK
        MOV      #3,RETRY      ;RETRY LIMIT
        JSR      PC,$RETRY      ;RETRY ORDER
        BR       2$            ;RETRY UNSUCCESSFUL
1$:      JSR      PC,LINE6C      ;RETRY SUCCESSFUL
        JSR      PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
        RTS      PC             ;EXIT
2$:      JSR      PC,LINE6D      ;PRINT LINE 6D OF ERROR MESSAGE
        BR       1$            ;FINISH ERROR MESSAGE

;REPORT 'IAE' ERROR

IAEER:   JSR      PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
        DISPLY   EM35          ;REPORT 'IAE'
        JSR      PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
        JSR      PC,LINE3F      ;PRINT LINE 3F OF ERROR MESSAGE
        JSR      PC,INCTOT      ;INCREMENT TOTAL ERROR COUNT
        JSR      PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
        RTS      PC             ;RETURN

;REPORT 'WLE' ERROR

WLEER:   JSR      PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
        DISPLY   EM36          ;REPORT 'WLE'
        JSR      PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
        JSR      PC,INCTOT      ;INCREMENT TOTAL ERROR COUNT
        JSR      PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
        RTS      PC             ;RETURN

;REPORT FORMAT ERROR

```

```

4268
4269 011712 004737 020406 FMTER: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
4270 011716 104414 047267 DISPLY EM26 ;FORMAT ERROR
4271 011722 004737 020462 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
4272 011726 004737 021070 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
4273 011732 004737 021540 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
4274 011736 032760 040000 000244 BIT #BIT14,$RMCS2(RO) ;'WCE' ERROR ALSO ?
4275 011744 001402 BEQ 1$ ;BR IF NOT
4276 011746 004737 021630 JSR PC,LINE5 ;DISPLAY WORDS WHICH CAUSED 'WCE'
4277 011752 004737 021730 1$: JSR PC,LINE5A ;PRINT LINE 5A OF ERROR MESSAGE
4278 011756 004737 023644 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
4279 011762 004737 022150 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
4280 011766 000207 RTS PC
4281
4282 ;REPORT HEADER COMPARE ERROR
4283
4284 011770 004737 020406 HCEER: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
4285 011774 104414 047314 DISPLY EM27 ;HEADER COMPARE ERROR
4286 012000 004737 020462 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
4287 012004 004737 021070 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
4288 012010 004737 021540 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
4289 012014 032760 040000 000244 BIT #BIT14,$RMCS2(RO) ;'WCE' ERROR ALSO ?
4290 012022 001402 BEQ 1$ ;BR IF NOT
4291 012024 004737 021630 JSR PC,LINE5 ;DISPLAY WORDS WHICH CAUSED 'WCE'
4292 012030 004737 021730 1$: JSR PC,LINE5A ;PRINT LINE 5A OF ERROR MESSAGE
4293 012034 004737 023644 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
4294 012040 004737 022150 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
4295 012044 004737 006336 JSR PC,REFMT ;REFORMAT THE ERROR SECTOR
4296 012050 000207 RTS PC
4297
4298 ;PROCESS CONTROL/INTERFACE TRANSFER ERROR
4299
4300 012052 004737 020406 TRFER: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
4301 012056 104414 047771 DISPLY EM40 ;RH11 OR UNIBUS TRANSFER ERROR
4302 012062 004737 020462 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
4303 012066 004737 021070 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
4304 012072 004737 021540 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
4305 012076 004737 023644 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
4306 012102 032760 121400 000244 BIT #BIT15:BIT13:BIT9:BIT8,$RMCS2(RO) ;'DLT','UPE','MXF','MOPE' SET ?
4307 012110 001415 BEQ 2$ ;BR IF NONE SET
4308 012112 012737 000003 001350 MOV #3,RETRY ;RETRY LIMIT
4309 012120 005027 001346 CLR MASK ;CLEAR ERROR MASK
4310 012124 004737 015344 JSR PC,$RETRY ;RETRY THE OPERATION
4311 012130 000403 BR 1$ ;RETURN HERE IF RETRY UNSUCCESSFUL
4312 012132 004737 022066 JSR PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
4313 012136 000402 BR 2$ ;FINISH THE ERROR REPORT
4314 012140 004737 022074 1$: JSR PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
4315 012144 004737 022150 2$: JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
4316 012150 000207 RTS PC
4317
4318 ;PROCESS 'SKI' OR 'OCYL' ERRORS
4319
4320 012152 004737 020406 SKIER: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
4321 012156 104414 050445 DISPLY EM50 ;'SKI' OR 'OCYL' ERROR
4322 012162 004737 020462 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
4323 012166 004737 021104 JSR PC,LINE3B ;PRINT LINE 3B OF ERROR MESSAGE

```

```

4324 012172 004737 023644      JSR    PC,INCTOT      ;INCREMENT TOTAL ERROR COUNT
4325 012176 004737 023574      JSR    PC,INCSKI      ;INCREMENT 'SKI' OR 'OCYL' ERROR COUNT
4326 012202 004737 022276      JSR    PC,LINE7A      ;PRINT LINE 7A OF ERROR MESSAGE
4327 012206 000207              RTS    PC
4328
4329                                ;REPORT WRITE CLOCK FAILURE ('WCF')
4330
4331 012210 004737 020406      WCFER: JSR    PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
4332 012214 104414 047556      DISPLY  EM34      ;REPORT WRITE CLOCK FAILURE
4333 012220 004737 020462      JSR    PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
4334 012224 004737 021076      JSR    PC,LINE3A      ;PRINT LINE 3A OF ERROR MESSAGE
4335 012230 004737 021540      JSR    PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
4336 012234 004737 023644      JSR    PC,INCTOT      ;INCREMENT TOTAL ERROR COUNT
4337 012240 004737 014776      JSR    PC,PRTBAD      ;SEE IF BAD SECTOR TO BE PRINTED
4338 012244 012737 000003      MOV     #3,RETRY      ;RETRY COUNT
4339 012252 012737 000040      MOV     #BIT05,MASK      ;ERROR MASK
4340 012260 004737 015344      JSR    PC,$RETRY      ;RETRY THE ORDER
4341 012264 000405              BR     2$      ;RETURN HERE IF RETRY UNSUCCESSFUL
4342 012266 004737 022066      JSR    PC,LINE6C      ;PRINT LINE 6C OF ERROR MESSAGE
4343 012272 004737 022150      1$: JSR    PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
4344 012276 000207              RTS    PC
4345 012300 004737 022074      2$: JSR    PC,LINE6D      ;PRINT LINE 6D OF ERROR MESSAGE
4346 012304 000772              BR     1$
4347
4348                                ;PROCESS DRIVE UNSAFE ERROR
4349
4350 012306 004737 020406      UNSAF: JSR    PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
4351 012312 104414 050546      DISPLY  EM60      ;REPORT DRIVE UNSAFE
4352 012316 004737 020462      JSR    PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
4353 012322 004737 021070      JSR    PC,LINE3      ;PRINT LINE 3 OF ERROR MESSAGE
4354 012326 004737 023644      JSR    PC,INCTOT      ;INCREMENT TOTAL ERROR COUNT
4355 012332 012737 000003      MOV     #3,RETRY      ;RETRY COUNT
4356 012340 004737 015344      JSR    PC,$RETRY      ;RETRY THE ORDER
4357 012344 000403              BR     1$      ;RETRY WAS UNSUCCESSFUL
4358 012346 004737 022066      JSR    PC,LINE6C      ;PRINT LINE 6C OF ERROR MESSAGE
4359 012352 000402              BR     2$      ;CONTINUE WITH ERROR REPORT
4360 012354 004737 022074      1$: JSR    PC,LINE6D      ;PRINT LINE 6D OF ERROR MESSAGE
4361 012360 004737 022150      2$: JSR    PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
4362 012364 000207              RTS    PC      ;RETURN
4363
4364                                ;REPORT AN 'UNKNOWN' DATA PATTERN
4365
4366 012366 105737 001376      NOMTCH: TSTB   FRSTER      ;FIRST ERROR IN THE SECTOR ?
4367 012372 001013              BNE     1$      ;BR IF NOT OR IF PROCESSING 'DCKER'
4368 012374 004737 020406      JSR    PC,LINE1      ;TYPE LINE 1 OF ERROR MESSAGE
4369 012400 104414 050154      DISPLY  EM43      ;'CAN'T MATCH DATA WITH PATTERN'
4370 012404 004737 020462      JSR    PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
4371 012410 004737 021076      JSR    PC,LINE3A      ;PRINT LINE 3A OF ERROR MESSAGE
4372 012414 004737 021540      JSR    PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
4373 012420 000404              BR     2$      ;CONTINUE PROCESSING ERROR
4374 012422 104414 050154      1$: DISPLY  ,EM43      ;'CAN'T MATCH DATA WITH PATTERN'
4375 012426 104414 001201      DISPLY  ,$CRLF      ;CR-LF
4376 012432 104414 052563      2$: DISPLY  ,LIN9I      ;HEADER FOR DATA PRINTOUT
4377 012436 010146              MOV     R1,-(SP)      ;ADDRESS OF WORD 1
4378 012440 004737 022430      JSR    PC,LINOC      ;TYPE WORD 1
4379 012444 104414 053246      DISPLY  ,LINSP      ;SPACES

```


4380	012450	012146			MOV	(R1)+, -(SP)	: ADDRESS OF WORD 1
4381	012452	004737	022430		JSR	PC, LINOCT	: TYPE WORD 1
4382	012456	104414	001201		DISPLY	\$CRLF	: CR-LF
4383	012462	010146			MOV	R1, -(SP)	: ADDRESS OF WORD 2
4384	012464	004737	022430		JSR	PC, LINOCT	: TYPE WORD 2
4385	012470	104414	053246		DISPLY	LINSP	: SPACES
4386	012474	012146			MOV	(R1)+, -(SP)	: ADDRESS OF WORD 2
4387	012476	004737	022430		JSR	PC, LINOCT	: TYPE WORD 2
4388	012502	104414	001201		DISPLY	\$CRLF	: CR-LF
4389	012506	010146			MOV	R1, -(SP)	: ADDRESS OF WORD 3
4390	012510	004737	022430		JSR	PC, LINOCT	: TYPE WORD 3
4391	012514	104414	053246		DISPLY	LINSP	: SPACES
4392	012520	012146			MOV	(R1)+, -(SP)	: ADDRESS OF WORD 3
4393	012522	004737	022430		JSR	PC, LINOCT	: TYPE WORD 3
4394	012526	104414	001201		DISPLY	\$CRLF	: CR-LF
4395	012532	010146			MOV	R1, -(SP)	: ADDRESS OF WORD 4
4396	012534	004737	022430		JSR	PC, LINOCT	: TYPE WORD 4
4397	012540	104414	053246		DISPLY	LINSP	: SPACES
4398	012544	012146			MOV	(R1)+, -(SP)	: ADDRESS OF WORD 4
4399	012546	004737	022430		JSR	PC, LINOCT	: TYPE WORD 4
4400	012552	104414	001201		DISPLY	\$CRLF	: CR-LF
4401	012556	062701	000770		ADD	#(252.*2.), R1	: INCREMENT BUFFER POINTER
4402	012562	162737	000400	001410	SUB	#256, CMCNT	: ADJUST WORD COUNT FOR MATCH
4403	012570	132760	000001	000024	BITB	#BIT00, \$CODE(R0)	: HEADER OPERATION INVOLVED ?
4404	012576	001405			BEQ	3\$	: NO
4405	012600	062701	000004		ADD	#4, R1	: ADJUST BUFFER ADDRESS
4406	012604	162737	000002	001410	SUB	#2, CMCNT	: ADJUST WORD COUNT
4407	012612	005002			CLR	R2	: CLEAR 'WORDS TO COMPARE' COUNT IN R2
4408	012614	112737	000001	001376	MOVB	#1, FRSTER	: SET 'NOT FIRST ERROR' INDICATOR
4409	012622	112737	177777	001377	MOVB	#-1, FRSTER+1	: SET ERROR FOUND INDICATOR
4410	012630	013737	001472	001406	MOV	CMPLMT, LIMIT	: RESET THE COMPARE ERROR TIMEOUT LIMIT
4411	012636	000207			RTS	PC	: RETURN
4412							
4413							
4414							
4415	012640	032760	060000	000234	CKERR:	BIT #60000, \$RMCS1(R0)	: SEE IF 'TRE' OR 'MCPE' SET
4416	012646	001012			BNE	1\$	: BR IF EITHER SET
4417	012650	032760	177400	000244	BIT	#177400, \$RMCS2(R0)	: SEE IF ERROR BITS IN CS2 SET
4418	012656	001006			BNE	1\$	: BR IF ANY SET
4419	012660	005760	000250		TST	\$RMER1(R0)	: ANY BITS SET IN ER1
4420	012664	001003			BNE	1\$	: BR IF ANY SET
4421	012666	005760	000276		TST	\$RMER2(R0)	: ANY BITS SET IN ER2 ?
4422	012672	001416			BEQ	2\$	: BR IF NONE SET
4423	012674	004737	020406		JSR	PC, LINE1	: PRINT LINE 1 OF ERROR MESSAGE
4424	012700	104414	050221		DISPLY	EM44	: ERROR BITS SET BUT 'SC' OR 'TRE' NOT SET
4425	012704	004737	020462		JSR	PC, LINE2	: PRINT LINE 2 OF ERROR MESSAGE
4426	012710	004737	021070		JSR	PC, LINE3	: PRINT LINE 3 OF ERROR MESSAGE
4427	012714	004737	021540		JSR	PC, LINE4	: PRINT LINE 4 OF ERROR MESSAGE
4428	012720	004737	023644		JSR	PC, INCTOT	: INCREMENT TOTAL ERROR COUNT
4429	012724	004737	022150		JSR	PC, LINE7	: PRINT LINE 7 OF ERROR MESSAGE
4430	012730	000207			RTS	PC	: RETURN
4431							
4432							
4433							
4434	012732	005760	000236		CKBUS:	TST \$RMWC(R0)	: CHECK WORD COUNT
4435	012736	001010			BNE	1\$	: BR IF NOT ZERO

```

4436 012740 016046 000020      MOV      $WDL(R0),-(SP) ;WORD LENGTH
4437 012744 006316      ASL      (SP) ;CHANGE INTO BYTE COUNT
4438 012746 066016 000006      ADD      $BUF(R0),(SP) ;ADD THE STARTING LOCATION
4439 012752 022660 000240      CMP      (SP)+,$RMB(R0) ;BUFFER ADDRESS PROPER ?
4440 012756 001416      BEQ      2$ ;BR IF OK
4441 012760 004737 020406 1$: JSR      PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
4442 012764 104414 050027      DISPLY  EM41 ;BUS ADDRESS OR WORD COUNT INCORRECT
4443 012770 004737 020462      JSR      PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
4444 012774 004737 021126      JSR      PC,LINE3D ;PRINT LINE 3D OF ERROR MESSAGE
4445 013000 004737 021540      JSR      PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
4446 013004 004737 023644      JSR      PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
4447 013010 004737 022150      JSR      PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
4448 013014 000207 2$: RTS      PC
4449
4450 ;COMPARE THE BUFFER
4451
4452 013016 132760 000004 000024 CMPAR: BITB  #BIT02,$CODE(R0) ;SEE IF READ ORDER
4453 013024 001001      BNE      1$ ;BR IF IT IS
4454 013026 000207      RTS      PC ;RETURN
4455 013030 005037 001376 1$: CLR      FASTER ;CLEAR 'FIRST ERROR' INDICATOR
4456 013034 032777 000002 166112 CMPARD: BIT  #SW01,$SWR ;IS SWITCH 1 SET?
4457 013042 001401      BEQ      1$ ;BR IF NOT
4458 013044 000207      RTS      PC ;YES, DON'T COMPARE
4459 013046 005037 001404 1$: CLR      ERCTR ;CLEAR THE ERROR COUNTER
4460 013052 016001 000006      MOV      $BUF(R0),R1 ;BUFFER ADDRESS
4461 013056 016037 000020 001410      MOV      $WDL(R0),CMCNT ;WORD COUNT TO WORKING LOCATION
4462 013064 066037 000236 001410      ADD      $RMC(R0),CMCNT ;CALCULATE ACTUAL WORDS TRANSFERED
4463 013072 016037 000012 001412      MOV      $CYL(R0),CMCYL ;CYLINDER ADDRESS WORKING LOCATION
4464 013100 052737 010000 001412      BIS      #BIT12,CMCYL ;SET FORMAT BIT
4465 013106 052737 140000 001412      BIS      #140000,CMCYL ;SET MFG AND USER BITS
4466 013114 016037 000010 001414      MOV      $SEC(R0),CMSEC ;SECTOR & TRACK ADDRESSES TO WORKING LOCNS
4467 013122 013737 001472 001406      MOV      CMPLMT,LIMIT ;DISPLAY LIMIT
4468 013130 005237 001406      INC      LIMIT ;CONVERT PARAMETER INTO LIMIT VALUE
4469 013134 012737 177777 001374 CMSTR: MOV  #-1,ZROIND ;CLEAR THE 'ZERO'S' INDICATOR
4470 013142 005037 001400      CLR      SAVER1 ;CLEAR THE R1 SAVE WORD
4471 013146 005037 001402      CLR      SAVER5 ;CLEAR THE R5 SAVE WORD
4472 013152 023760 001410 000022      CMP      CMCNT,$$SEC(R0) ;IS BUFFER SIZE GREATER THAN ONE SECTOR ?
4473 013160 101005      BHI      1$ ;BR IF IT IS
4474 013162 013702 001410      MOV      CMCNT,R2 ;LESS THAN, USE REMAINING BUFFER
4475 013166 005037 001410      CLR      CMCNT ;SET COUNTER TO 0
4476 013172 000405      BR      2$
4477 013174 016002 000022 1$: MOV      $$SEC(R0),R2 ;COMPARE SECTOR
4478 013200 166037 000022 001410      SUB      $$SEC(R0),CMCNT ;DECREMENT WORD COUNT
4479 013206 126027 000024 000005 2$: CMPB   $CODE(R0),#5 ;READ HEADER & DATA?
4480 013214 001025      BNE      CMDAT ;BR IF NOT
4481 013216 052711 140000      CMHED: BIS  #BIT15:BIT14,(R1) ;SET BIT14,BIT15 IN CASE BAD SPOT ENCOUNTER
4482 013222 023721 001412      CMP      CMCYL,(R1)+ ;CHECK CYLINDER
4483 013226 001402      BEQ      1$ ;BR IF COMPARE OK
4484 013230 004737 013256      JSR      PC,CMSTR2 ;REPORT ERROR
4485 013234 023721 001414 1$: CMP      CMSEC,(R1)+ ;COMPARE SECTOR & TRACK
4486 013240 001402      BEQ      4$ ;BR IF EQ
4487 013242 004737 013256      JSR      PC,CMSTR2 ;REPORT ERROR
4488 013246 162702 000002 4$: SUB      #2,R2 ;SUBTRACT HEADER LENGTH FROM SIZE
4489 013252 003566      BLE      CMPRX ;BR IF FINISHED
4490 013254 000405      BR      CMDAT ;COMPARE THE DATA PORTION
4491 013256 005237 001404      CMSTR2: INC  ERCTR ;INCREMENT THE ERROR COUNT

```


4492	013262	004737	013636		JSR	PC,CMPRT	:REPORT THE COMPARISON ERROR	
4493	013266	000207			RTS	PC	:CHECK THE REST OF THE HEADER	
4494	013270	004737	014160	CMDAT:	JSR	PC,MATCH	:FIND THE PATTERN	
4495	013274	000403			BR	2\$	:FOUND A PATTERN	
4496	013276	004737	012366		JSR	PC,NOMTCH	:RETURN HERE IF NO MATCH WITH PATTERN MADE	
4497	013302	000456			BR	8\$	:BYPASS COMPARE ROUTINE	
4498	013304	011405		2\$:	MOV	(R4),R5	:ADDRESS OF PATTERN ADDRESS IN R4	
4499	013306	012703	000020		MOV	#20,R3	:R3 IS PATTERN POS COUNTER	
4500	013312	022125		3\$:	CMP	(R1)+,(R5)+	:COMPARE BUFFER WITH PATTERN	
4501	013314	001016			BNE	5\$	:BR IF NOT EQUAL	
4502	013316	005737	001404		TST	ERCTR	:ERRORS DETECTED ?	
4503	013322	001406			BEQ	4\$	:BR IF NO ERRORS	
4504	013324	032777	000010	165622	BIT	#SW3,JSWR	:SWITCH 3 SET ?	
4505	013332	001402			BEQ	4\$	:BR IF NOT SET	
4506	013334	004737	013636		JSR	PC,CMPRT	:DISPLAY THE WORD	
4507	013340	005302		4\$:	DEC	R2	:DECREMENT SIZE COUNT	
4508	013342	003436			BLE	8\$	:BR WHEN AT END	
4509	013344	005303			DEC	R3	:DECREMENT PATT POS COUNT	
4510	013346	001361			BNE	3\$	:BR IF NOT AT END OF PATT	
4511	013350	000755			BR	2\$	:RESTART THE PATTERN	
4512	013352	005761	177776	5\$:	TST	-2(R1)	:IS MISCOMPARED CHARACTER=0	
4513	013356	001410			BEQ	6\$	:BR IF YES	
4514	013360	012737	177777	001374	MOV	#-1,ZROIND	:SET NON-ZERO MISCOMPARED INDICATOR	
4515	013366	005237	001404		INC	ERCTR	:INCREMENT THE ERROR COUNTER	
4516	013372	004737	013636		JSR	PC,CMPRT	:REPORT ERROR	
4517	013376	000760			BR	4\$	:CONTINUE COMPARE	
4518	013400	105737	001376	6\$:	TSTB	FRSTER	:FIRST ERROR?	
4519	013404	100407			BMI	7\$	:BR IF NOT	
4520	013406	005037	001374		CLR	ZROIND	:SET THE ZERO INDICATOR	
4521	013412	010137	001400		MOV	R1,SAVER1	:SAVE CURRENT R1	
4522	013416	010537	001402		MOV	R5,SAVER5	:SAVE CURRENT R5	
4523	013422	000746			BR	4\$	:CONTINUE COMPARE	
4524	013424	005737	001374	7\$:	TST	ZROIND	:ANY MISCOMPARISONS NOT ZEROS ?	
4525	013430	001743			BEQ	4\$	:BR IF NONE-ALL ERRORS=ZERO	
4526	013432	004737	013636		JSR	PC,CMPRT	:REPORT ERROR	
4527	013436	000740			BR	4\$	:CONTINUE COMPARING	
4528	013440	023727	001410	000004	8\$:	CMP	CMCNT,#4	:LAST 3 WORDS ?
4529	013446	002470			BLT	CMPRX	:YES	
4530	013450	126027	000024	000005	CMPB	\$CODE(R0),#5	:READ HEAD AND DATA ?	
4531	013456	001414			BEQ	9\$	:YES	
4532	013460	013702	001410	13\$:	MOV	CMCNT,R2	:SET COUNTER = REMAIN BUFFER LENGTH	
4533	013464	020227	000004		CMP	R2,#4	:LAST 3 WORDS ?	
4534	013470	002457			BLT	CMPRX	:YES EXIT	
4535	013472	162737	000400	001410	SUB	#256,CMCNT	:GREATER THAN A SECTOR ?	
4536	013500	003673			BLE	CMDAT	:NO RETURN TO COMPARE LOOP	
4537	013502	012702	000400		MOV	#256,R2	:SET COUNTER =SECTOR SIZE	
4538	013506	000670			BR	CMDAT	:RETURN TO COMPARE LOOP	
4539	013510	105237	001414	9\$:	INCB	CMSEC	:INCREMENT COUNTER	
4540	013514	123727	001414	000040	CMPB	CMSEC,#32.	:MAX SECTOR # ?	
4541	013522	103421			BLO	10\$	:NO	
4542	013524	105037	001414		CLRB	CMSEC	:RESET SECTOR #	
4543	013530	105237	001415		INCB	CMTRK	:INCREMENT TRACK #	
4544	013534	123727	001415	000005	CMPB	CMTRK,#5	:MAX TRACK # ?	
4545	013542	103411			BLO	10\$	:NO	
4546	013544	105037	001415		CLRB	CMTRK	:RESET TRACK #	
4547	013550	005237	001412		INC	CMCYL	:INCREMENT CYLINDER NUMBER	

```

4548 013554 023727 001412 151466      CMP      CMCYL,#150000+822.      ;LAST CYLINDER ?
4549 013562 103401      BLO      10$      ;NO
4550 013564 000421      BR       CMPRX      ;NORMAL RETURN,NOT WRAP AROUND
4551 013566 052711 140000      BIS      #BIT15:BIT14,(R1) ;SET BIT15/14 INCASE BAD SPOT ENCOUNTER
4552 013572 023721 001412      CMP      CMCYL,(R1)+      ;COMPARE 1ST HEADER WORD
4553 013576 001402      BEQ      11$      ;MATCH
4554 013600 004737 013256      JSR      PC,CMSTR2      ;NOT MATCH
4555 013604 023721 001414      CMP      CMSEC,(R1)+      ;SECOND WORD OF HEADER
4556 013610 001402      BEQ      12$      ;MATCH
4557 013612 004737 013256      JSR      PC,CMSTR2      ;NOT MATCH
4558 013616 162737 000002 001410 12$: SUB      #2,CMCNT      ;ADJUST WORD COUNT
4559 013624 003401      BLE      CMPRX      ;COMPARE IS DONE
4560 013626 000714      BR       13$      ;RETURN TO COMPARE LOOP
4561
4562 013630 004737 014112      CMPRX: JSR      PC,ENDCMP      ;PRINT LAST LINE IF ERRORS
4563 013634 000207      RTS      PC
4564
4565      ;TYPE DATA COMPARE ERRORS
4566
4567 013636 005737 001400      CMPRT: TST      SAVER1      ;PRINT SAVED VALUES ?
4568 013642 001010      BNE      2$      ;BR IF NOT
4569 013644 105737 001376      TSTB     FRSTER      ;FIRST ERROR?
4570 013650 100402      BMI      1$      ;BR IF NOT
4571 013652 004737 013732      JSR      PC,4$      ;PRINT INITIAL MESSAGE INFO
4572 013656 004737 014014      JSR      PC,8$      ;PRINT REMAINDER OF MESSAGE
4573 013662 000422      BR       3$      ;EXIT
4574 013664
4575 013664 010146      MOV      R1,-(SP)      ;PUSH R1 ON STACK
4576 013666 010546      MOV      R5,-(SP)      ;PUSH R5 ON STACK
4577 013670 013701 001400      MOV      SAVER1,R1      ;DISPLAY SAVED R1
4578 013674 013705 001402      MOV      SAVER5,R5      ;DISPLAY SAVED R5
4579 013700 004737 013732      JSR      PC,4$      ;PRINT INITIAL MESSAGE INFO
4580 013704 004737 014014      JSR      PC,8$      ;PRINT SAVED VALUES
4581 013710 005037 001400      CLR      SAVER1      ;CLEAR SAVED REGISTER INDICATORS
4582 013714 005037 001402      CLR      SAVER5      ;CLEAR THE OTHER ONE
4583 013720 012605      MOV      (SP)+,R5      ;POP STACK INTO R5
4584 013722 012601      MOV      (SP)+,R1      ;POP STACK INTO R1
4585 013724 004737 014014      JSR      PC,8$      ;PRINT REMAINDER OF MESSAGE
4586 013730 000207      RTS      PC      ;RETURN
4587 013732 105737 001376      4$: TSTB     FRSTER      ;FIRST ERROR ?
4588 013736 100425      BMI      7$      ;BR IF NOT
4589 013740 001013      BNE      5$      ;BR IF FIRST ERROR AND PROCESSING 'DCK' ERROR
4590 013742 004737 020406      JSR      PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
4591 013746 104414 050073      DISPLY   EM42      ;DATA COMPARE ERROR
4592 013752 004737 020462      JSR      PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
4593 013756 004737 021076      JSR      PC,LINE3A     ;PRINT LINE 3A OF ERROR MESSAGE
4594 013762 004737 021540      JSR      PC,LINE4     ;PRINT LINE 4 OF ERROR MESSAGE
4595 013766 000404      BR       6$      ;GO TO TYPE HEADER
4596 013770 104414 052456      5$: DISPLY   ,LIN9B      ;HEADER MESSAGE OF PROCESSING 'DCK' ERROR
4597 013774 104414 001201      DISPLY   ,$CRLF      ;CR-LF
4598 014000 104414 052505      6$: DISPLY   ,LIN9H      ;DISPLAY HEADER
4599 014004 012737 177777 001376  MOV      #1,FRSTER      ;SET ERROR FLAG
4600 014012 000207      RTS      PC      ;RETURN
4601 014014 005737 001406      7$: TST      LIMIT      ;TYPEOUT LIMIT REACHED ?
4602 014020 001403      BEQ      9$      ;BR IF IT HAS
4603 014022 005337 001406      DEC      LIMIT      ;DECREMENT LIMIT COUNTER

```

```

4604 014026 001005      000200 165116 9$: BNE 10$ ;BR IF NOT AT LIMIT
4605 014030 032777      000200 165116 9$: BIT #SW07,SWR ;PRINT ALL DATA COMPARE ERRORS ?
4606 014036 001001      000200 165116 9$: BNE 10$ ;BR IF YES
4607 014040 000207      000200 165116 9$: RTS PC ;RETURN
4608 014042 010146      000200 165116 10$: MOV R1, -(SP) ;BUFFER ADDRESS
4609 014044 162716 000002      000200 165116 10$: SUB #2, (SP) ;ADJUST ADDRESS
4610 014050 004737 022430      000200 165116 10$: JSR PC, LINOCT ;TYPE IT
4611 014054 104414 053246      000200 165116 10$: DISPLY ,LINSF ;2 SPACES
4612 014060 016546 177776      000200 165116 10$: MOV -2(R5), -(SP) ;PUT GOOD DATA ON THE STACK
4613 014064 004737 022430      000200 165116 10$: JSR PC, LINOCT ;TYPE IT
4614 014070 104414 053246      000200 165116 10$: DISPLY ,LINSF ;2 SPACES
4615 014074 016146 177776      000200 165116 10$: MOV -2(R1), -(SP) ;BAD DATA
4616 014100 004737 022430      000200 165116 10$: JSR PC, LINOCT ;TYPE IT
4617 014104 104414 001201      000200 165116 10$: DISPLY $CRLF ;CR-LF
4618 014110 000207      000200 165116 10$: RTS PC ;RETURN
4619
4620 ;LAST LINE OF COMPARE ERROR REPORTING
4621
4622 014112 105737 001377      000200 165116 ENDCMP: TSTB FRSTER+1 ;ANY COMPARE ERRORS FOUND ?
4623 014116 001417      000200 165116 ENDCMP: BEQ 2$ ;BR IF NOT
4624 014120 005737 001404      000200 165116 ENDCMP: TST ERCTR ;SEE HOW MANY ERRORS
4625 014124 001410      000200 165116 ENDCMP: BEQ 1$ ;BR IF ONLY CAN'T MATCH PATTERN
4626 014126 104414 052603      000200 165116 ENDCMP: DISPLY ,LINSF ;'NUMBER OF ERRORS='
4627 014132 013746 001404      000200 165116 ENDCMP: MOV ERCTR, -(SP) ;NUMBER OF ERRORS
4628 014136 004737 022462      000200 165116 ENDCMP: JSR PC, LINDC ;TYPE IT
4629 014142 104414 001201      000200 165116 ENDCMP: DISPLY $CRLF ;CR-LF
4630 014146 004737 023644      000200 165116 1$: JSR PC, INCTOT ;INCREMENT TOTAL ERROR COUNT
4631 014152 004737 022150      000200 165116 1$: JSR PC, LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
4632 014156 000207      000200 165116 2$: RTS PC ;RETURN
4633
4634 ;ROUTINE TO MATCH THE DATA WITH A PATTERN
4635 ;CALL:
4636 ;
4637 ; MOV #BUFFER, R1 ;BUFFER ADDRESS
4638 ; JSR PC, MATCH
4639 ; RETURN1 ;PATTERN ADDRESS IN R4
4640 ; RETURN2 ;COULDN'T MATCH PATTERN
4641
4642 014160 010146      000044      000002 MATCH: MOV R1, -(SP) ;SAVE R1 ON THE STACK
4643 014162 012704 000044      000002 MATCH: MOV #44, R4 ;PATTERN TABLE INDEX
4644 014166 011601      000044      000002 1$: MOV (SP), R1 ;RELOAD R1
4645 014170 162704 000002      000002 1$: SUB #2, R4 ;DECREMENT INDEX
4646 014174 001413      000044      000002 1$: BEQ 3$ ;BR IF PATTERN NOT MATCH
4647 014176 016405 002476      000044      000002 1$: MOV STNDAT(R4), R5 ;ADDRESS OF PATTERN ADDRESS
4648 014202 012703 000004      000044      000002 1$: MOV #4, R3 ;NUMBER OF LOCATIONS TO CHECK
4649 014206 022125      000044      000002 2$: CMP (R1)+, (R5)+ ;COMPARE THE BUFFER AGAINST THE PATTERN
4650 014210 001366      000044      000002 2$: BNE 1$ ;BR IF NOT EQUAL, TRY NEXT PATTERN
4651 014212 005303      000044      000002 2$: DEC R3 ;FINISHED CHECKING?
4652 014214 001374      000044      000002 2$: BNE 2$ ;BR IF NOT FINISHED
4653 014216 062704 002476      000044      000002 2$: ADD #STNDAT, R4 ;MAKE PATTERN ADDRESS ABSOLUTE
4654 014222 000403      000044      000002 2$: BR 4$ ;EXIT
4655 014224 062766 000002 000002 3$: ADD #2, 2(SP) ;INCREMENT RETURN ADDRESS
4656 014232 012601      000002 000002 4$: MOV (SP)+, R1 ;RESTORE R1
4657 014234 000207      000002 000002 4$: RTS PC ;RETURN
4658
4659 ;USE ECC TO CORRECT THE DATA ERROR

```

Address	Offset	Value	Label	Operation	Comment	
4660	014236	016037	000240	001420	ECC: MOV \$RMB(A(R0), ECSEC	ADDRESS OF LAST LOCN XFERED
4661	014244	016046	000236		MOV \$RMC(WC(R0), -(SP)	ACT WORDS XFERED (2'S COMP)
4662	014250	066016	000020		ADD \$W(RD(R0), (SP)	ADD WORDS REQUESTED
4663	014254	005046			CLR -(SP)	CLEAR NEXT STACK LOCN
4664	014256	016046	000022		MOV \$SSEC(P0), -(SP)	SECTOR SIZE
4665	014262	004737	027522		JSR PC LINK0,	DIVIDE WORDS XFERED BY SECTOR SIZE
4666	014266	005716			TST (SP)	PARTIAL SECTOR XFERED ?
4667	014270	001413			BEQ 15	BR IF NOT
4668	014272	006316			ASL (SP)	CONVERT INTO NUMBER OF BYTES
4669	014274	161637	001420		SUB (SP), ECSEC	SUBTRACT SECTOR RESIDUE
4670	014300	126027	000024	000005	CMPB \$CODE(R0), #5	WAS OP READ HEAD & DATA
4671	014306	001007			BNE 25	BR IF NOT
4672	014310	062737	000004	001420	ADD #4, ECSEC	ADD HEADER SIZE (IN BYTES) BACK IN
4673	014316	000403			BR 25	GO ADJUST THE STACK POINTER
4674	014320	162737	001000	001420	SUB #1000, ECSEC	SUBTRACT SECTOR DATA FIELD SIZE (IN BYTES)
4675	014326	062706	000004		ADD #4, SP	ADJUST THE STACK POINTER
4676	014332	016037	000300	001416	MOV \$RMEC1(R0), ECBIT	ECC POSITION COUNT
4677	014340	005337	001416		DEC ECBIT	ADJUST THE POSITION COUNT
4678	014344	013737	001416	001426	MOV ECBIT, ECWRD	LOAD THE WORD COUNT LOCATION
4679	014352	042737	177760	001416	BIC #17, ECBIT	SAVE THE BIT OFFSET COUNT
4680	014360	042737	000017	001426	BIC #17, ECWRD	CLEAR THE BIT OFFSET
4681	014366	006237	001426		ASR ECWRD	CHANGE TO BYTE COUNT
4682	014372	006237	001426		ASR ECWRD	CHANGE TO BYTE COUNT
4683	014376	006237	001426		ASR ECWRD	CHANGE TO BYTE COUNT
4684	014402	104414	052662		DISPLY LIN10A	'ERROR BURST BEGINS AT '
4685	014406	013746	001426		MOV ECWRD, -(SP)	PUT THE WORD COUNT ON THE STACK
4686	014412	006216			ASR (SP)	CONVERT TO WORD COUNT FOR MESSAGE
4687	014414	004737	030436		JSR PC, \$SB20	CONVERT THE WORD COUNT
4688	014420	004737	030036		JSR PC, \$SUPRS	PRINT IT
4689	014424	104414	052716		DISPLY LIN10B	' IN DATA FIELD OF ERROR SECTOR'
4690	014430	063737	001420	001426	ADD ECSEC, ECWRD	FIND THE BEGINNING OF THE ERROR BURST
4691	014436	026037	000240	001426	CMP \$RMB(A(R0), ECWRD	SEE IF BURST WAS IN DATA READ
4692	014444	101002			BHI +6	BR IF IN DATA READ
4693	014446	000137	014764		JMP ECC2	NOT IN DATA READ - REPORT IT
4694	014452	016037	000302	001422	MOV \$RMEC2(R0), ECMSK0	GET THE ERROR MASK
4695	014460	005037	001424		CLR ECMSK1	CLEAR THE UPPER MASK WORD
4696	014464	005737	001416		TST ECBIT	BIT OFFSET EQUAL ZERO
4697	014470	001407			BEQ 45	BR IF IT IS
4698	014472	005337	001416		DEC ECBIT	DECREMENT THE BIT OFFSET COUNT
4699	014476	006337	001422		ASL ECMSK0	SHIFT THE ERROR MASK
4700	014502	006137	001424		ROL ECMSK1	SHIFT THE LOWER INTO THE UPPER
4701	014506	000766			BR 35	CONTINUE THE SHIFT
4702	014510	017737	164712	001432	MOV ECWRD, ECBAD0	SAVE THE INCORRECT WORD
4703	014516	005037	001434		CLR ECWRD1	CLEAR SECOND INCORRECT WORD ADDRESS
4704	014522	013746	001422		MOV ECMSK0, -(SP)	PUT LOWER MASK ON STACK
4705	014526	047716	164674		BIC ECWRD, (SP)	CLEAR ERRONEOUS ONE BITS FROM MASK
4706	014532	043777	001422	164666	BIC ECMSK0, ECWRD	CLEAR ERRONEOUS ONE BITS FROM BAD WORD
4707	014540	052677	164662		BIS (SP)+, ECWRD	SET DROPPED BITS
4708	014544	005737	001424		T	

```

4716 014602 000414          BR      ECC1      ;PRINT WORD CORRECTED
4717 014604 017737 164624 001440 5$:      MOV      DECWRD1,ECBAD1 ;SAVE THE SECOND BAD WORD
4718 014612 013746 001424          MOV      ECMSK1,-(SP) ;PUT THE UPPER MASK ON THE STACK
4719 014616 047716 164612          BIC      DECWRD1,(SP) ;CLEAR ERRONEOUS ONE BITS FROM UPPER MASK
4720 014622 043777 001424 164604          BIC      ECMSK1,DECWRD1 ;CLEAR ERRONEOUS ONE BITS FROM DATA WORD
4721 014630 052677 164600          BIS      (SP)+,DECWRD1 ;SET DROPPED BITS
4722 014634 104414 053064          ECC1:  DISPLY  LIN10H ;HEADER
4723 014640 013746 001426          MOV      ECWRD,-(SP) ;PUT ECWRD ON THE STACK
4724 014644 004737 022430          JSR      PC,LIN0CT ;TYPE ECWRD
4725 014650 104414 053246          DISPLY  LINSP ;SPACES
4726 014654 013746 001432          MOV      ECBAD0,-(SP) ;PUT ECBAD0 ON THE STACK
4727 014660 004737 022430          JSR      PC,LIN0CT ;TYPE ECBAD0
4728 014664 104414 053246          DISPLY  LINSP ;SPACES
4729 014670 017746 164532          MOV      DECWRD,-(SP) ;PUT DECWRD ON THE STACK
4730 014674 004737 022430          JSR      PC,LIN0CT ;TYPE DECWRD
4731 014700 104414 053246          DISPLY  LINSP ;SPACES
4732 014704 005737 001434          TST      ECWRD1 ;PRINT THE NEXT WORD ?
4733 014710 001427          BEQ      ECCX      ;BR IF NOT
4734 014712 104414 001201          DISPLY  $CRLF ;CR-LF
4735 014716 013746 001434          MOV      ECWRD1,-(SP) ;PUT ECWRD1 ON THE STACK
4736 014722 004737 022430          JSR      PC,LIN0CT ;TYPE ECWRD1
4737 014726 104414 053246          DISPLY  LINSP ;SPACES
4738 014732 013746 001440          MOV      ECBAD1,-(SP) ;PUT ECBAD1 ON THE STACK
4739 014736 004737 022430          JSR      PC,LIN0CT ;TYPE ECBAD1
4740 014742 104414 053246          DISPLY  LINSP ;SPACES
4741 014746 017746 164462          MOV      DECWRD1,-(SP) ;PUT DECWRD1 ON THE STACK
4742 014752 004737 022430          JSR      PC,LIN0CT ;TYPE DECWRD1
4743 014756 104414 053246          DISPLY  LINSP ;SPACES
4744 014762 000402          BR      ECCX      ;EXIT
4745 014764 104414 052757          ECC2:  DISPLY  ,LIN10C ;ERROR BURST WAS NOT TRANSFERED TO MEMORY
4746 014770 104414 001201          ECCX:  DISPLY  $CRLF ;CR-LF
4747 014774 000207          RTS      PC      ;RETURN
4748
4749 ;ROUTINE TO DISPLAY THE SECTOR WHICH GAVE THE HARD ERROR
4750
4751 014776 032777 000010 164150 PRTBAD: BIT      $SW3,$SWR ;PRINT THE BAD SECTOR ?
4752 015004 001460          BEQ      6$      ;BR IF NOT
4753 015006 016001 000240          MOV      $RMB8(R0),R1 ;PUT THE END ADDRESS INTO R1
4754 015012 016046 000020          MOV      $WRDL(R0),-(SP) ;FIND THE BEGINNING OF THE SECTOR
4755 015016 066016 000236          ADD      $RMWC(R0),-(SP) ;SUBTRACT THE WORDS NOT TRANSFERED
4756 015022 005046          CLR      -(SP) ;MAKE THE UPPER DIVIDEND 0
4757 015024 016046 000022          MOV      $SSEC(R0),-(SP) ;DIVIDE THE WORDS TRANSFERED BY THE SECTOR SIZE
4758 015030 004737 027522          JSR      PC,LINKDV ;DIVIDE
4759 015034 005716          TST      (SP) ;REMAINDER = 0 ?
4760 015036 001403          BEQ      1$      ;BR IF IT IS - COMPLETE SECTOR TRANSFERED
4761 015040 006316          ASL      (SP) ;CONVERT THE RESIDUAL SECTOR SIZE INTO BYTE COUNT
4762 015042 161601          SUB      (SP),R1 ;SUBTRACT IT FROM THE END ADDRESS
4763 015044 000410          BR      2$      ;FINISH THE SIZING
4764 015046 162701 001000          SUB      $1000,R1 ;SUBTRACT FULL SECTOR SIZE FROM END ADDR
4765 015052 126027 000024 000005 1$:      CMPB      $CODE(R0),#5 ;WAS OPERATION READ HEADER & DATA ?
4766 015060 001002          BNE      2$      ;BR IF NOT
4767 015062 162701 000004          SUB      #4,R1 ;SUBTRACT HEADER SIZE FROM ADDR
4768 015066 062706 000004          ADD      #4,SP ;RESTORE THE STACK POINTER
4769 015072 104414 053151          DISPLY  ,LIN11H ;PRINT THE HEADER
4770 015076 012702 000007          MOV      #7,R2 ;R2 CONTAINS THE WORDS/LINE COUNT
4771 015102 010146          MOV      R1,-(SP) ;PUT THE ADDRESS ON THE STACK

```

```

4772 015104 004737 022430
4773 015110 020160 000240
4774 015114 001412
4775 015116 104414 053246
4776 015122 012146
4777 015124 004737 022430
4778 015130 005302
4779 015132 001366
4780 015134 104414 001201
4781 015140 000756
4782 015142 104414 001201
4783 015146 000207
4784
4785
4786
4787
4788
4789
4790
4791 015150 111037 046062
4792 015154 112737 000117 046064
4793 015162 004037 035364
4794 015166 046062
4795 015170 000774
4796 015172 000207
4797
4798
4799
4800
4801
4802
4803
4804
4805
4806
4807
4808
4809
4810
4811 015174 111037 046062
4812 015200 112737 000107 046064
4813 015206 004037 035364
4814 015212 046062
4815 015214 000774
4816 015216 005737 046100
4817 015222 001775
4818 015224 000207
4819
4820
4821
4822
4823
4824
4825
4826
4827 015226 111037 046062

      JSR      PC,LINOC      ;TYPE THE ADDRESS
4S:    CMP      R1,$RMB4(R0) ;PRINTED ALL THE SECTOR ?
      BEQ      SS           ;BR IF ALL PRINTED
      DISPLY   LINSF        ;SPACES
      MOV      (R1)+,-(SP)  ;PUT THE DATA ON THE STACK
      JSR      PC,LINOC      ;TYPE THE DATA
      DEC      R2           ;DECREMENT THE HORIZONTAL COUNT
      BNE      4S          ;BR IF NOT AT THE END OF THE LINE
      DISPLY   $CARLF       ;CR-LF
      BR       SS          ;RESTORE THE WORDS/LINE COUNT
4S:    DISPLY   $CARLF       ;PRINT WHAT REMAINS IN THE BUFFER
6S:    RTS      PC          ;RETURN

;ROUTINE TO DO AN RTC - DRIVE SELECTED IN R0
;CALL:
      MOV      #DPB,R0      ;DPB ADDRESS
      JSR      PC,RTNCTR
      RETURN

RTNCTR: MOV      (R0),GENDPB  ;MOVE THE DRIVE # TO THE GENERAL DPB
      MOV      #RTC,GENDPB+$COMND ;COMMAND CODE
1S:    JSR      R0,RM03      ;DRIVER ENTRANCE
      GENDPB   ;DPB ADDRESS FOR ORDER
      BR       1S          ;DRIVER DIDN'T ACCEPT ORDER
      RTS      PC          ;RETURN

;ROUTINE TO DO A RECALIBRATE - DRIVE SELECTED IN R0
;CALL:
      MOV      #DPB,R0      ;DPB ADDRESS
      JSR      PC,RECALT
      RETURN

;OR
      MOV      #DPB,R0      ;DPB ADDRESS
      MOV      #DRIVE,GENDPB ;DRIVE ADDRESS
      JSR      PC,RECALTO
      RETURN

RECALT: MOV      (R0),GENDPB  ;MOVE THE DRIVE # TO THE GENERAL DPB
RECALO: MOV      #RECAL,GENDPB+$COMND ;RECALIBRATE COMMAND
1S:    JSR      R0,RM03      ;DRIVER ENTRANCE
      GENDPB   ;DPB ADDRESS FOR ORDER
      BR       1S          ;DRIVER DIDN'T ACCEPT THE ORDER
2S:    TST      GENDPB+$STATUS ;SEE IF FINISHED
      BEQ      2S          ;BR IF NOT FINISHED
      RTS      PC          ;RETURN

;OFFSET THE DRIVE IN R0 (OFFSET CODE PRELOADED INTO 'RM0F')
;CALL:
      MOV      #OFFSET,GENDPB+$FMT ;OFFSET CODE
      MOV      #DPB,R0      ;DPB ADDRESS
      JSR      PC,OFFST
      RETURN

OFFST: MOV      (R0),GENDPB  ;DRIVE # TO GENERAL DPB

```



```

4828 015232 112737 000115 046064      MOV      #OFFSET,GENDPB+$COMND ;COMMAND
4829 015240 004037 035364      JSR      RO,RM03 ;DRIVER ENTRANCE
4830 015244 046062              GENDPB ;DPB ADDRESS FOR ORDER
4831 015246 000774              BR      1$ ;DRIVER DIDN'T ACCEPT ORDER
4832 015250 000207              RTS      PC
4833
4834 ;UTILITY READ HEADER ROUTINE
4835 ;CALL:
4836      MOV      #DPB,RO ;DPB ADDRESS
4837      MOV      #SECTOR,-(SP) ;SECTOR ADDRESS
4838      MOV      #TRACK,-(SP) ;TRACK ADDRESS
4839      MOV      #CYLINDER,-(SP) ;CYLINDER ADDRESS
4840      JSR      PC,READOR
4841      RETURN
4842
4843 015252 116637 000004 046073 READHD: MOV      4(SP),GENDPB+$TRK ;TRACK ADDRESS
4844 015260 116637 000006 046072      MOV      6(SP),GENDPB+$SEC ;SECTOR ADDRESS
4845 015266 016637 000002 046074      MOV      2(SP),GENDPB+$CYL ;CYLINDER ADDRESS
4846 015274 111037 046062      MOV      (RO),GENDPB ;DRIVE NUMBER
4847 015300 112737 000173 046064      MOV      #RDHD,GENDPB+$COMND ;COMMAND
4848 015306 012737 177776 046066      MOV      #-2,GENDPB+$WORDM ;WORD CTR = 2
4849 015314 004037 035364      JSR      RO,RM03 ;DRIVER ENTRANCE
4850 015320 046062              GENDPB ;DPB ADDRESS FOR ORDER
4851 015322 000774              BR      1$ ;DRIVER DIDN'T ACCEPT COMMAND
4852 015324 005737 046100      2$: TST      GENDPB+$STATUS ;FINISHED?
4853 015330 001775              BEQ      2$ ;BR IF NOT
4854 015332 011666 000006      MOV      (SP),6(SP) ;ADJUST STACK FOR RETURN
4855 015336 062706 000006      ADD      #6,SP ;ADJUST RETRUN POINTER
4856 015342 000207              RTS      PC ;RETURN
4857
4858 ;RETRY THE PRESENT OPERATION
4859 ;CALL:
4860      MOV      #COUNT,RETRY ;RETRY COUNT
4861      JSR      PC,$RETRY
4862      RETURN1 ;RETRY UNSUCCESSFUL
4863      RETURN2 ;SUCCESSFUL RETRY
4864 ;NOTE: IF A DIFFERENT ERROR OCCURS DURING
4865 ;RETRY, THE ROUTINE EXITS TO 'ERPRC1'
4866
4867 015344 004737 016342 $RETRY: JSR      PC,GODRIV ;RE-START ORDER
4868 015350 005760 000016 1$: TST      $STATUS(RO) ;ORDER FINISHED?
4869 015354 001775              BEQ      1$ ;BR IF NOT
4870 015356 100405              BMI      2$ ;BR IF ERROR
4871 015360 105237 001351      INCB      RETRY+1 ;INCREMENT RETRY COUNT
4872 015364 062716 000002      ADD      #2,(SP) ;INCREMENT RETURN
4873 015370 000425              BR      5$ ;GO TO EXIT
4874 015372 032750 000200 000016 2$: BIT      #BIT7,$STATUS(RO) ;DID ORDER TERMINATE NORMALLY ?
4875 015400 001430              BEQ      7$ ;BR IF NOT
4876 015402 005737 001346      TST      MASK ;IS ERROR MASK 0 ?
4877 015406 001004              BNE      3$ ;BR IF NOT
4878 015410 005760 000250      TST      $RMER1(RO) ;MAKE SURE THAT THE DRIVE ERROR REG IS CLEAR
4879 015414 001014              BNE      6$ ;BR IF NOT
4880 015416 000404              BR      4$ ;CONTINUE RETRY
4881 015420 033760 001346 000250 3$: BIT      MASK,$RMER1(RO) ;SAME ERROR?
4882 015426 001407              BEQ      6$ ;BR IF NOT
4883 015430 105237 001351      4$: INCB      RETRY+1 ;INCREMENT RETRY COUNT

```

```

4884 015434 123737 001350 001351      CMPB    RETRY,RETRY+1      ;DONE ?
4885 015442 001340                      BNE      $RETRY          ;BR IF NOT DONE
4886 015444 000207                      RTS      PC              ;RETURN
4887 015446 004737 022416      5$:      JSR      PC,LINE8      ;REPORT DIFFERENT ERROR
4888 015452 004737 022150      6$:      JSR      PC,LINE7      ;PRINT LINE 7
4889 015456 005726                      TST      (SP)+          ;ADJUST STACK POINTER FOR DIRECT RETURN
4890 015460 000207                      RTS      PC              ;RETURN
4891 015462 104414 052433      7$:      DISPLY  LIN8M          ;'DIFFERENT ERROR DURING RETRY'
4892 015466 000137 006564      JMP      ERPRC1        ;REPORT THE ERROR
4893
4894      ;ROUTINE TO UPDATE THE PERFORMANCE SUMMARY STATISTICS
4895      ;CALL:
4896      ;      MOV      #DPB,RO      ;DPB ADDRESS
4897      ;      JSR      PC,STATIS
4898      ;      RETURN
4899
4900 015472 032760 000300 000016  STATIS: BIT      #BIT07:BIT06,$STATUS(RO) ;CHECK FOR DATA TERMINATION
4901 015500 001451                      BEQ      3$              ;BR IF NOT DATA TERMINATION
4902 015502 016037 000240 015626      MOV      $RMB8(RO),FACTOR ;STORE THE FINAL BUFFER ADDRESS
4903 015510 166037 000006 015626      SUB     $BUF(RO),FACTOR ;SUBTRACT THE INITIAL ADDRESS
4904 015516 001431                      BEQ      2$              ;BR IF NO DATA TRANSFER
4905 015520 006237 015626      ASR      FACTOR          ;CONVERT TO A WORD COUNT
4906 015524 063760 015626 000046      ADD     FACTOR,$TRANS(RO) ;UPDATE WORD COUNT
4907 015532 005560 000050      ADC      $TRANS+2(RO) ;ADD ANY CARRY
4908 015536 132760 000002 000024      BITB    #BIT01,$CODE(RO) ;SEE IF ORDER READ OR WRITE
4909 015544 001016                      BNE      2$              ;BRANCH IF ORDER WRITE
4910 015546 126027 000024 000001      CMPB    $CODE(RO),#1 ;PRESENT OPERATION AN AUTOMATIC WRITE CHECK ?
4911 015554 101005                      BHI      1$              ;BR IF NOT
4912 015556 066060 000020 000046      ADD     $WORDL(RO),$TRANS(RO) ;ADD WORDS WRITTEN
4913 015564 005560 000050      ADC      $TRANS+2(RO) ;ADD A CARRY
4914 015570 063760 015626 000052 1$:      ADD     FACTOR,$READ(RO) ;UPDATE THE READ WORD COUNT
4915 015576 005560 000054      ADC      $READ+2(RO) ;ADD ANY CARRY
4916 015602 026060 000012 000270 2$:      CMP     $CYL(RO),$RMD0(RO) ;DID MID-TRANSFER SEEK OCCUR
4917 015610 001405                      BEQ      3$              ;BR IF NOT
4918 015612 062760 000001 000042      ADD     #1,$POSIT(RO) ;INCREMENT SEEK COUNT
4919 015620 005560 000044      ADC      $POSIT+2(RO) ;ADD CARRY TO UPPER WORD
4920 015624 000207                      RTS      PC
4921
4922 015626 000000      FACTOR: .WORD    0 ;USED FOR WORDS TRANSFERED
4923
4924      ;ROUTINE TO GET A BUFFER
4925      ;CALL:
4926      ;      MOV      #DPB,RO      ;DPB ADDRESS
4927      ;      CLR      -(SP)          ;CLEAR THE STACK
4928      ;      JSR      PC,GETBUF
4929      ;      RETURN
4930      ;      ;BUFFER ADDRESS WILL BE ON THE STACK
4931      ;      ;STACK WILL BE ZERO IF NO BUFFER AVAILABLE
4932
4932 015630 010146                      GETBUF: MOV     R1,-(SP) ;SAVE R1
4933 015632 010246                      MOV     R2,-(SP) ;SAVE R2
4934 015634 010346                      MOV     R3,-(SP) ;SAVE R3
4935 015636 013702 001700                      MOV     BUFTBL,R2 ;NUMBER OF SEPARATE BUFFERS
4936 015642 001444                      BEQ      6$              ;IF NONE AVAILABLE
4937 015644 012701 001702                      MOV     #BUFTBL+2,R1 ;FIRST ADDRESS OF ALLOCATION TABLE
4938 015650 026061 000020 000002 1$:      CMP     $WORDL(RO),2(R1) ;SEE IF THERE IS A BLOCK LARGE ENOUGH
4939 015656 101405                      BLOS    3$              ;BRANCH IF IT IS

```



```

4940 015660 005302          DEC      R2          ; DECREMENT TABLE COUNT
4941 015662 001434          BEQ      6$          ; BR IF THROUGH TABLE
4942 015664 062701 000004    ADD      #4,R1      ; INCREMENT TABLE POINTER
4943 015670 000767          BR        1$          ; CONTINUE LOOKING
4944 015672 011166 000010    MOV      (R1),10(SP) ; BUFFER ADDRESS TO STACK
4945 015676 166061 000020 000002 3$: SUB      $WDL(R0),2(R1) ; ADJUST BUFFER SIZE
4946 015704 001407          BEQ      4$          ; BR IF DIFFERENCE IS ZERO
4947 015706 006360 000020    ASL      $WDL(R0)    ; CONVERT # WORDS TO BYTES
4948 015712 066011 000020    ADD      $WDL(R0),(R1) ; MAKE NEW STARTING ADDRESS
4949 015716 006260 000020    ASR      $WDL(R0)    ; RETURN # BYTES TO WORDS
4950 015722 000414          BR        6$          ; RETURN
4951 015724 005337 001700    4$: DEC      BUFTBL    ; DECREMENT ENTRIES COUNT
4952 015730 001411          BEQ      6$          ; BR IF ALLOCATION TABLE EMPTY
4953 015732 005302          DEC      R2          ; DECREMENT TABLE COUNT
4954 015734 001407          BEQ      6$          ; BR IF ITEM WERE LAST ENTRY
4955 015736 010103          MOV      R1,R3      ; MOVE TABLE POINTER
4956 015740 062703 000004    ADD      #4,R3      ; POINT TO NEXT ENTRY
4957 015744 012321 5$: MOV      (R3)+,(R1)+    ; MOVE ITEMS
4958 015746 012321          MOV      (R3)+,(R1)+
4959 015750 005302          DEC      R2          ; DECREMENT TABLE COUNT
4960 015752 001374          BNE      5$          ; CONTINUE IF NOT AT END OF TABLE
4961 015754 012603 6$: MOV      (SP)+,R3      ; RESTORE R3
4962 015756 012602          MOV      (SP)+,R2      ; RESTORE R2
4963 015760 012601          MOV      (SP)+,R1      ; RESTORE R1
4964 015762 000207          RTS      PC          ; RETURN
4965
4966
4967          ; ROUTINE TO PUT BUFFER BACK IN TABLE
4968          ; CALL:
4969          ;      MOV      #DPB,R0          ; DPB ADDRESS
4970          ;      JSR      PC,RELBUF
4971          ;      RETURN
4972
4973 015764 010146          RELBUF: MOV      R1,-(SP)    ; SAVE R1
4974 015766 010246          MOV      R2,-(SP)    ; SAVE R2
4975 015770 010446          MOV      R4,-(SP)    ; SAVE R4
4976 015772 010546          MOV      R5,-(SP)    ; SAVE R5
4977 015774 012701 001702    MOV      #BUFTBL+2,R1 ; BEGINNING OF TABLE
4978 016000 013702 001700    MOV      BUFTBL,R2    ; ENTRY COUNT
4979 016004 001424          BEQ      2$          ; BR IF EMPTY TABLE
4980 016006 016003 000020    MOV      $WDL(R0),R3 ; TRIAL ADDRESS
4981 016012 006303          ASL      R3          ; CHANGE TO BYTE COUNT
4982 016014 066003 000006    ADD      $BUF(R0),R3 ; ADDRESS OF HIGHER ADJACENT BLOCK
4983 016020 021103 1$: CMP      (R1),R3          ; UPPER ADJACENT BLOCK
4984 016022 001424          BEQ      4$          ; BR IF YES
4985 016024 062701 000004    ADD      #4,R1      ; INCREMENT POINTER
4986 016030 005302          DEC      R2          ; DECREMENT ENTRY COUNT
4987 016032 001372          BNE      1$          ; CONTINUE SEARCHING
4988 016034 016011 000006    MOV      $BUF(R0),(R1) ; PUT THE BUFFER BLOCK INTO THE TABLE
4989 016040 016061 000020 000002    MOV      $WDL(R0),2(R1) ; BLOCK SIZE
4990 016046 005237 001700    INC      BUFTBL      ; INCREMENT ENTRY COUNT
4991 016052 005202          INC      R2          ; INCREMENT R2 FOR USE LATER
4992 016054 000414          BR        5$          ; SEE IF A LOWER ADJACENT BLOCK IS IN THE TABLE
4993 016056 016021 000006 2$: MOV      $BUF(R0),(R1)+ ; BLOCK ADDRESS TO TABLE
4994 016062 016021 000020    MOV      $WDL(R0),(R1)+ ; SIZE TO TABLE
4995 016066 005237 001700    INC      BUFTBL      ; INCREMENT ENTRY COUNT

```

```

4996 016072 000443          BR      10$          ;EXIT
4997 016074 016011 000006 4$: MOV    $BUF(R0), (R1) ;RELEASED BUFFER IS LOWER ADJACENT
4998 016100 066061 000020 000002 ADD    $WORDL(R0), 2(R1) ;INCREMENTED SIZE
4999 016106 010246          5$: MOV    R2, -(SP) ;SAVE R2
5000 016110 013702 001700      MOV    BUFTBL, R2 ;ENTRY COUNT
5001 016114 012705 001702      MOV    #BUFTBL+2, R5 ;BEGINNING OF TABLE
5002 016120 016504 000002 6$: MOV    2(R5), R4 ;BLOCK SIZE (IN WORDS)
5003 016124 006304          ASL      R4 ;CHANGE TO BYTE COUNT
5004 016126 061504          ADD     (R5), R4 ;ADD BLOCK BEGINNING ADDRESS
5005 016130 020411          CMP     R4, (R1) ;R1 STILL POINTS TO INSERTED ENTRY
5006 016132 001406          BEQ     8$ ;LOWER ADJACENT IN TABLE
5007 016134 062705 000004      ADD     #4, R5 ;INCREMENT POINTER
5008 016140 005302          DEC     R2 ;DECREMENT ENTRY COUNT
5009 016142 001366          BNE     6$ ;CONTINUE LOOKING
5010 016144 005726          TST     (SP)+ ;RESTORE STACK POINTER
5011 016146 000415          BR      10$          ;END
5012 016150 012602          8$: MOV    (SP)+, R2 ;RESTORE R2
5013 016152 066165 000002 000002 ADD    2(R1), 2(R5) ;INCREMENT LOWER BLOCK LENGTH
5014 016160 005337 001700      DEC     BUFTBL ;DECREMENT ENTRY COUNT
5015 016164 010105          MOV     R1, R5 ;GET READY TO COMPRESS
5016 016166 062705 000004      ADD     #4, R5 ;INCREMENT TO NEXT ENTRY
5017 016172 012521          9$: MOV    (R5)+, (R1)+ ;COMPRESS TABLE
5018 016174 012521          MOV    (R5)+, (R1)+ ;MOVE SIZE FIELD DOWN
5019 016176 005302          DEC     R2 ;DECREMENT ENTRY COUNT
5020 016200 001374          BNE     9$ ;BR IF NOT FINISHED
5021 016202 012605          10$: MOV   (SP)+, R5 ;RESTORE R5
5022 016204 012604          MOV   (SP)+, R4 ;RESTORE R4
5023 016206 012602          MOV   (SP)+, R2 ;RESTORE R2
5024 016210 012601          MOV   (SP)+, R1 ;RESTORE R1
5025 016212 000207          RTS     PC ;RETURN
5026
5027
5028 ;FILL THE ASSIGNED BUFFER (IF WRITE OR WRITE CHECK ORDER)
5029 ;CALL:
5030 ;
5031 ;      MOV    #DPB, R0 ;DPB ADDRESS
5032 ;      MOV    #BUFADR, $BUF(R0) ;LOAD BUFFER ADDRESS INTO THE DPB
5033 ;      MOVB   #PATTERN, $PATTC(R0) ;PATTERN CODE
5034 ;      JSR    PC, FILBUF
5035 ;
5036 016214 104412          FILBUF: SAVREG ;SAVE THE REGISTERS
5037 016216 132760 000004 000024 BITB   #BIT02, $CODE(R0) ;SEE IF READ ORDER
5038 016224 001044          BNE     4$ ;BR IF READ
5039 016226 016001 000006 1$: MOV    $BUF(R0), R1 ;BUFFER ADDRESS
5040 016232 016002 000020      MOV    $WORDL(R0), R2 ;POSITIVE WORD COUNT
5041 016236 132760 000001 000024 BITB   #BIT00, $CODE(R0) ;SEE IF WRITE HEADER TYPE ORDER
5042 016244 001413          BEQ     2$ ;BR IF NOT
5043 016246 016011 000012      MOV    $CYL(R0), (R1) ;CYLINDER ADDRESS
5044 016252 052711 010000      BIS    #BIT12, (R1) ;SET FMT22 BIT
5045 016256 052721 140000      BIS    #140000, (R1)+ ;SET MFG AND USER BITS
5046 016262 016021 000010      MOV    $SEC(R0), (R1)+ ;MOVE SECTOR & TRACK
5047 016266 162702 000002      SUB     #2, R2 ;ADJUST THE WORD COUNT
5048 016272 003421          BLE     4$ ;BR IF END OF PATTERN
5049 016274 005004          2$: CLR     R4 ;CLEAR R4
5050 016276 116004 000030      MOVB   $PATTC(R0), R4 ;RELATIVE PATTERN ADDRESS
5051 016302 016405 002476      MOV    STNDAT(R4), R5 ;PATTERN ADDRESS

```

5052	016306	012703	000020			MOV	#20,R3		;PATTERN COUNT
5053	016312	012521				3\$: MOV	(R5)+,(R1)+		;MOVE THE PATTERN INTO THE BUFFER
5054	016314	005302				DEC	R2		;DECREMENT THE WORD COUNT
5055	016316	003407				BLE	4\$		;BR IF DONE (WORD COUNT = 0)
5056	016320	005303				DEC	R3		;DECREMENT THE PATTERN COUNT
5057	016322	001373				BNE	3\$		;BR IF MORE PATTERN
5058	016324	012703	000020			MOV	#20,R3		;RESTORE PATTERN COUNT
5059	016330	016405	002476			MOV	STNDAT(R4),R5		;RESTORE THE ADDRESS
5060	016334	000766				BR	3\$		;CONTINUE DISTRIBUTING THE PATTERN
5061	016336	104413				4\$: RESREG			;RESTORE THE REGISTERS
5062	016340	000207				RTS	PC		;RETURN
5063									
5064									;START THE ORDER FOR THE DPB IN R0
5065						CALL:			
5066						MOV	#DPB,R0		;DPB ADDRESS
5067						JSR	PC,GODRIV		
5068						RETURN			
5069									
5070	016342	010046				GODRIV: MOV	R0,-(SP)		;SAVE R0
5071	016344	010037	016354			MOV	R0,2\$		;CURRENT DPB ADDRESS
5072	016350	004037	035364			1\$: JSR	R0,RM03		;CALL THE DRIVE HANDLER
5073	016354	000000				2\$: .WORD	0		;DRIVE BLOCK ADDRESS GOES HERE
5074	016356	000000				HALT			;DRIVER REJECTED REQUEST
5075	016360	012600				MOV	(SP)+,R0		;RESTORE R0
5076	016362	062760	000001	000036		ADD	#1,\$OPERC(R0)		;INCREMENT THE OPERATION COUNT
5077	016370	005560	000040			ADC	\$OPERC+2(R0)		
5078	016374	026060	000034	000012		CMP	\$PREVA+2(R0),\$CYL(R0)		;DID ORDER REQUIRE A CYLINDER CHANGE
5079	016402	001405				BEQ	3\$		;BR IF NOT
5080	016404	062760	000001	000042		ADD	#1,\$POSIT(R0)		;INCREMENT SEEK COUNT
5081	016412	005560	000044			ADC	\$POSIT+2(R0)		;ADD ANY CARRY
5082	016416	000207				3\$: RTS	PC		
5083									
5084									;GENERATE PARAMETERS FOR THE OPERATION
5085						CALL:			
5086						MOV	#DPB,R0		;DPB ADDRESS
5087						JSR	PC,SELPAR		
5088						RETURN			
5089									
5090	016420	004737	033652			SELPAR: JSR	PC,\$RAND		;CYCLE THE RANDOM NUMBER GENERATOR
5091	016424	032777	000001	162522		BIT	#SW0,\$SWR		;SEE IF SW0 SET
5092	016432	001012				BNE	2\$		;BR IF SET - READ ONLY
5093	016434	012705	000010			1\$: MOV	#10,R5		;READ/WRITE SELECTION DIVISOR
5094	016440	004737	027474			JSR	PC,GETREM		;GET SELECTION VALUE
5095	016444	020537	001500			CMP	R5,RFTIO		;DETERMINE IF READ OR WRITE
5096	016450	103003				BHIS	2\$		;BR IF READ
5097	016452	004737	017276			JSR	PC,RANWRT		;SELECT A WRITE ORDER
5098	016456	000410				BR	THEAD		;SELECT ADDRESS
5099	016460	013705	033752			2\$: MOV	\$LONLM,R5		;SELECT READ OPERATION CODE
5100	016464	042705	177776			BIC	#1C1,R5		;MASK OUT ALL BUT BIT 0
5101	016470	062705	000004			ADD	#4,R5		;TABLE OFFSET FOR READ CODE
5102	016474	110560	000074			MOVB	R5,\$NCODE(R0)		;ORDER SELECTION CODE TO CONTROL BLOCK
5103	016500	005737	001512			THEAD: TST	HEADER		;ENABLE RANDOM ADDRESS SELECT ?
5104	016504	001425				BEQ	RANSEC		;YES
5105	016506	016060	000242	000076		MOV	\$RMDA(R0),\$NSEC(R0)		;SECTOR AND TRACK
5106	016514	016060	000270	000100		MOV	\$RMDC(R0),\$NCYL(R0)		;CYLINDER NUMBER
5107	016522	026060	000100	000106		CMP	\$NCYL(R0),MAXCYL(R0)		;OVER MAX CYLINDER #

```

5108 016530 103520          BLO      XWCNT          ;NO
5109 016532 016060 000110 000100  MOV      MINCYL(RO), $NCYL(RO) ;RESET CYLINDER NUMBER
5110 016540 116060 000120 000076  MOV      MINSEC(RO), $NSEC(RO) ;RESET SECTOR NUMBER
5111 016546 116060 000114 000077  MOV      MINTRK(RO), $NTRK(RO) ;RESET TRACK NUMBER
5112 016554 000137 016772          JMP      XWCNT          ;SELECT BUFFER SIZE
5113
5114          ;GENERATE A RANDOM SECTOR ADDRESS BETWEEN VALUES 'MINSEC' & 'MAXSEC'
5115
5116 016560 016005 000116  RANSEC: MOV      MAXSEC(RO), R5          ;GET MAXIMUM SECTOR ADDRESS
5117 016564 026005 000120          CMP      MINSEC(RO), R5          ;'MINSEC' AND 'MAXSEC' THE SAME ?
5118 016570 001417          BEQ      2$          ;BR IF THEY ARE
5119 016572 166005 000120          SUB      MINSEC(RO), R5          ;SUBTRACT MINIMUM SECTOR ADDRESS
5120 016576 100002          BPL      1$          ;BR IF MAX LARGER THAN MIN
5121 016600 062705 000040          ADD      #32., R5          ;CORRECT THE NUMBER
5122 016604 005205          1$: INC      R5          ;INCREMENT DIFFERENCE TO USE AS DIVISOR
5123 016606 004737 027474          JSR      PC, GETREM          ;GET THE RANDOM AUGMENT
5124 016612 066005 000120          ADD      MINSEC(RO), R5          ;NEW ADDRESS
5125 016616 020527 000037          CMP      R5, #31.          ;IS VALUE TOO LARGE ?
5126 016622 101402          BLOS      2$          ;BR IF NOT
5127 016624 162705 000040          SUB      #32., R5          ;CORRECT VALUE
5128 016630 110560 000076          2$: MOV      R5, $NSEC(RO)          ;STORE SECTOR ADDRESS IN DPB
5129
5130          ;GENERATE A RANDOM TRACK ADDRESS BETWEEN VALUES 'MINTRK' & 'MAXTRK'
5131
5132 016634 016005 000112  RANTRK: MOV      MAXTRK(RO), R5          ;GET MAXIMUM TRACK ADDRESS
5133 016640 026005 000114          CMP      MINTRK(RO), R5          ;'MINTRK' AND 'MAXTRK' THE SAME ?
5134 016644 001417          BEQ      2$          ;BR IF THEY ARE
5135 016646 166005 000114          SUB      MINTRK(RO), R5          ;SUBTRACT MINIMUM TRACK ADDRESS
5136 016652 100002          BPL      1$          ;BR IF MAX LARGER THAN MIN
5137 016654 062705 000005          ADD      #5., R5          ;CORRECT THE NUMBER
5138 016660 005205          1$: INC      R5          ;INCREMENT DIFFERENCE TO USE AS DIVISOR
5139 016662 004737 027474          JSR      PC, GETREM          ;GET THE RANDOM AUGMENT
5140 016666 066005 000114          ADD      MINTRK(RO), R5          ;NEW TRACK ADDRESS
5141 016672 020527 000004          CMP      R5, #4.          ;IS VALUE TOO LARGE ?
5142 016676 101402          BLOS      2$          ;BR IF NOT
5143 016700 162705 000005          SUB      #5., R5          ;CORRECT VALUE
5144 016704 110560 000077          2$: MOV      R5, $NTRK(RO)          ;STORE TRACK ADDRESS IN DPB
5145
5146          ;GENERATE A RANDOM CYLINDER ADDRESS BETWEEN VALUES 'MINCYL' & 'MAXCYL'
5147
5148 016710 016037 000106 001446  RANCYL: MOV      MAXCYL(RO), CYLIMT          ;ASSUME AN RM03
5149 016716 016005 000106          MOV      MAXCYL(RO), R5          ;GET MAXIMUM CYLINDER ADDRESS
5150 016722 026005 000110          CMP      MINCYL(RO), R5          ;'MINCYL' AND 'MAXCYL' THE SAME ?
5151 016726 001417          BEQ      2$          ;BR IF THEY ARE
5152 016730 166005 000110          SUB      MINCYL(RO), R5          ;SUBTRACT MINIMUM CYLINDER ADDRESS
5153 016734 100002          BPL      1$          ;BR IF MAX LARGER THAN MIN
5154 016736 063705 001446          ADD      CYLIMT, R5          ;CORRECT THE NUMBER
5155 016742 005205          1$: INC      R5          ;INCREMENT DIFFERENCE TO USE AS DIVISOR
5156 016744 004737 027474          JSR      PC, GETREM          ;GET THE RANDOM AUGMENT
5157 016750 066005 000110          ADD      MINCYL(RO), R5          ;NEW CYLINDER ADDRESS
5158 016754 023705 001446          CMP      CYLIMT, R5          ;IS VALUE TOO LARGE ?
5159 016760 003002          BGT      2$          ;BR IF NOT
5160 016762 016005 000110          MOV      MINCYL(RO), R5          ;CORRECT VALUE
5161 016766 010560 000100          2$: MOV      R5, $NCYL(RO)          ;STORE CYLINDER ADDRESS IN DPB
5162 016772 132760 000001 000074  YWCNT: MOV      #BIT00, $NCODE(RO) ;HEADER OPERATION INVOLVED ?
5163 017000 001414          BEG      RANSIZ          ;NO

```

```

5164 017002 012760 000402 000102 MOV #258,$NWRDL(R0) ;CHANGE WORD LENGTH TO 258 FOR WRTHD ORDER
5165 017010 122760 000005 0C0074 3$: CMPB #5,$NCODE(R0) ;READ HEADER AND DATA ?
5166 017016 001461 BEQ RANXIT ;YES
5167 017020 004537 017172 JSR R5,CHKADR ;IF WRITE HEAD AND DATA COMMAND
5168 ;AVOID WRITING BAD SPOT HEADER
5169 017024 000452 BR RANPAT ;BRANCH IF NOT ON BAD SPOT
5170 017026 000137 016420 JMP SELPAR ;SELECT THE PARAMETERS AGAIN
5171
5172 ;GENERATE A RANDOM BUFFER LENGTH BETWEEN 4 & THE VALUE IN 'MAXDL'
5173
5174 017032 013705 001462 RANSIZ: MOV MAXDL,R5 ;GET BUFFER SIZE
5175 017036 005737 001476 TST WSEL ;SELECT A RANDOM WORD COUNT ?
5176 017042 001010 BNE 1$ ;BR IF NOT
5177 017044 005205 INC R5 ;INCREMENT THE MAXIMUM SIZE
5178 017046 004737 027474 JSR PC,GETREM ;DIVIDE BY MAX VALUE
5179 017052 005705 TST R5 ;IS THE REMAINDER 0 ?
5180 017054 001003 BNE 1$ ;NOT 0, CONTINUE
5181 017056 004737 033652 JSR PC,$RAND ;CYCLE THE RANDOM NUMBER GENERATOR
5182 017062 000763 BR RANSIZ ;TRY AGAIN
5183 017064 022705 000004 1$: CMP #4,R5 ;LESS THAN 4 ?
5184 017070 003403 BLE 2$ ;NO
5185 017072 012705 000004 MOV #4,R5 ;SET SIZE TO 4
5186 017076 000405 BR 3$
5187 017100 023705 001462 2$: CMP MAXDL,R5 ;LARGE THAN MAX ALLOWED ?
5188 017104 101002 BHI 3$ ;NO
5189 017106 013705 001462 MOV MAXDL,R5 ;RESET COUNTER
5190 017112 132760 000004 000074 3$: BITB #BIT02,$NCODE(R0) ;READ OPERATION ?
5191 017120 001006 BNE 4$ ;YES
5192 017122 042705 000377 BIC #377,R5 ;SECTOR BOUNDARY FOR WRITE OP
5193 017126 005705 TST R5 ;NONE
5194 017130 003002 BGT 4$ ;NO
5195 017132 012705 000400 MOV #400,R5 ;AT LEAST ONE SECTOR
5196 017136 010560 000102 4$: MOV #5,$NWRDL(R0) ;WORD COUNT
5197 017142 132760 000004 000074 BITB #BIT02,$NCODE(R0) ;READ OP ?
5198 017150 001004 BNE RANXIT ;YES
5199 ;GET A RANDOM PATTERN NUMBER
5200
5201 017152 004737 017332 RANPAT: JSR PC,GETPAT ;GET PATTERN CODE
5202 017156 110560 000075 MOVB R5,$NPATC(R0) ;MOVE PATTERN CODE TO CONTROL BLOCK
5203 017162 012760 177777 000104 RANXIT: MOV #-1,$NEXT(R0) ;SET PARAMETERS SELECTED INDICATOR
5204 017170 000207 RTS PC ;RETURN
5205
5206 ;ROUTINE TO CHECK THE SELECTED ADDRESS IS NOT ON THE BAD SPOT
5207 ;CALLING SEQ
5208 ;RO=DPB ADDRESS
5209 JSR R5,CHKADR
5210 RET1 ;NORMAL RETURN
5211 RET2 ;ERROR RET
5212
5213 017172 CHKADR: MOV R1,-(SP) ;PUSH R1 ON STACK
5214 017172 010146 MOV R2,-(SP) ;PUSH R2 ON STACK
5215 017174 010246 MOV R3,-(SP) ;PUSH R3 ON STACK
5216 017176 010346 MOV #16,R1 ;TOTAL 16 SETS OF BAD SECTOR ADDRESSES
5217 017200 012701 000020 MOV R0,R2 ;TABLE ADDRESS
5218 017204 010002 MOV #80SEC,R2 ;DBP ADDRESS + TABLE ADDRES OFFSET
5219 017206 062702 000124 ADD

```

```

5220 017212 022712 177777      1$: CMP      #1,(R2)      ;EMPTY ENTRY ?
5221 017216 001423              BEQ      3$          ;BRANCH IF SO
5222 017220 026012 000100      CMP      $NCYL(R0),(R2) ;ON THE SAME CYLINDER ?
5223 017224 001013              BNE      2$          ;BRANCH IF NOT
5224 017226 126062 000011 000003  CMPB    $TRK(R0),3(R2) ;ON THE SAME TRACK ?
5225 017234 001007              BNE      2$          ;BRANCH IF NOT
5226 017236 126062 000010 000002  CMPB    $SEC(R0),2(R2) ;ON THE SAME SECTOR ?
5227 017244 001003              BNE      2$          ;BRANCH IF NOT
5228 017246 062705 000002      ADD      #2,R5      ;OTHERWISE,TAKE ERROR EXIT
5229 017252 000405              BR       3$          ;EXIT
5230 017254 005301      2$: DEC      R1          ;ALL 16 BAD SPOT CHECKED ?
5231 017256 003403              BLE      3$          ;BRANCH IF SO
5232 017260 062702 000004      ADD      #4,R2      ;ADJUST TO NEXT TABLE ENTRY
5233 017264 000752              BR       1$          ;LOOP BACK
5234 017266              3$:
5235 017266 012603              MOV      (SP)+,R3      ;POP STACK INTO R3
5236 017270 012602              MOV      (SP)+,R2      ;POP STACK INTO R2
5237 017272 012601              MOV      (SP)+,R1      ;POP STACK INTO R1
5238 017274 000205              RTS      R5          ;RETRUN
5239
5240 ;ROUTINE TO SELECT A WRITE (OR WRITE HEAD AND DATA) OPERATION
5241
5242 017276 012705 000002      RANWRT: MOV    #2,R5      ;SET WRITE DATA COMMAND
5243 017302 005737 001474      TST      FORMAT      ;ALLOW FORMAT OPERATION ?
5244 017306 001406              BEQ      1$          ;NO
5245 017310 122760 000004 000024  CMPB    #4,$CODE(R0) ;PREVIOUS A READ DATA COMMAND ?
5246 017316 001002              BNE      1$          ;NO
5247 017320 012705 000003      MOV      #3,R5      ;SET A WRITE HEAD AND DATA COMMAND
5248 017324 110560 000074      1$: MOVB   R5,$NCODE(R0) ;SELECT THE WRITE COMMAND
5249 017330 000207              RTS      PC          ;EXIT
5250
5251 ;ROUTINE TO SELECT A PATTERN
5252
5253 017332 012705 000020      GETPAT: MOV    #20,R5     ;SELECT PATTERN
5254 017336 005737 001510      TST      PATTEN      ;ENABLE RANDOM PATTERN SELECTION ?
5255 017342 001403              BEQ      2$          ;YES
5256 017344 013705 001510      MOV      PATTEN,R5    ;USE INDEXED PATTERN
5257 017350 000407              BR       1$          ;NO
5258 017352 004737 027474      2$: JSR      PC,GETREM    ;GET CODE
5259 017356 005705              TST      R5          ;WAS PATTERN ZERO SELECTED ?
5260 017360 001003              BNE      1$          ;BR IF NOT ZERO
5261 017362 004737 033652      JSR      PC,$RAND      ;CYCLE THE RANDOM NUMBER GENERATOR
5262 017366 000761              BR       GETPAT      ;TRY AGAIN
5263 017370 006305              1$: ASL      R5          ;MAKE CODE INTO TABLE INDEX
5264 017372 000207              RTS      PC
5265
5266 ;ROUTINE TO GET THE PREVIOUSLY SELECTED PARAMETER VALUES
5267 ;CALL:
5268 ;
5269 ;
5270 ;
5271 ;
5272 ;
5273 017374 010546      GETPAR: MOV    R5,-(SP)      ;SAVE R5
5274 017376 116060 000234 000027  MOVB    $RACSI(R0),$PREV0(R0) ;SAVE CURRENT PARAMETERS
5275 017404 142760 177701 000027  BICB    #1C76,$PREV0(R0) ;STRIP GO,AND IE BITS

```



```

5276 017412 032760 000006 000074      BIT      #6,$NCODE(R0) ;SEE IF NEXT OPERATION IS READ OR WRITE
5277 017420 001007      BNE      1$      ;BR IF EITHER
5278 017422 016060 000012 000034      MOV      $CYL(R0),$PREVA+2(R0) ;SAVE STARTING CYLINDER
5279 017430 016060 000010 000032      MOV      $SEC(R0),$PREVA(R0) ;SAVE STARTING SECTOR AND TRACK
5280 017436 000410      BR        22$
5281 017440 004737 022502      1$:      JSR      PC,READR ;GET THE DECREMENTED SECTOR AND TRACK ADDRESSES
5282 017444 012660 000034      MOV      (SP)+,$PREVA+2(R0) ;CYLINDER ADDRESS
5283 017450 112660 000033      MOV      (SP)+,$PREVA+1(R0) ;TRACK ADDRESS
5284 017454 112660 000032      MOV      (SP)+,$PREVA(R0) ;SECTOR ADDRESS
5285      MOV      $CYL(R0),$PREVA+2(R0) ;CURRENT CYLINDER
5286 017460 032777 000100 161466      22$:      BIT      #SW06,$SWR ;SWITCH 6 SET ?
5287 017466 001073      BNE      5$      ;BR IF SET
5288 017470 116060 000074 000024      MOV      $NCODE(R0),$CODE(R0) ;LOGICAL CODE FOR OPERATION
5289 017476 116005 000074      MOV      $NCODE(R0),R5 ;LOAD R5 FOR USE AS TABLE INDEX
5290 017502 116560 002122 000002      MOV      COMTBL(R5),$COMND(R0) ;RM03 COMMAND CODE
5291 017510 122760 000151 000002      CMP      #151,$COMND(R0) ;WRITE CHECK DATA COMMAND ?
5292 017516 001013      BNE      3$      ;NO,DONOT CARE
5293 017520 122760 000060 000027      CMP      #60,$PREVO(R0) ;PREVIOUS A WRITE DATA COMMAND ?
5294 017526 001420      BEQ      4$      ;YES,OK
5295 017530 112760 000171 000002      2$:      MOV      #171,$COMND(R0) ;CHANG TO READ DATA COMMAND
5296 017536 112760 000004 000024      MOV      #4,$CODE(R0) ;CODE NUMBER CHANGED TO READ DATA
5297 017544 000411      BR        4$      ;EXIT
5298 017546 122760 000153 000002      3$:      CMP      #153,$COMND(R0) ;WRITE CHECK HEAD AND DATA COMMAND ?
5299 017554 001005      BNE      4$      ;NO,THEN EXIT
5300 017556 122760 000062 000027      CMP      #62,$PREVO(R0) ;PREVIOUS A WRITE HEAD AND DATA COMMAND ?
5301 017564 001401      BEQ      4$      ;YES,EXIT
5302 017566 000760      BR        2$      ;SET TO READ DATA COMMAND
5303 017570      4$:
5304 017570 116060 000075 000030      MOV      $NPATC(R0),$PATTC(R0) ;PATTERN CODE
5305 017576 016060 000076 000010      MOV      $NSEC(R0),$SEC(R0) ;TRACK AND SECTOR ADDRESSES
5306 017604 016060 000100 000012      MOV      $NCYL(R0),$CYL(R0) ;CYLINDER ADDRESS
5307 017612 016060 000102 000020      MOV      $NWRDL(R0),$WRDL(R0) ;BUFFER SIZE
5308 017620 016060 000102 000004      MOV      $NWRDL(R0),$WRDL(R0) ;WORD COUNT FOR THE RH11
5309 017626 005460 000004      NEG      $WRDL(R0) ;COMPLEMENT IT
5310 017632 012760 000400 000022      MOV      #256,$SSEC(R0) ;INITIAL VALUE OF SECTOR SIZE
5311 017640 032760 000001 000024      BIT      #1,$CODE(R0) ;HEADER OPERATION ?
5312 017646 001403      BEQ      5$      ;BR IF NOT
5313 017650 062760 000002 000022      ADD      #2,$SSEC(R0) ;ADD HEADER SIZE
5314 017656 005060 000104      5$:      CLR      $NEXTR(R0) ;RESET 'PARAMETERS LOADED' INDICATOR
5315 017662 012605      MOV      (SF)+,R5 ;RESTORE R5
5316 017664 000207      RTS      PC ;RETURN
5317
5318 ;ROUTINE TO COMPRESS A LIST
5319 ;CALL:
5320 ;      MOV      #ADDRS,R1 ;COMPRESS LIST STARTING AT THIS ADDRESS
5321 ;      JSR      PC,COMPRES
5322 ;      RETURN
5323
5324 017666 016111 000002      COMPRES: MOV      2(R1),(R1) ;COMPRESS THE TABLE IN R1
5325 017672 001403      BEQ      1$      ;BR WHEN ZERO FOUND
5326 017674 062701 000002      ADD      #2,R1 ;INCREMENT R1
5327 017700 000772      BR        COMPRES ;CONTINUE COMPRESSING TABLE
5328 017702 000207      1$:      RTS      PC ;RETURN
5329
5330 ;ROUTINE TO SETUP PARAMETERS FOR A SEQUENTIAL READ OR WRITE OF THE DISK
5331 ;CALL:

```

```

5332      :      MOV      #DPB,RO      ;DPB ADDRESS
5333      :      MOV      #-1,$PACK(RO) ;'WRITE PACK' FLAG
5334      :      OR
5335      :      MOV      #1,$PACK(RO)   ;'READ PACK' FLAG
5336      :      JSR      PC,WRTPK
5337      :      RETURN
5338
5339 017704 004737 033652      WRTPK: JSR      PC,$RAND      ;CYCLE THE RANDOM NUMBER GENERATOR
5340 017710 005760 000040      TST      $OPERC+2(RO)      ;SEE IF FIRST OPERATION
5341 017714 001007      BNE      WRTPK1      ;BR IF UPPER WORD OF COUNTER NOT ZERO
5342 017716 005760 000036      TST      $OPERC(RO)      ;LOWER WORD ZERO ?
5343 017722 001004      BNE      WRTPK1      ;BR IF NOT 1ST OPERATION
5344 017724 105760 000026      TSTB     $PACK(RO)      ;SEE WHICH - 'R' OR 'W'
5345 017730 100530      BMI      WRTPK3      ;BR IF 'W'
5346 017732 000515      BR         WRTPK2      ;'R' OPERATION
5347 017734 116060 000234 000027 WRTPK1: MOVB     $RMCS1(RO),$PREV0(RO) ;SAVE CURRENT PARAMETERS
5348 017742 004737 022502      JSR      PC,READDR      ;GET THE DECREMENTED SECTOR AND TRACK ADDRESSES
5349 017746 012660 000034      MOV      (SP)+,$PREVA+2(RO) ;CYLINDER ADDRESS
5350 017752 112660 000033      MOVB     (SP)+,$PREVA+1(RO) ;TRACK ADDRESS
5351 017756 112660 000032      MOVB     (SP)+,$PREVA(RO) ;SECTOR ADDRESS
5352 017762 016060 000242 000010      MOV      $RMDA(RO),$SEC(RO) ;NEW SECTOR & TRACK ADDRESS
5353 017770 016060 000270 000012      MOV      $RMDC(RO),$CYL(RO) ;NEW CYLINDER ADDRESS
5354 017776 026060 000012 000106      CMP      $CYL(RO),MAXCYL(RO) ;SEE IF AT END
5355 020004 103436      BLO      2$      ;BR IF LESS THAN 'MAXCYL'
5356 020006 101004      BHI      1$      ;BR IF GREATER THAN 'MAXCYL'
5357 020010 126060 000011 000112      CMPB     $TRK(RO),MAXTRK(RO) ;SEE IF AT MAX TRACK
5358 020016 103431      BLO      2$      ;BR IF NOT GREATER
5359 020020 116060 000114 000011 1$: MOVB     MINTRK(RO),$TRK(RO) ;RESET TRACK ADDRESS
5360 020026 116060 000120 000010      MOVB     MINSEC(RO),$SEC(RO) ;RESET SECTOR ADDRESS
5361 020034 016060 000110 000012      MOV      MINCYL(RO),$CYL(RO) ;RESET CYLINDER ADDRESS
5362 020042 112760 000004 000024      MOVB     #4,$CODE(RO) ;SET CODE TO READ DATA
5363 020050 122760 177776 000026      CMPB     #-2,$PACK(RO) ;WT OPERATION IN PROCESSING
5364 020056 001475      BEQ      WRTPK5      ;YES
5365 020060 004737 027166      JSR      PC,EOP2      ;DROP THE DRIVE (NORMAL TERMINATION)
5366 020064 032777 000020 161062      BIT      #SW04,$SWR      ;IS SWITCH 4 SET ?
5367 020072 001003      BNE      2$      ;BR IF SET
5368 020074 005726      TST      (SP)+      ;INCREMENT THE STACK POINTER
5369 020076 000137 005420      JMP      MAIN      ;RETURN DIRECTLY TO 'MAIN'
5370 020102 013760 001462 000020 2$: MOV      MAXDL,$WRDL(RO) ;BUFFER SIZE IS MAXIMUM
5371 020110 042760 000377 000020      BIC      #377,$WRDL(RO) ;SECTOR BOUNDARY FOR WRITTING
5372 020116 013760 001462 000004      MOV      MAXDL,$WRDM(RO) ;WORD COUNT
5373 020124 042760 000377 000004      BIC      #377,$WRDM(RO) ;SECTOR BOUNDARY FOR WRITTING
5374 020132 005760 000004      TST      $WRDM(RO) ;SIZE=0 ?
5375 020136 003006      BGT      3$      ;NO
5376 020140 012760 000400 000004      MOV      #400,$WRDM(RO) ;SET TO ONE SECTOR
5377 020146 012760 000400 000020      MOV      #400,$WRDL(RO) ;SET ONE SECTOR
5378 020154 005460 000004 3$: NEG      $WRDM(RO) ;CHANGE WORD COUNT TO 2'S COMPLEMENT
5379 020160 105760 000026      TSTB     $PACK(RO) ;READ OR WRITE ?
5380 020164 100412      BMI      WRTPK3      ;BR IF WRITE
5381 020166 012760 000402 000022 WRTPK2: MOV      #258,$SSEC(RO) ;SECTOR SIZE FOR READ
5382 020174 112760 000005 000024      MOVB     #5,$CODE(RO) ;CODE FOR READ HEADER & DATA
5383 020202 112760 000173 000002      MOVB     #R0HD,$COMND(RO) ;DRIVE CODE FOR OPERATION
5384 020210 000415      BR         WRTPK4      ;SET UP FOR EXIT
5385 020212 012760 000400 000022 WRTPK3: MOV      #256,$SSEC(RO) ;SECTOR SIZE
5386 020220 112760 000002 000024      MOVB     #2,$CODE(RO) ;CODE FOR WRTDAT
5387 020226 112760 000161 000002      MOVB     #WRTDAT,$COMND(RO) ;OP CODE

```



```

5388 020234 004737 017332      JSR    PC,GETPAT      ;GET PATTERN CODE
5389 020240 110560 000030      MOV     R5,$PATTC(R0)    ;PATTERN CODE
5390 020244 005060 000104      WRTPK4: CLR    $NEXT(R0)    ;CLEAR 'PARAMETER SELECTED' INDICATOR
5391 020250 000207              RTS     PC                ;RETURN
5392 020252 005037 001316      WRTPK5: CLR    PACK          ;CLEAR WT FLAG
5393 020256 105060 000026      CLR     $PACK(R0)    ;CLEAR WT FLAG
5394 020262 005726              TST     (SP)+             ;CLEAR STACK LEVEL
5395 020264 000137 005420      JMP     MAIN          ;JUMP TO MAIN BACKGROUND LOOP
5396
5397      ;ROUTINE TO DETERMINE OF ERROR IS AT A LOCATION ON THE PACK DEFINED
5398      ;IN THE BAD TRACK/SECTOR TABLE FOR THE DRIVE.
5399      ;CALL:
5400      ;      JSR    PC,SPOTCK
5401      ;      RETURN1      ;ERROR AT AN ADDRESS IN TABLE
5402      ;      RETURN2      ;NO TABLE ENTRY FOR ERROR ADDRESS OR
5403      ;                  ;PARAMETER 'NOTPRT' IS 0
5404
5405      SPOTCK:
5406      MOV     R1,-(SP)      ;PUSH R1 ON STACK
5407      MOV     R2,-(SP)      ;PUSH R2 ON STACK
5408      MOV     #500SEC,R1    ;INCREMENT FOR BAD SECTOR TABLE
5409      ADD     R0,R1          ;ADD THE BLOCK'S STARTING ADDRESS
5410      MOV     #16,R2        ;BAD SECTOR TABLE SIZE COUNT
5411      JSR     PC,READDR     ;DECREMENT THE SECTOR/TRACK ADDRESS
5412      CMP     (R1),(SP)+    ;ON THE SAME CYLINDER ?
5413      BNE     2$            ;BRANCH IF NOT
5414      CMPB    (SP)+,3(R1)    ;COMPARE THE TRACK ADDRESS
5415      BNE     3$            ;BR IF IT IS NOT EQUAL
5416      CMPB    (SP)+,2(R1)    ;COMPARE THE SECTOR ADDRESS
5417      BNE     4$            ;BR IF NOT EQUAL
5418      BR      5$            ;CHECK 'NOTPRT'
5419      TST     (SP)+          ;CLEAR OFF THE STACK
5420      TST     (SP)+          ;INCREMENT THE STACK POINTER
5421      ADD     #4,R1          ;GO TO THE NEXT LOCATION IN THE TABLE
5422      TST     (R1)          ;PAST THE TABLE ENTRIES ?
5423      BMI     6$            ;BR IF PAST
5424      DEC     R2            ;DECREMENT THE MAXIMUM ENTRY COUNT
5425      BNE     1$            ;BR IF MORE TO CHECK
5426      BR      6$            ;END, EXIT
5427      TST     NOTPRT        ;PRINT THE ERROR ANYWAY ?
5428      BNE     7$            ;BR IF NOT
5429      MOV     #-1,BADSEC     ;SET THE INDICATOR FOR THE IDENTIFICATION LINE
5430      ADD     #2,4(SP)       ;INCREMENT THE RETURN
5431      7$:
5432      MOV     (SP)+,R2        ;POP STACK INTO R2
5433      MOV     (SP)+,R1        ;POP STACK INTO R1
5434      RTS     PC            ;RETURN
5435
5436      ;*****
5437      .SBTTL  ERROR MESSAGE GENERATION ROUTINES
5438
5439      ;*****
5440
5441      ;PRINT LINE 1 OF ERROR MESSAGE:
5442      ;'HH:MM:SS'
5443

```

```

5444 020406 032777 002000 160540 LINE1: BIT      #SW10,2SWR      ;SWITCH 10 SET ?
5445 020414 001402                BEQ      1$              ;BR IF NOT
5446 020416 104401                TYPE     $BELL             ;RING THE BELL
5447 020422 032777 020000 160524 1$: BIT      #SW13,2SWR      ;INHIBIT TIMEOUT ?
5448 020430 001407                BEQ      2$              ;BR IF NOT
5449 020432 104414 001201        DISPLY    , $CRLF           ;CR-LF
5450 020436 104414 001201        DISPLY    , $CRLF           ;CR-LF
5451 020442 104414 001201        DISPLY    , $CRLF           ;CR-LF
5452 020446 000404                BR       3$              ;EXIT
5453 020450 004737 023670        JSR      PC,$TIME          ;TYPE THE TIME
5454 020454 104414 053247        DISPLY    ,LINSPO          ;SPACES
5455 020460 000207                RTS       PC              ;RETURN & TYPE DESCRIPTION
5456
5457
5458
5459
5460
5461
5462
5463
5464
5465
5466
5467
5468 020462
5469 020462 010346                LINE2: MOV      R3,-(SP)      ;: PUSH R3 ON STACK
5470 020464 010446                MOV      R4,-(SP)      ;: PUSH R4 ON STACK
5471 020466 010546                MOV      R5,-(SP)      ;: PUSH R5 ON STACK
5472 020470 104414 001201        DISPLY    , $CRLF           ;: CR-LF
5473 020474 005037 020622        CLR      4$              ;: CLEAR MESSAGE ADDRESS STORAGE
5474 020500 005004                CLR      R4              ;: WORKING REGISTER
5475 020502 012737 051352 020622 MOV      #LIN2C,4$      ;: ADDRESS OF 'PRESENT ORDER = ' MSG
5476 020510 116004 000234        MOV      $RMCS1(R0),R4      ;: GET THE OPCODE
5477 020514 042704 177701        BIC      #1C76,R4        ;: SAVE ONLY SIGNIFICANT BITS
5478 020520 004737 020556        JSR      PC,1$          ;: TYPE THE FIRST MNEMONIC
5479 020524 005737 020626        TST      5$              ;: SEE IF MNEMONIC ENTRY FOUND
5480 020530 001440                BEQ      LINE2A          ;: BR IF NOT
5481 020532 012737 051373 020622 MOV      #LIN2P,4$      ;: ADDRESS OF 'PREVIOUS ORDER = ' MSG
5482 020540 116004 000027        MOV      $PREV0(R0),R4      ;: PREVIOUS OPERATION CODE
5483 020544 042704 177701        BIC      #1C76,R4        ;: SAVE ONLY SIGNIFICANT BITS
5484 020550 004737 020556        JSR      PC,1$          ;: TYPE THE PREVIOUS MNEMONIC
5485 020554 000426                BR       LINE2A          ;: CONTINUE
5486 020556 005005                1$: CLR      R5              ;: CLEAR THE TABLE INDEX
5487 020560 126504 002130        2$: CMPB    OPTBL(R5),R4      ;: LOOK FOR THE OPCODE
5488 020564 001405                BEQ      3$              ;: BR WHEN OPCODE COUNT EQUALS OPCODE
5489 020566 105765 002130        TSTB     OPTBL(R5)        ;: LOOK FOR END OF TABLE
5490 020572 100402                BMI      3$              ;: BR IF END
5491 020574 005205                INC      R5              ;: INCREMENT THE POINTER
5492 020576 000770                BR       2$              ;: CONTINUE - NOT END OF TABLE
5493 020600 006305                3$: ASL      R5              ;: SHIFT INDEX
5494 020602 006305                ASL      R5              ;: SHIFT THE INDEX
5495 020604 006305                ASL      R5              ;: SHIFT THE INDEX
5496 020606 012737 002152 020626 MOV      #MNTBL,5$      ;: ADDRESS OF ASCII TEXT TABLE
5497 020614 060537 020626        ADD      R5,5$          ;: ADD THE INDEX
5498 020620 104414                DISPLY    .WORD 0          ;: TYPE IT
5499 020622 000000                4$:

```

5500	020624	104414		DISPLY		;TYPE THE OPERATION MNEMONIC
5501	020626	000000		SS: .WORD	0	;ADDRESS OF MESSAGE
5502	020630	000207		RTS	PC	;RETURN TO MAIN ROUTINE
5503	020632	005737	001362	LINE2A: TST	BADSEC	;PRINT THE BAD SECTOR LINE ?
5504	020636	001404		BEQ	LINE2B	;BR IF NOT
5505	020640	104414	001201	DISPLY	, \$CRLF	;CR-LF
5506	020644	104414	051417	DISPLY	, LIN2S	;ERROR ADDRESS DEFINED AS BAD AREA
5507	020650	104414	001201	LINE2B: DISPLY	, \$CRLF	;CR-LF
5508	020654	104414	050746	DISPLY	, DH14	;STANDARD RM03 REGISTER HEADER
5509	020660	104414	053247	DISPLY	, LINSPO	;TYPE A SPACE
5510	020664	013746	001344	MOV	UNIT, -(SP)	;PUT THE DRIVE NUMBER ON THE STACK
5511	020670	004737	022462	JSR	PC, LINDEC	;TYPE DRIVE NUMBER
5512	020674	104414	053246	DISPLY	, LINSPO	;SPACES
5513	020700	012705	051272	MOV	DOT14, R5	;REGISTER INDEXES
5514	020704	004737	021034	JSR	PC, 3\$	;PRINT THE REGISTERS
5515	020710	032777	000040	BIT	\$SW05, \$SWR	;PRINT THE OPTIONAL REGISTERS ?
5516	020716	001014		BNE	1\$	;BR IF NOT
5517	020720	104414	051051	DISPLY	, DH15	
5518	020724	012705	051314	MOV	DOT15, R5	;SECOND DATA LINE
5519	020730	004737	021034	JSR	PC, 3\$	;PRINT THEM
5520	020734	104414	051150	DISPLY	, DH16	
5521	020740	012705	051336	MOV	DOT16, R5	;THIRD DATA LINE
5522	020744	004737	021034	JSR	PC, 3\$	;PRINT THE REGISTERS
5523	020750	032760	000100	1\$: BIT	\$BIT6, \$STATUS(R0)	;DATA ERROR ?
5524	020756	001422		BEQ	2\$	;BR IF NOT
5525	020760	016046	000020	MOV	\$WDL(R0), -(SP)	;TRANSFER SIZE
5526	020764	066016	000236	ADD	\$RMWC(R0), (SP)	;ADD REMAINING WORD COUNT
5527	020770	006316		ASL	(SP)	;CONVERT TO AN BYTE INCREMENT
5528	020772	066016	000006	ADD	\$BUF(R0), (SP)	;BUFFER STARTING ADDRESS
5529	020776	022660	000240	CMP	(SP)+, \$RMB(R0)	;CORRECT BUFFER ADDRESS ?
5530	021002	001410		BEQ	2\$	;BR IF YES
5531	021004	104414	050373	DISPLY	, EM46	;BUS ADDRESS AND WORD COUNT ARE NOT CONSISTENT
5532	021010	104414	001201	DISPLY	, \$CRLF	;CR-LF
5533	021014	004737	021126	JSR	PC, LINE3D	;PRINT LINE 3D OF ERROR MESSAGE
5534	021020	004737	021540	JSR	PC, LINE4	;PRINT LINE 4 OF ERROR MESSAGE
5535	021024			2\$: MOV	(SP)+, R5	;POP STACK INTO R5
5536	021024	012605		MOV	(SP)+, R4	;POP STACK INTO R4
5537	021026	012604		MOV	(SP)+, R3	;POP STACK INTO R3
5538	021030	012603		RTS	PC	;RETURN TO ERROR PROCESSING ROUTINE
5539	021032	000207		3\$: MOV	(R5)+, -(SP)	;PUT THE REGISTER INDEX ON THE STACK
5540	021034	012546		ADD	R0, (SP)	;ADD DRIVE'S TABLE ADDRESS
5541	021036	060016		MOV	2(SP), -(SP)	;VALUE
5542	021040	017646	000000	JSR	PC, LINOCT	;TYPE IT
5543	021044	004737	022430	TST	(SP)+	;CORRECT THE STACK POINTER
5544	021050	005726		DISPLY	, LINSPO	;PRINT 2 SPACES
5545	021052	104414	053246	TST	(R5)	;AT END OF LINE ?
5546	021056	005715		BNE	3\$	;BR IF NOT
5547	021060	001365		4\$: DISPLY	, \$CRLF	;CR-LF
5548	021062	104414	001201	RTS	PC	;RETURN
5549	021066	000207				
5550						
5551						;PRINT LINE 3 OF ERROR MESSAGE
5552						; 'ERROR AT CCC TT SS' PREVIOUS ADR = CCC TT SS'
5553						
5554	021070	104414	051453	LINE3: DISPLY	, LINM3	;LINE 3 ENTRANCE
5555	021074	000517		BR	LIN3.1	;FINISH PRINTOUT

```
5556  
5557  
5558  
5559  
5560 021076 104414 051471  
5561 021102 000514  
5562  
5563  
5564  
5565  
5566 021104 004737 021442  
5567 021110 104414 001201  
5568 021114 000207  
5569  
5570  
5571  
5572  
5573 021116 004737 021442  
5574 021122 000137 021474  
5575  
5576  
5577  
5578  
5579 021126 032777 000040 160020  
5580 021134 001416  
5581 021136 104414 051642  
5582 021142 016046 000240  
5583 021146 004737 022430  
5584 021152 104414 051652  
5585 021156 016046 000236  
5586 021162 004737 022430  
5587 021166 104414 001201  
5588 021172 000207  
5589  
5590  
5591  
5592  
5593 021174 104414 051536  
5594 021200 016046 000012  
5595 021204 004737 022462  
5596 021210 104414 053246  
5597 021214 104414 051664  
5598 021220 005046  
5599 021222 116016 000011  
5600 021226 004737 022462  
5601 021232 104414 053246  
5602 021236 104414 051701  
5603 021242 005046  
5604 021244 116016 000010  
5605 021250 004737 022462  
5606 021254 104414 001201  
5607 021260 000207  
5608  
5609  
5610  
5611
```

```
;PRINT LINE 3A OF ERROR MESSAGE  
;'START CYL = CCC END CYL = CCC'  
LINE3A: DISPLY LINN3 ;LINE 3A ENTRANCE  
BR LIN3.1 ;FINISH ERROR LINE  
;PRINT LINE 3B OF ERROR MESSAGE  
;'START CYL = CCC END CYL = CCC ACTUAL CYL = CCC'  
LINE3B: JSR PC,LIN3.3 ;LINE 3B ENTRANCE  
DISPLY $CRLF  
RTS PC  
;PRINT LINE 3C OF ERROR MESSAGE  
;'START CYL = CCC END CYL = CCC ACTUAL CYL = CCC TRK = TT'  
LINE3C: JSR PC,LIN3.3 ;LINE 3C ENTRANCE  
JMP LIN3.4 ;FINISH MESSAGE  
;PRINT LINE 3D OF ERROR MESSAGE  
;'RMB A = XXXXXX RMWC = XXXXXX'  
LINE3D: BIT #SW05,JSWR ;SWITCH 5 SET ?  
BEQ IS ;BR IF IT IS  
DISPLY LINB3 ;'RMB A = '  
MOV $RMB A(R0),-(SP) ;BUFFER ADDR REG CONTENTS  
JSR PC,LIN3OCT ;CONVERT TO OCTAL AND TYPE IT  
DISPLY LINW3 ;'RMWC = '  
MOV $RMWC(R0),-(SP) ;WORD COUNT REGISTER CONTENTS  
JSR PC,LIN3OCT ;CONVERT TO OCTAL AND TYPE IT  
DISPLY $CRLF  
IS: RTS PC  
;PRINT LINE 3E OF ERROR MESSAGE  
;'START CYL = CCC START TRK = TT START SEC = SS'  
LINE3E: DISPLY LINS3 ;'START CYL = '  
MOV $CYL(R0),-(SP) ;MOVE CYL TO STACK  
JSR PC,LINDEC ;TYPE IT IN DECIMAL  
DISPLY LINSP ;SPACES  
DISPLY LINST3 ;'START TRK = '  
CLR -(SP) ;CLEAR STACK  
MOVB $TRK(R0),(SP) ;TRACK TO STACK  
JSR PC,LINDEC ;TYPE IT IN DECIMAL  
DISPLY LINSP ;SPACES  
DISPLY LINSS3 ;'START SEC = '  
CLR -(SP) ;CLEAR STACK  
MOVB $SEC(R0),(SP) ;SECTOR ADDR TO STACK  
JSR PC,LINDEC ;TYPE IT IN DECIMAL  
DISPLY $CRLF  
RTS PC  
;PRINT LINE 3F OF ERROR MESSAGE  
;'RMD A = XXXXXX RMCA = XXXXXX'
```

```

5612 021262 032777 000040 157664 LINE3F: BIT      #SW5,2SWR      ;SWITCH 5 SET ?
5613 021270 001420          BEQ      IS              ;BR IF NOT
5614 021272 104414 051632      DISPLY  LINDA3          ;RMDA =
5615 021276 016046 000242      MOV      $RMDA(RO),-(SP) ;PUT SECTOR/TRACK ADDRESS ON THE STACK
5616 021302 004737 022430      JSR      PC,LINOCIT ;TYPE IT
5617 021306 104414 053246      DISPLY  LINSP          ;SPACES
5618 021312 104414 051621      DISPLY  LINCA3          ;RMDC =
5619 021316 016046 000270      MOV      $RMDC(RO),-(SP) ;PUT DESIRED CYLINDER ADDRESS ON THE STACK
5620 021322 004737 022430      JSR      PC,LINOCIT ;TYPE IT
5621 021326 104414 001201      DISPLY  $CRLF
5622 021332 000207          IS:      RTS      PC
5623
5624          ;'CCC TT SS   PREV ADR = CCC TT SS'
5625
5626          ;LIN3.1:
5627 021334 004737 022502      MOV      $RMDC(RO),-(SP) ;PUT CYLINDER ADDR ON STACK
5628 021340 004737 022462      LIN3.1: JSR      PC,READDR ;DECREMENT TRACK AND SECTOR ADDRESS
5629 021344 104414 051466      JSR      PC,LINDEC ;TYPE IT IN DECIMAL
5630          DISPLY  T ;PRINT 'T'
5631          ; JSR      PC,READDR ;DECREMENT TRACK AND SECTOR ADDRESSES
5632 021350 004737 022462      JSR      PC,LINDEC ;TYPE TRACK IN DECIMAL
5633 021354 104414 051512      DISPLY  S ;PRINT 'S'
5634 021360 004737 022462      JSR      PC,LINDEC ;TYPE SECTOR ADDRESS
5635 021364 104414 051515      DISPLY  LINP3 ;PRINT 'PREV ADDR'
5636 021370 016046 000034      MOV      $PREVA+2(RO),-(SP) ;PREVIOUS CYLINDER
5637 021374 004737 022462      JSR      PC,LINDEC ;TYPE IT IN DECIMAL
5638 021400 104414 051466      DISPLY  T ;PRINT 'T'
5639 021404 005046          CLR      -(SP) ;MAKE ROOM ON THE STACK
5640 021406 116016 000033      MOV      $PREVA+1(RO),(SP) ;PREVIOUS TRACK ADDRESS
5641 021412 004737 022462      JSR      PC,LINDEC ;TYPE IT IN DECIMAL
5642 021416 104414 051512      DISPLY  S ;PRINT 'S'
5643 021422 005046          CLR      -(SP) ;MAKE ROOM ON THE STACK
5644 021424 116016 000032      MOV      $PREVA(RO),(SP) ;PREVIOUS SECTOR ADDRESS
5645 021430 004737 022462      JSR      PC,LINDEC ;TYPE IT IN DECIMAL
5646 021434 104414 001201      DISPLY  $CRLF
5647 021440 000207          RTS      PC
5648
5649          ;'START CYL = CCC   END CYL = CCC'
5650
5651 021442 104414 051536      LIN3.3: DISPLY  LIN3 ;LINE '3B & 3C' ENTRANCE
5652 021446 016046 000034      MOV      $PREVA+2(RO),-(SP) ;PREVIOUS CYLINDER
5653 021452 004737 022462      JSR      PC,LINDEC ;TYPE IT IN DECIMAL
5654 021456 104414 051553      DISPLY  LINEN3 ;PRINT 'END CYL'
5655 021462 016046 000270      MOV      $RMDC(RO),-(SP) ;PRESENT CYLINDER
5656 021466 004737 022462      JSR      PC,LINDEC ;TYPE IT IN DECIMAL
5657 021472 000207          RTS      PC
5658
5659          ;'ACTUAL CYL = CCC   TRK = TT'
5660
5661 021474 104414 051570      LIN3.4: DISPLY  LIN3 ;PRINT 'ACTUAL'
5662 021500 013746 056512      MOV      CYLDER, -(SP) ;ACTUAL CYLINDER
5663 021504 042716 010000      BIC      #BIT12,(SP) ;CLEAR THE FORMAT BIT
5664 021510 004737 022462      JSR      PC,LINDEC ;TYPE IT IN DECIMAL
5665 021514 104414 051610      DISPLY  LIN3 ;PRINT TRACK
5666 021520 005046          CLR      -(SP) ;CLEAR STACK WORD
5667 021522 116016 000243      MOV      $RMDA+1(RO),(SP) ;PUT TRACK ON STACK
5668 021526 004737 022462      JSR      PC,LINDEC ;TYPE IT IN DECIMAL

```

5668 021532 104414 001201
5669 021536 000207
5670
5671
5672
5673
5674 021540 032760 000100 000016
5675 021546 001427
5676 021550 104414 051716
5677 021554 016046 000006
5678 021560 004737 022430
5679 021564 104414 051735
5680 021570 016046 000020
5681 021574 004737 022462
5682 021600 104414 051747
5683 021604 016046 000240
5684 021610 166016 000006
5685 021614 006216
5686 021616 004737 022462
5687 021622 104414 001201
5688 021626 000207
5689
5690
5691
5692
5693 021630 104414 052002
5694 021634 162760 000002 000240
5695 021642 017046 000240
5696 021646 004737 022430
5697 021652 104414 052017
5698 021656 016046 000256
5699 021662 004737 022430
5700 021666 016046 000236
5701 021672 066016 000020
5702 021676 005046
5703 021700 016046 000022
5704 021704 004737 027522
5705 021710 012616
5706 021712 104414 052035
5707 021716 004737 022462
5708 021722 104414 001201
5709 021726 000207
5710
5711
5712
5713
5714 021730
5715 021730 104414 052053
5716 021734 013746 056512
5717 021740 004737 022430
5718 021744 104414 053246
5719 021750 013746 056514
5720 021754 004737 022430
5721 021760 104414 053246
5722 021764 104414 053251
5723 021770 104414 001201

DISPLY \$CRLF
RTS PC

;PRINT LINE 4 OF ERROR MESSAGE
; 'BUFFER ADDR = XXXXXX SIZE = XXXX ACTUAL NMBR WPDs XFRD = XXX'

LINE4: BIT #BIT06,STATUS(RO) ;DATA ERROR ?
BEQ 1\$;BR IF NOT
DISPLY LINM4 ;'PRINT BUFFER'
MOV \$BUF(RO),-(SP) ;BUFFER ADDR ON STACK
JSR PC,LINOC ;CONVERT TO OCTAL & PRINT
DISPLY LINS4 ;PRINT 'SIZE'
MOV \$WORDL(RO),-(SP) ;BUFFER SIZE
JSR PC,LINDEC ;TYPE IT IN DECIMAL
DISPLY LINX4 ;'ACTUAL NMBR WPDs XFRD = '
MOV \$RMB(RO),-(SP) ;VALUE IN BUFFER ADDR REGISTER
SUB \$BUF(RO),1(SP) ;SUBTRACT STARTING ADDRESS
ASR (SP) ;CONVERT INTO A WORD COUNT
JSR PC,LINDEC ;TYPE IT IN DECIMAL
DISPLY \$CRLF ;CR-LF
1\$: RTS PC ;RETURN

;PRINT LINE 5 OF ERROR MESSAGE
; 'GOOD DATA = XXXXXX BAD DATA = XXXXXX SECT POS = XXX'

LINES: DISPLY LINDS ;PRINT 'GOOD DATA'
SUB 12,\$RMB(RO) ;BACK THE ADDRESS UP
MOV 2,\$RMB(RO),-(SP) ;'GOOD' DATA - AT THE BUFFER LOCATION
JSR PC,LINOC ;TYPE IT
DISPLY LINBS ;PRINT 'BAD DATA'
MOV \$RMB(RO),-(SP) ;BAD DATA FROM BUFFER
JSR PC,LINOC ;TYPE IT
MOV \$RMC(RO),-(SP) ;WORD LENGTH ON STACK
ADD \$WORDL(RO),(SP) ;MAKE INTO A POSITIVE NUMBER
CLR -(SP) ;UPPER DIVIDEND TO ZERO
MOV \$SSEC(RO),-(SP) ;SECTOR SIZE ON THE STACK
JSR PC,LINKDV ;DIVIDE WORDS XFERED BY SECTOR SIZE
MOV (SP)+,(SP) ;MOVE REMAINDER UP THE STACK
DISPLY LINPS ;PRINT 'SECT POS'
JSR PC,LINDEC ;TYPE THE POSITION
DISPLY \$CRLF
RTS PC

;PRINT LINE 5A OF THE ERROR MESSAGE
; 'HEADER FROM ERROR SECTOR XXXXXX XXXXXX XXXXXX XXXXXX'

LINESA:
2\$: DISPLY LINS5 ;'HEADER CONTENTS OF ERROR SECTOR'
MOV CYLDER, -(SP) ;HEADER POSITION
JSR PC,LINOC ;TYPE IT
DISPLY LINS4 ;SPACES
MOV CYLDER+2, -(SP) ;HEADER POSITION +2
JSR PC,LINOC ;TYPE IT
DISPLY LINS4 ;SPACES
DISPLY LINX5 ;APPENDING INFO 1/23 77
3\$: DISPLY \$CRLF


```

5724 021774 000207          RTS      PC
5725
5726      ;PRINT LINE 5B OF ERROR MESSAGE
5727      ;'RMEC1 = XXXXXX  RMEC2 = XXXXXX'
5728
5729 021776 104414 052107      LINE5B: DISPLY  LINEP5      ;'RMEC1 = '
5730 022002 016046 000300      MOV      $RMEC1(R0),-(SP) ;PUT REGISTER CONTENTS ON THE STACK
5731 022006 004737 022430      JSR      PC,LINOC7 ;TYPE IT
5732 022012 104414 053246      DISPLY  ,LINSP      ;SPACES
5733 022016 104414 052120      DISPLY  ,LINE05      ;' RMEC2 = '
5734 022022 016046 000302      MOV      $RMEC2(R0),-(SP) ;PUT REGISTER CONTENTS ON THE STACK
5735 022026 004737 022430      JSR      PC,LINOC7 ;TYPE IT
5736 022032 104414 001201      DISPLY  $CRLF
5737 022036 000207          RTS      PC      ;RETURN
5738
5739      ;PRINT LINE 6 OF ERROR MESSAGE
5740      ;'SECTOR IS ECC CORRECTABLE'
5741
5742 022040 104414 052132      LINE6:  DISPLY  ,LINB6      ;ECC CORRECTABLE
5743 022044 104414 001201      DISPLY  $CRLF
5744 022050 000207          RTS      PC
5745
5746      ;PRINT LINE 6A OF THE ERROR MESSAGE
5747      ;'SECTOR READ CORRECTLY AT OFFSET N'
5748
5749 022052 104414 052165      LINE6A: DISPLY  ,LIN66      ;PRINT 'READ CORRECTLY AT OFFSET N'
5750 022056 000411          BR      LIN6.1      ;TYPE THE REST OF THE LINE
5751
5752      ;PRINT LINE 6B OF THE ERROR MESSAGE
5753      ;'SECTOR IS ECC CORRECTABLE AT OFFSET N'
5754
5755 022060 104414 052132      LINE6B: DISPLY  ,LINB6      ;PRINT 'SECTOR IS ECC CORRECTABLE '
5756 022064 000406          BR      LIN6.1
5757
5758      ;PRINT LINE 6C OF THE ERROR MESSAGE
5759      ;'CORRECTED ON NTH RETRY'
5760
5761 022066 104414 052214      LINE6C: DISPLY  ,LIN66      ;'CORRECTED ON NTH RETRY'
5762 022072 000414          BR      LIN6.2      ;TYPE THE REST OF THE LINE
5763
5764      ;PRINT LINE 6D OF THE ERROR MESSAGE
5765      ;'UNCORRECTABLE AFTER N RETRIES'
5766
5767 022074 104414 052243      LINE6D: DISPLY  ,LINU06      ;'UNCORRECTABLE AFTER N RETRIES'
5768 022100 000411          BR      LIN6.2      ;FINISH
5769
5770      ;TYPE THE OFFSET VALUE IN MICRO-INCHES
5771
5772 022102 006301          LIN6.1: ASL      R1      ;DOUBLE THE OFFSET TABLE INDEX
5773 022104 016137 002470 022114      MOV      OFMTBL(R1),1$ ;ADDRESS OF OFFSET POSITION MESSAGE
5774 022112 104414          DISPLY  ,WORD      ;
5775 022114 000000          1$:      .WORD      0      ;OFFSET VALUE
5776 022116 104414 001201      DISPLY  $CRLF
5777 022122 000207          RTS      PC
5778
5779      ;RETRY COUNT TYPEOUT

```

5780
5781 022124 005046
5782 022126 113716 001351
5783 022132 004737 022462
5784 022136 104414 052232
5785 022142 104414 001201
5786 022146 000207
5787
5788
5789
5790

LINE6.2: CLR -(SP) ;CLEAR STACK
MOV RETRY+1,(SP) ;RETRY COUNT
JSR PC,LINDEC ;TYPE IT IN DECIMAL
DISPLY ,LINR6 ;'RETRY'
DISPLY ,SCRLF
RTS PC

;PRINT LINE 7 OF THE ERROR MESSAGE
;'ORDERS:XXXXX TOTAL ERRORS:XXX WRDS XFRD:XXXXXXX WRDS READ:XXXXXXX'

5791 022150 104414 052316
5792 022154 012746 000036
5793 022160 060016
5794 022162 004737 034050
5795 022166 004737 030036
5796 022172 104414 052370
5797 022176 016046 000056
5798 022202 004737 022462
5799 022206 104414 052402
5800 022212 012746 000046
5801 022216 060016
5802 022220 004737 034050
5803 022224 004737 030036
5804 022230 104414 052416
5805 022234 012746 000052
5806 022240 060016
5807 022242 004737 034050
5808 022246 004737 030036
5809 022252 104414 001201
5810 022256 104414 001201
5811 022262 032777 100000 156664
5812 022270 001401
5813 022272 000000
5814 022274 000207
5815
5816
5817
5818

LINE7: DISPLY ,LIN70 ;PRINT ORDER COUNT
MOV ,SOPERC,-(SP) ;TO STACK
ADD RO,(SP) ;ADD THE BASE ADDRESS
JSR PC,\$OB2D ;CONVERT IT
JSR PC,\$SUPRS ;PRINT IT
DISPLY ,LIN7T ;TOTAL ERRORS
MOV ,TOTAL(RO),-(SP) ;TO STACK
JSR PC,LINDEC ;TYPE IT IN DECIMAL
DISPLY ,LIN7X ;PRINT 'WRDS XFR'
MOV ,STRANS,-(SP) ;ADDRESS OF LOW WORD ON STACK
ADD RO,(SP)
JSR PC,\$OB2D ;CONVERT
JSR PC,\$SUPRS ;PRINT
DISPLY ,LIN7R ;'BITS READ'
MOV ,SREAD,-(SP) ;LOW WORD ADDRESS
ADD RO,(SP)
JSR PC,\$OB2D ;CONVERT
JSR PC,\$SUPRS ;PRINT
DISPLY ,SCRLF
DISPLY ,SCRLF
BIT ,SW15,\$SWR ;SEE IF 'HALT ON ERROR' - SWITCH 15
BEQ 15 ;BR IF NOT
HALT ;SWITCH 15 HALT
15: RTS PC

;PRINT LINE 7A OF ERROR MESSAGE
;'ORDERS:XXXXX TOTAL SEEKS=XXXXX TOTAL MISPOS ERR = XXX TOTAL SKI= XXX'

5819 022276 104414 052316
5820 022302 012746 000036
5821 022306 060016
5822 022310 004737 034050
5823 022314 004737 030036
5824 022320 104414 052326
5825 022324 012746 000042
5826 022330 060016
5827 022332 004737 034050
5828 022336 004737 030036
5829 022342 104414 052270
5830 022346 016046 000066
5831 022352 004737 022462
5832 022356 104414 052346
5833 022362 016046 000064
5834 022366 004737 022462
5835 022372 104414 001201

LINE7A: DISPLY ,LIN70 ;'ORDERS = '
MOV ,SOPERC,-(SP) ;ORDER COUNT INCREMENT
ADD RO,(SP) ;ADD BASE ADDRESS
JSR PC,\$OB2D ;CONVERT THE COUNT
JSR PC,\$SUPRS ;PRINT IT
DISPLY ,LIN7P ;'TOTAL SEEKS = '
MOV ,SPOSIT,-(SP) ;TOTAL SEEKS
ADD RO,(SP) ;DEVICE TABLE ADDRESS
JSR PC,\$OB2D ;CONVERT THE SEEK COUNT
JSR PC,\$SUPRS ;PRINT IT
DISPLY ,LIN7M ;'TOTAL MISPOS ERR = '
MOV ,MISPO(RO),-(SP) ;TOTAL ERRORS
JSR PC,LINDEC ;TYPE IT IN DECIMAL
DISPLY ,LIN7S ;'TOTAL SKI,OCYL ERR = '
MOV ,SKI(RO),-(SP) ;CONVERT & PRINT IT
JSR PC,LINDEC ;TYPE IT IN DECIMAL
DISPLY ,SCRLF


```

5836 022376 104414 001201      DISPLY  , $CRLF
5837 022402 032777 100000 156544  BIT      $SW15, $SWR      ;SEE IF HALT ON ERROR - SWITCH 15 SET
5838 022410 001401              BEQ      1$      ;BR IF NOT
5839 022412 000000              HALT              ;SWITCH 15 HALT
5840 022414 000207      1$:      RTS      PC
5841
5842      ;PRINT LINE 8 OF THE ERROR MESSAGE
5843      ;'DIFFERENT ERROR DURING RETRY'
5844
5845 022416 104414 052433      LINE8:  DISPLY  , LIN8M
5846 022422 004737 020462      JSR      PC, LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
5847 022426 000207      RTS      PC
5848
5849      ;OCTAL TYPEOUT ROUTINE
5850      ;CALL:
5851      ;      MOV      NUM, -(SP)      ;PUT THE NUMBER ON THE STACK
5852      ;      JSR      PC, LINOCT
5853      ;      RETURN
5854
5855 022430 016646 000002      LINOCT:  MOV      2(SP), -(SP)      ;PUT NUMBER IN PROPER LOCATION ON STACK
5856 022434 004737 030466      JSR      PC, $S820      ;CONVERT THE NUMBER TO OCTAL
5857 022440 012637 022454      MOV      (SP)+, 1$      ;GET THE ADDRESS OF THE ASCII STRING
5858 022444 062737 000005 022454  ADD      $5., 1$      ;ADDRESS THE LAST 6 ASCII DIGITS
5859 022452 104414              DISPLY              ;TYPE IT
5860 022454 000000      1$:      .WORD      0      ;ADDRESS
5861 022456 012616      MOV      (SP)+, (SP)      ;CORRECT THE STACK
5862 022460 000207      RTS      PC      ;RETURN
5863
5864      ;ROUTINE TO CONVERT THE INPUT NUMBER TO DECIMAL AND TYPE IT WITH
5865      ;LEADING ZERO SUPPRESSION
5866      ;CALL:
5867      ;      MOV      NUM, -(SP)      ;PUT THE NUMBER ON THE STACK
5868      ;      JSR      PC, LINDEC
5869      ;      RETURN
5870
5871 022462 016646 000002      LINDEC:  MOV      2(SP), -(SP)      ;SET UP STACK FOR CONVERT
5872 022466 004737 030436      JSR      PC, $S820      ;CONVERT IT TO DECIMAL
5873 022472 004737 030036      JSR      PC, $SUPRS      ;TYPE IT (WITH LEADING ZEROS SUPRESSED)
5874 022476 012616      MOV      (SP)+, (SP)      ;RESTORE STACK POINTER
5875 022500 000207      RTS      PC
5876
5877      ;*****
5878
5879      .SBTTL  GENERAL SUPPORT SUBROUTINES
5880
5881      ;*****
5882
5883      ;DECREMENT THE SECTOR-TRACK ADDRESS
5884      ;CALL:
5885      ;      MOV      $DPB, R0      ;DPB ADDRESS
5886      ;      JSR      PC, READDR
5887      ;      RETURN
5888      ;      (SP) CONTAINS THE CYLINDER ADDRESS
5889      ;      2(SP) CONTAINS THE TRACK ADDRESS
5890      ;      4(SP) CONTAINS THE SECTOR ADDRESS
5891

```

```

5892 022502 162706 000006 READR: SUB #6,SP ;DECREMENT THE STACK POINTER
5893 022506 016616 000006 MOV 6(SP), (SP) ;MOVE THE RETURN ADDR DOWN THE STACK
5894 022512 005066 000006 CLR 6(SP) ;CLEAR STACK FOR SECTOR
5895 022516 005066 000004 CLR 4(SP) ;CLEAR STACK FOR TRACK
5896 022522 005066 000002 CLR 2(SP) ;CLEAR STACK FOR CYLINDER
5897 022526 116066 000242 000006 MOVB $RMDA(R0), 6(SP) ;SECTOR ON STACK
5898 022534 116066 000243 000004 MOVB $RMDA+1(R0), 4(SP) ;TRACK ADDRESS
5899 022542 016066 000270 000002 MOV $RMDC(R0), 2(SP) ;CYLINDER ADDRESS
5900 022550 005766 000006 TST 6(SP) ;SECTOR 0 ?
5901 022554 001403 BEQ 1$ ;BRANCH IF 50
5902 022556 105366 000006 DECB 6(SP) ;DECREMENT ONE SECTOR
5903 022562 000421 BR 3$ ;BRANCH TO EXIT
5904 022564 005766 000004 1$: TST 4(SP) ;ALSO ON TRACK 0 ?
5905 022570 001406 BEQ 2$ ;BRANCH IF 50
5906 022572 112766 000037 000006 MOVB #31, 6(SP) ;LAST SECTOR
5907 022600 105366 000004 DECB 4(SP) ;DECREMENT ONE TRACK
5908 022604 000410 BR 3$ ;EXIT
5909 022606 112766 000037 000006 2$: MOVB #31, 6(SP) ;LAST SECTOR
5910 022614 112766 000004 000004 MOVB #4, 4(SP) ;LAST TRACK
5911 022622 005366 000002 DEC 2(SP) ;DECREMENT ONE CYLINDER COUNT
5912 022626 000240 3$: NOP ;DONE
5913 022630 000207 RTS PC ;RETURN
5914
5915 ;ROUTINE TO CHECK FOR KW11-L OR KW11-P CLOCKS
5916
5917 022632 012737 177777 001310 CKCLK: MOV #1, CLKFLG ;CLEAR CLOCK AVAILABILITY FLAG
5918 022640 012737 177777 001306 MOV #1, PCLOCK ;CLEAR KW11-P CLOCK AVAILABILITY FLAG
5919 022646 012737 022726 000004 MOV #CKCLK1, ERRVEC ;SET UP VECTOR FOR CLOCK CHECK
5920 022654 005037 000006 CLR #ERRVEC+2 ;NEW PSW
5921 022660 005777 156410 TST #CLKCSR ;CHECK FOR KW11-P
5922 022664 005037 001310 CLR CLKFLG ;SET CLOCK AVAILABILITY FLAG
5923 022670 005037 001306 CLR PCLOCK ;SET KW11-P CLOCK FLAG
5924 022674 013701 001300 MOV $LPVEC, R1 ;KW11-P VECTOR ADDRESS
5925 022700 012721 023766 MOV #CLOCK, (R1)+ ;SET UP KW11-P VECTOR
5926 022704 012711 000300 MOV #300, (R1) ;PSW - PRI 6
5927 022710 012777 174575 156360 MOV #-1667, #CLKCSB ;LOAD COUNTER BUFFER WITH 16.67
5928 022716 012777 000131 156350 MOV #131, #CLKCSR ;SET CLOCK - CNT UP, 10US, CONT INT
5929 022724 000437 BR CKCLK3
5930 022726 062706 000004 CKCLK1: ADD #4, SP ;RESTORE THE STACK POINTER
5931 022732 012737 022774 000004 MOV #CKCLK2, #ERRVEC ;CHANGE ERROR VECTOR TO CHECK FOR KW11-L
5932 022740 005777 156336 TST #CLKS ;LOOK FOR KW11-L
5933 022744 005037 001310 CLR CLKFLG ;SET CLOCK FLAG
5934 022750 013701 001304 MOV $LLVEC, R1 ;KW11-L VECTOR ADDRESS
5935 022754 012721 023766 MOV #CLOCK, (R1)+ ;SET UP KW11-L VECTOR
5936 022760 012711 000300 MOV #300, (R1) ;PSW - PRI 6
5937 022764 012777 000100 156310 MOV #100, #CLKS ;SET KW11-L INTERRUPT
5938 022772 000414 BR CKCLK3
5939 022774 062706 000004 CKCLK2: ADD #4, SP ;RESTORE THE STACK POINTER
5940 023000 104401 054054 TYPE #, #EDCLK ;"P OR L CLOCK MUST BE ON SYSTEM"
5941 023004 005737 000042 TST 42 ;UNDER MONITOR CONTROL ?
5942 023010 001402 BEQ 1$ ;BR IF NOT
5943 023012 000137 027440 JMP $GET42 ;ABORT PROGRAM
5944 023016 000000 1$: HALT ;HALT
5945 023020 000137 003622 JMP START ;TRY AGAIN
5946 023024 012737 000006 000004 CKCLK3: MOV #6, #ERRVEC ;RESTORE THE ERROR VECTOR
5947 023032 000207 RTS PC

```

```

5948
5949
5950      ;ROUTINE TO DISPLAY STATISTICS FOR ALL DRIVES ASSIGNED
5951      ;CALL:
5952      JSR      PC,STATPR
5953      RETURN
5954
5954      023034 010046      STATPR: MOV      R0,-(SP)      ;SAVE R0
5955      023036 010446      MOV      R4,-(SP)      ;SAVE R4
5956      023040 005737 001544      TST      ASNLST      ;ANY DRIVES ASSIGNED ?
5957      023044 001423      BEQ      3$      ;BR IF NOT
5958      023046 004737 023150      JSR      PC,SHDTYP      ;TYPE THE HEADING
5959      023052 005004      CLR      R4      ;CLEAR THE DRIVE INDEX
5960      023054 006304      1$: ASL      R4      ;CHANGE TO INDEX WORDS
5961      023056 016400 002102      MOV      BLKADR(R4),R0 ;GET THE DRIVE'S BLOCK ADDRESS
5962      023062 006204      ASR      R4      ;RESTORE R4
5963      023064 136437 034606 001544      BITB     ATABIT(R4),ASNLST ;IS THIS DRIVE ASSIGNED ?
5964      023072 001404      BEQ      2$      ;BR IF NOT
5965      023074 004737 023172      JSR      PC,SDOTAL      ;TYPE THE PERFORMANCE SUMMARY
5966      023100 104401 001201      TYPE     $CRLF      ;CR-LF
5967      023104 005204      2$: INC      R4      ;INCREMENT THE INDEX
5968      023106 020427 000010      CMP      R4,#8      ;FINISHED ?
5969      023112 001360      BNE      1$      ;BR IF NOT
5970      023114 012604      3$: MOV      (SP)+,R4      ;RESTORE R4
5971      023116 012600      MOV      (SP)+,R0      ;RESTORE R0
5972      023120 000207      RTS      PC      ;RETURN
5973
5974      ;ROUTINE TO TYPE STATISTICS FOR AN INDIVIDUAL DRIVE
5975      ;CALL:
5976      MOV      #DPB,R0      ;DPB ADDRESS
5977      JSR      PC,TYPEST
5978      RETURN
5979
5980      023122 010046      TYPEST: MOV      R0,-(SP)      ;SAVE R0
5981      023124 010446      MOV      R4,-(SP)      ;SAVE R4
5982      023126 004737 023150      JSR      PC,SHDTYP      ;TYPE THE HEADING
5983      023132 005004      CLR      R4      ;CLEAR R4 FOR DRIVE NUMBER
5984      023134 111004      MOV      (R0),R4      ;DRIVE NUMBER
5985      023136 004737 023172      JSR      PC,SDOTAL      ;TYPE THE STATISTICS
5986      023142 012604      MOV      (SP)+,R4      ;RESTORE R4
5987      023144 012600      MOV      (SP)+,R0      ;RESTORE R0
5988      023146 000207      RTS      PC      ;RETURN
5989
5990      ;TYPE THE HEADER FOR THE DRIVE PERFORMANCE SUMMARY TYPEOUT
5991      ;CALL:
5992      JSR      PC,SHDTYP
5993      RETURN
5994
5995      023150 004737 023670      SHDTYP: JSR      PC,$TIME      ;TYPE THE TIME OF DAY
5996      023154 004537 030076      JSR      R5,TYPRI4      ;TYPE AT PRIORITY 4
5997      023160 001201      $CRLF      ;CR-LF
5998      023162 004537 030076      JSR      R5,TYPRI4      ;TYPE THE HEADER
5999      023166 053524      STATHD      ;HEADER
6000      023170 000207      RTS      PC      ;RETURN
6001
6002      ;TYPE THE PERFORMANCE SUMMARY DATE LINE
6003      ;CALL:

```

6004				:	MOV	#DRIVE,R4	:DRIVE NUMBER
6005				:	MOV	#DPB,R0	:DPB ADDRESS
6006				:	RETURN		
6007				:			
6008	023172	010246		SDETAL:	MOV	R2,-(SP)	:SAVE R2
6009	023174	010002			MOV	R0,R2	:DPB ADDRESS
6010	023176	062702	000036		ADD	#\$OPERC,R2	:FIRST STATISTICAL FIELD
6011	023202	010446			MOV	R4,-(SP)	:SAVE R4 FOR TYPEOUT
6012							:TYPE DRIVE NUMBER
6013	023204	104403			TYPE		:GO TYPE--OCTAL ASCII
6014	023206	002			.BYTE	2	:TYPE 2 DIGIT(S)
6015	023207	000			.BYTE	0	:SUPPRESS LEADING ZEROS
6016	023210	104401	053246		TYPE	LINSP	:SPACES
6017	023214	016046	000070		MOV	\$PASSC(R0),-(SP)	:PUT THE PASS COUNT ON THE STACK
6018	023220	004737	030436		JSR	PC,\$SB20	:CONVERT IT
6019	023224	004537	027746		JSR	R5,REPLZ	:TYPE IT
6020	023230	000003			.WORD	3	:TYPE 3 DIGITS
6021	023232	104401	053246		TYPE	LINSP	:SPACES
6022	023236	010246			MOV	R2,-(SP)	:PUT \$OPERC ON THE STACK
6023	023240	004737	034050		JSR	PC,\$DB20	:CONVERT IT
6024	023244	004537	027746		JSR	R5,REPLZ	:TYPE \$OPERC
6025	023250	000006			.WORD	6	:TYPE 6 DIGITS
6026	023252	104401	053246		TYPE	LINSP	:SPACES
6027	023256	062702	000004		ADD	#4,R2	:INCREMENT R2
6028	023262	010246			MOV	R2,-(SP)	:PUT \$POSIT ON THE STACK
6029	023264	004737	034050		JSR	PC,\$DB20	:CONVERT IT
6030	023270	004537	027746		JSR	R5,REPLZ	:TYPE \$POSIT
6031	023274	000006			.WORD	6	:TYPE 6 DIGITS
6032	023276	104401	053246		TYPE	LINSP	:SPACES
6033	023302	062702	000004		ADD	#4,R2	:INCREMENT R2
6034	023306	010246			MOV	R2,-(SP)	:PUT \$TRANS ON THE STACK
6035	023310	004737	034050		JSR	PC,\$DB20	:CONVERT \$TRANS
6036	023314	004537	027746		JSR	R5,REPLZ	:TYPE IT
6037	023320	000012			.WORD	10	:TYPE 10 DIGITS
6038	023322	104401	053246		TYPE	LINSP	:SPACES
6039	023326	062702	000004		ADD	#4,R2	:INCREMENT R2
6040	023332	010246			MOV	R2,-(SP)	:PUT \$READ ON THE STACK
6041	023334	004737	034050		JSR	PC,\$DB20	:CONVERT \$READ
6042	023340	004537	027746		JSR	R5,REPLZ	:TYPE IT
6043	023344	000012			.WORD	10	:TYPE 10 DIGITS
6044	023346	104401	053247		TYPE	LINSP0	:1 SPACE
6045	023352	062702	000004		ADD	#4,R2	:INCREMENT R2
6046	023356	062702	000002		ADD	#2,R2	:INCREMENT R2 AGAIN
6047	023362	012246			MOV	(R2)+,-(SP)	:PUT \$SOFT ON THE STACK
6048	023364	004737	030436		JSR	PC,\$SB20	:CONVERT \$SOFT
6049	023370	004537	027746		JSR	R5,REPLZ	:TYPEOUT \$SOFT
6050	023374	000004			.WORD	4	:TYPE 4 DIGITS
6051	023376	104401	053247		TYPE	LINSP0	:SPACES
6052	023402	012246			MOV	(R2)+,-(SP)	:PUT \$HARD ON THE STACK
6053	023404	004737	030436		JSR	PC,\$SB20	:CONVERT \$HARD
6054	023410	004537	027746		JSR	R5,REPLZ	:TYPEOUT \$HARD
6055	023414	000004			.WORD	4	:TYPE 4 DIGITS
6056	023416	104401	053247		TYPE	LINSP0	:SPACES
6057	023422	012246			MOV	(R2)+,-(SP)	:PUT \$SKI ON THE STACK
6058	023424	004737	030436		JSR	PC,\$SB20	:CONVERT \$SKI
6059	023430	004537	027746		JSR	R5,REPLZ	:TYPEOUT \$SKI

```

6060 023434 000004          .WORD 4          ;TYPE 4 DIGITS
6061 023436 104401 053247  .TYPE 4      ;SPACES
6062 023442 012246          MOV 4(R2)+,-(SP) ;PUT $MISPO ON THE STACK
6063 023444 004737 030436  .JSR PC,$S820 ;CONVERT $MISPO
6064 023450 004537 027746  .JSR RS,REPLZ ;TYPEOUT $MISPO
6065 023454 000004          .WORD 4          ;TYPE 4 DIGITS
6066 023456 104401 053247  .TYPE 4      ;SPACES
6067 023462 016046 000056  .MOV $TOTAL(R0),-(SP) ;CALCULATE NUMBER OF OTHER ERRORS
6068 023466 166016 000060  .SUB $SOFT(R0),{SP} ;SUBTRACT $SOFT FROM $TOTAL
6069 023472 166016 000062  .SUB $SHARD(R0),{SP} ;SUBTRACT $SHARD FROM $TOTAL
6070 023476 166016 000064  .SUB $SKI(R0),{SP} ;SUBTRACT $SKI FROM $TOTAL
6071 023502 166016 000066  .SUB $MISPO(R0),{SP} ;SUBTRACT $MISPO FROM $TOTAL
6072 023506 004737 030436  .JSR PC,$S820 ;CONVERT 'OTHER' COUNT
6073 023512 004537 027746  .JSR RS,REPLZ ;TYPE IT
6074 023516 000004          .WORD 4          ;TYPE 4 DIGITS
6075 023520 005726          .TST (SP)+      ;CLEAR STACK
6076 023522 000207          .RTS PC
6077
6078
6079
6080          ;ROUTINE TO INCREMENT $SOFT
6081          ;
6082          ;NOTE: $SOFT WILL NOT BE INCREMENTED BEYOND 9999 (10)
6083
6084 023524 005737 001362  INCSOF: .TST BADSEC ;SEE IF BAD TRK/SEC INDICATOR SET
6085 023530 001006          .BNE 1$ ;BR IF IT'S SET, DON'T INCREMENT COUNT
6086 023532 026027 000060 023417 .CMP $SOFT(R0),#9999. ;IS $SOFT ALREADY AT MAXIMUM ?
6087 023540 103002          .BHS 1$ ;BR IF IT IS
6088 023542 005260 000060  .INC $SOFT(R0) ;INCREMENT $SOFT
6089 023546 000207          1$: .RTS PC ;RETURN
6090
6091
6092          ;ROUTINE TO INCREMENT $SHARD
6093          ;
6094          ;NOTE: $SHARD WILL NOT BE INCREMENTED BEYOND 9999 (10)
6095
6096 023550 005737 001362  INCHRD: .TST BADSEC ;SEE IF BAD TRK/SEC INDICATOR SET
6097 023554 001006          .BNE 1$ ;BR IF IT'S SET, DON'T INCREMENT COUNT
6098 023556 026027 000062 023417 .CMP $SHARD(R0),#9999. ;IS $SHARD ALREADY AT MAXIMUM ?
6099 023564 103002          .BHS 1$ ;BR IF IT IS
6100 023566 005260 000062  .INC $SHARD(R0) ;INCREMENT $SHARD
6101 023572 000207          1$: .RTS PC ;RETURN
6102
6103
6104          ;ROUTINE TO INCREMENT $SKI
6105          ;
6106          ;NOTE: $SKI WILL NOT BE INCREMENTED BEYOND 9999 (10)
6107
6108 023574 005737 001362  INCSKI: .TST BADSEC ;SEE IF BAD TRK/SEC INDICATOR SET
6109 023600 001006          .BNE 1$ ;BR IF IT'S SET, DON'T INCREMENT COUNT
6110 023602 026027 000064 023417 .CMP $SKI(R0),#9999. ;IS $SKI ALREADY AT MAXIMUM ?
6111 023610 103002          .BHS 1$ ;BR IF IT IS
6112 023612 005260 000064  .INC $SKI(R0) ;INCREMENT $SKI
6113 023616 000207          1$: .RTS PC ;RETURN
6114
6115

```

```

6116 ;ROUTINE TO INCREMENT $MISPO
6117 ;
6118 ;NOTE: $MISPO WILL NOT BE INCREMENTED BEYOND 9999 (10)
6119
6120 023620 005737 001362 INCMIS: TST BADSEC ;SEE IF BAD TRK/SEC INDICATOR SET
6121 023624 001006 BNE 1$ ;BR IF IT'S SET, DON'T INCREMENT COUNT
6122 023626 026027 000066 023417 CMP $MISPO(R0),#9999 ;IS $MISPO ALREADY AT MAXIMUM ?
6123 023634 103002 BHIS 1$ ;BR IF IT IS
6124 023636 005260 000066 INC $MISPO(R0) ;INCREMENT $MISPO
6125 023642 000207 1$: RTS PC ;RETURN
6126
6127 ;ROUTINE TO INCREMENT $TOTAL
6128 ;
6129 ;NOTE: $TOTAL WILL NOT BE INCREMENTED BEYOND 9999 (10)
6130
6131 023644 005737 001362 INCTOT: TST BADSEC ;SEE IF BAD TRK/SEC INDICATOR SET
6132 023650 001006 BNE 1$ ;BR IF IT'S SET, DON'T INCREMENT COUNT
6133 023652 026027 000056 023417 CMP $TOTAL(R0),#9999 ;IS $TOTAL ALREADY AT MAXIMUM ?
6134 023660 103002 BHIS 1$ ;BR IF IT IS
6135 023662 005260 000056 INC $TOTAL(R0) ;INCREMENT $TOTAL
6136 023666 000207 1$: RTS PC ;RETURN
6137
6138 ;ROUTINE TO TYPE THE TIME
6139
6140 $TIME: TST CLKFLG ;CLOCK ON THE SYSTEM ?
6141 BNE 1$ ;BR IF NOT
6142 023670 005737 001310 TYPE $CRLF ;CR-LF
6143 023674 001033 MOV HOUR,-(SP) ;PUT 'HOURS' ON THE STACK
6144 023676 104401 001201 JSR PC,$S820 ;CONVERT TO DECIMAL
6145 023702 013746 001364 JSR RS,REPLZ ;TYPE IT
6146 023706 004737 030436 .WORD 2 ;TYPE 2 DIGITS
6147 023712 004537 027746 TYPE COLON ;'.'
6148 023716 000002 MOV MINUTE,-(SP) ;PUT 'MINUTES' ON THE STACK
6149 023720 104401 054330 JSR PC,$S820 ;CONVERT TO DECIMAL
6150 023724 013746 001366 JSR RS,REPLZ ;TYPE IT
6151 023730 004737 030436 .WORD 2 ;TYPE 2 DIGITS
6152 023734 004537 027746 TYPE COLON ;'.'
6153 023740 000002 MOV SECOND,-(SP) ;PUT SECONDS ON THE STACK
6154 023742 104401 054330 JSR PC,$S820 ;CONVERT TO DECIMAL
6155 023746 013746 001370 JSR RS,REPLZ ;TYPE IT
6156 023752 004737 030436 .WORD 2 ;TYPE 2 DIGITS
6157 023756 004537 027746 1$: RTS PC
6158 023762 000002
6159 023764 000207
6160
6161 ;CLOCK HANDLER ROUTINE
6162
6163 023766 005337 001372 CLOCK: DEC SIXTEE ;INCREMENT THE 1/60 SECOND COUNTER
6164 023772 001035 BNE 1$ ;BR IF A SECOND NOT COUNTED
6165 023774 013737 001312 001372 MOV HZ,SIXTEE ;RESTORE THE VALUE
6166 024002 005237 001370 INC SECOND ;COUNT THE SECOND
6167 024006 022737 000074 001370 CMP #60.,SECOND ;AT MAXIMUM ?
6168 024014 001024 BNE 1$ ;BR IF NOT
6169 024016 005037 001370 CLR SECOND ;CLEAR THE SECOND'S COUNTER
6170 024022 005237 001470 INC INTRVL+2 ;COUNT THE PERFORMANCE SUMMARY INTERVAL
6171 024026 005237 001366 INC MINUTE ;COUNT THE MINUTE

```


6172	024032	022737	000074	001366	CMP	#60.,MINUTE	:AT MAXIMUM ?
6173	024040	001012			BNE	1\$	:BR IF NOT
6174	024042	005037	001366		CLR	MINUTE	:CLEAR THE MINUTE'S COUNTER
6175	024046	005237	001364		INC	HOUR	:COUNT THE HOURS
6176	024052	022737	001747	001364	CMP	#999.,HOUR	:AT MAXIMUM
6177	024060	103002			BHIS	1\$	:BR IF NOT
6178	024062	005037	001364		CLR	HOUR	:CLEAR THE HOURS
6179	024066	012746	000020		MOV	#20,-(SP)	:1 MS ON THE STACK
6180	024072	004737	041122		JSR	PC,APTR	:DRIVER TIMER ROUTINE
6181	024076	005737	001466		TST	INTRVL	:DISPLAY THE PERFORMANCE SUMMARY ?
6182	024102	001411			BEQ	2\$	:BR IF NOT
6183	024104	023737	001466	001470	CMP	INTRVL,INTRVL+2	:DISPLAY INTERVAL FINISHED ?
6184	024112	001005			BNE	2\$	:BR IF NOT
6185	024114	012737	177777	001314	MOV	#-1,STATIN	:SET PERFORMANCE SUMMARY DISPLAY FLAG
6186	024122	005037	001470		CLR	INTRVL+2	:CLEAR THE PERFORMANCE INTERVAL COUNTER
6187	024126	000002			RTI		
6188							
6189							
6190							
6191							
6192							
6193							
6194							
6195							
6196							
6197	024130	005737	001540		KSR:	TST ORDERQ+16	:ANY OPERATIONS ACTIVE ?
6198	024134	001402			BEQ	KSR1	
6199	024136	104401	054441		TYPE	,BUSY	
6200	024142	104412					
6201	024144	012737	000200	177776	KSR1:	SAVREG	:SAVE THE REGISTERS
6202	024152	005037	001360		MOV	#PR4,PS	:SET PRIORITY TO 4
6203	024156	004737	023670		CLR	CFLAG	:CLEAR THE 'CONTROL C' FLAG
6204	024162	005777	154774		JSR	PC,STIME	:TYPE THE TIME
6205	024166	104401	054154		TST	#1KB	:CLEAR ANY GARBAGE IN THE TTY BUFFER
6206	024172	104411			TYPE	,ENTCOM	:ENTER COMMANDS
6207	024174	012605			RDIN		:READ THE KEYBOARD
6208	024176	005737	001360		MOV	(SP)+,R5	:GET ADDRESS OF INPUT STRING
6209	024202	011116			TST	CFLAG	:CHECK THE CONTROL C FLAG
6210	024204	005205			BNE	7\$	:EXIT IF 'CONTROL C' ENTERED
6211	024206	122715	000124		INC	R5	:POINT TO SECOND CHARACTER
6212	024212	001462			CMPB	#'T,(R5)	:EQ TO A 'T' ?
6213	024214	122715	000101		BEQ	8\$	:YES
6214	024220	001410			CMPB	#'A,(R5)	:EQ TO AN 'A'
6215	024222	121527	000067		BEQ	1\$	:BR IF IT IS
6216	024226	101102			CMPB	(R5),#'7	:DRIVE NUMBER GREATER THAN AN ASCII 7 ?
6217	024230	121527	000060		BHI	6\$	:BR IF IT IS
6218	024234	103477			CMPB	(R5),#'0	:DRIVE NUMBER LESS THAN AN ASCII 0 ?
6219	024236	142715	177770		BLO	6\$	:BR IF IT IS
6220	024242	122765	000124	177777	BICB	#1C7,(R5)	:LEAVE ONLY LOWER 3 BITS IF CHAR NOT 'A'
6221	024250	001003			CMPB	#'T,-1(R5)	:EQ TO 'T'
6222	024252	004737	025016		BNE	2\$	:BR IF NOT EQ
6223	024256	000470			JSR	PC,NEWASH	:ASSIGN DRIVE FOR TEST
6224	024260	122765	000104	177777	BR	7\$	:EXIT
6225	024266	001003			CMPB	#'D,-1(R5)	:EQ TO 'D' ?
6226	024270	004737	025026		BNE	3\$	:BR IF NOT EQ
6227	024274	000461			JSR	PC,DEASGN	:DEASSIGN DRIVE
					BR	7\$	:EXIT

```

6228 024276 122765 000123 177777 3$: CMPB #'S,-1(R5) ;EQ TO 'S'
6229 024304 001003 BNE 4$ ;BR IF NOT EQ
6230 024306 004737 025136 JSR PC,SCMND ;TYPE STATISTICS
6231 024312 000452 BR 7$ ;EXIT
6232 024314 122765 000127 177777 4$: CMPB #'W,-1(R5) ;EQ TO 'W'
6233 024322 001007 BNE 5$ ;BR IF NOT EQ
6234 024324 032777 000001 154622 BIT #SW0,2SWR ;IS SWITCH 0 SET ?
6235 024332 001040 BNE 6$ ;BR IF SET, CAN'T DO 'W' COMMAND
6236 024334 004737 025370 JSR PC,DATA PK ;WRITE A DATA PACK
6237 024340 000437 BR 7$ ;EXIT
6238 024342 122765 000122 177777 5$: CMPB #'R,-1(R5) ;EQ TO 'R' ?
6239 024350 001031 BNE 6$ ;BR IF NOT EQ
6240 024352 004737 025420 JSR PC,REDAPK ;READ A DATA PACK
6241 024356 000430 BR 7$ ;EXIT
6242 024360 122765 000127 177777 8$: CMPB #'W,-1(R5) ;WT COMMAND ?
6243 024366 001022 BNE 6$ ;NO
6244 024370 122765 000101 000001 CMPB #'A,1(R5) ;ALL DRIVES ?
6245 024376 001413 BEQ 9$ ;YES
6246 024400 126527 000001 000067 CMPB 1(R5),#'7 ;GREAT THAN ?
6247 024406 101012 BHI 6$ ;YES
6248 024410 126527 000001 000000 CMPB 1(R5),#0 ;LESS THAN 0
6249 024416 103406 BLO 6$ ;YES
6250 024420 142765 177770 000001 BICB #1C7,1(R5) ;CHOP OFF THE HIGHER BITS
6251 024426 004737 025402 9$: JSR PC,WATPAK ;ASSIGN DRIVES WITH WT COMMAND
6252 024432 000402 BR 7$ ;EXIT
6253 024434 104401 054132 6$: TYPE ,INVLD ;TYPE 'INVALID COMMAND' MESSAGE
6254 024440 104413 7$: RESREG ;RESTORE R0 - R5
6255 024442 005777 154514 TST 2$TKB ;CLEAR THE TTY BUFFER
6256 024446 052777 000100 154504 BIS #BIT06,2$TKS ;SET TTY INTERRUPT ENABLE
6257 024454 005037 177776 CLR PS ;SET PRIORITY BACK TO ZERO
6258 024460 000207 RTS PC ;RETURN
6259
6260 ;ROUTINE TO PROCESS THE ASSIGN REQUEST ('T', 'R', OR 'W' COMMANDS)
6261
6262 024462 122715 000101 ASSIGN: CMPB #'A,(R5) ;ASSIGN ALL DRIVES?
6263 024466 001426 BEQ ASGN2 ;BR IF ALL DRIVES
6264 024470 111504 ASGN1: MOV (R5),R4 ;PUT DRIVE # IN R4
6265 024472 012737 053356 026754 MOV #UNTSN,ASNMSG ;'DRIVE ASSIGNED' MESSAGE ADDRESS
6266 024500 136437 034606 001544 BITB ATABIT(R4),ASNLST ;DRIVE ALREADY ASSIGNED ?
6267 024506 001014 BNE 2$ ;BR IF IT IS
6268 024510 005704 TST R4 ;TRYING TO ASSIGN DRIVE 0 ?
6269 024512 001007 BNE 1$ ;BR IF NOT
6270 024514 012737 053443 026754 MOV #NOTAVL,ASNMSG ;'NOT AVAILABLE' MESSAGE ADDRESS
6271 024522 122737 000024 000041 CMPB #24,41 ;SEE IF LOADED FROM AN RM03
6272 024530 001403 BEQ 2$ ;BR IF RM03 IS THE LOAD DEVICE
6273 024532 004737 024626 1$: JSR PC,ASGN3 ;SEE IF DRIVE ON THE SYSTEM
6274 024536 000207 RTS PC ;RETURN
6275 024540 000137 026734 2$: JMP ASNERR ;EXIT ERROR
6276 024544 122737 000024 000041 ASGN2: CMPB #24,41 ;LOADED FROM AN RM03 ?
6277 024552 001003 BNE 1$ ;BR IF NOT
6278 024554 012704 000001 MOV #1,R4 ;START WITH DRIVE 1
6279 024560 000401 BR 2$ ;CONTINUE
6280 024562 005004 1$: CLR R4 ;START WITH DRIVE 0
6281 024564 012737 053356 026754 2$: MOV #UNTSN,ASNMSG ;ERROR MESSAGE
6282 024572 136437 034606 001544 BITB ATABIT(R4),ASNLST ;ALREADY ASSIGNED ?
6283 024600 001007 BNE 4$ ;YES

```


6284	024602	004737	024626		JSR	PC,ASGN3	:ASSIGN THE DRIVE
6285	024606	005204		3\$:	INC	R4	:INCREMENT DRIVE #
6286	024610	022704	000007		CMP	#7,R4	:ALL DRIVE CHECKED ?
6287	024614	002363			BGE	2\$	:NO
6288	024616	000207			RTS	PC	:YES
6289	024620	004737	026734	4\$:	JSR	PC,ASNERR	:ERROR MESSAGE
6290	024624	000770			BR	3\$	:TO LOOP
6291	024626	136437	034606	001544	ASGN3:	BITB	:DRIVE ALREADY ASSIGNED ?
6292	024634	001041			BNE	ASGN4	:BR IF IT IS
6293				1\$:	TST	DTUW	:DATA TRANSFER UNDER WAY ?
6294				:	BPL	1\$	:BR IF IT IS
6295	024636	110437	046062		MOVB	R4,GENDPB	:DRIVE NUMBER
6296	024642	004737	015200		JSR	PC,RECALD	:RECALIBRATE DRIVE
6297	024646	105764	034472		TSTB	DRVSTA(R4)	:DRIVE AVAILABLE?
6298	024652	001440			BEQ	ASGN7	:BR IF DRIVE OFFLINE OR NONEXISTENT
6299	024654	100432			BMI	ASGN6	:BR IF DRIVE UNSAFE
6300	024656	006304			ASL	R4	:MAKE R4 INTO WORD INDEX
6301				:	MOV	BLKADR(R4),NEWUNT	:PUT BLOCK'S ADDR INTO R0
6302	024660	016400	002102		MOV	BLKADR(R4),R0	:CLEAR BLOCK FOR DRIVE JUST ASSIGNED
6303	024664	004737	025432		JSR	PC,CLRDPB	:GET THE DRIVE'S ADDRESS LIMITS
6304	024670	004737	025632		JSR	PC,DRVPRM	:GET DRIVE I.D.
6305	024674	004737	026044		JSR	PC,GETID	:GET BAD SECTOR ADDRESSES
6306	024700	004537	026154		JSR	R5,GETADR	:BRANCH IF RETRIEVE FAILS
6307	024704	000414			BR	2\$	:BR IF NOT
6308	024706	016464	002102	001570	MOV	BLKADR(R4),NEWUNT	:SET READ/WRITE DATA PACK INDICATOR
6309	024714	012760	000001	000070	MOV	#1,\$PASSC(R0)	:RESTORE DRIVE ADDRESS
6310	024722	005737	001316		TST	PACK	:RETURN
6311	024726	001403			BEQ	2\$	:UNSAFE' MESSAGE ADDRESS
6312	024730	113760	001316	000026	MOVB	PACK,\$PACK(R0)	:TO ERROR ROUTINE
6313	024736	006204		2\$:	ASR	R4	:DRIVE PRESENT?
6314	024740	000207		ASGN4:	RTS	PC	:BR IF NOT
6315	024742	012737	053462	026754	ASGN6:	MOV	:BR IF DRIVE OFFLINE
6316	024750	000137	026734		JMP	ASNERR	:ADDRESS OF 'NOT RML3' MSG
6317	024754	105764	034502		TSTB	DRVSTP(R4)	:EXIT
6318	024760	001405		ASGN7:	BEQ	1\$	:ADDRESS OF 'NOT PRESENT' MSG
6319	024762	100010			BPL	2\$	:EXIT
6320	024764	012737	053404	026754	MOV	#NOTRM,ASNMSG	:ADDRESS OF 'DRIVE OFFLINE' MESSAGE
6321	024772	000407			BR	3\$	:ERROR RETURN
6322	024774	012737	053426	026754	1\$:	MOV	:TO ERROR ROUTINE
6323	025002	000403			BR	3\$	
6324	025004	012737	053313	026754	2\$:	MOV	
6325					3\$:	RTS	
6326	025012	000137	026734		3\$:	JMP	
6327							
6328							
6329							
6330	025016	005037	001316		NEWASN:	CLR	:CLEAR 'W' COMMAND INDICATOR
6331	025022	000137	024462		JMP	ASSIGN	:GO TO THE ASSIGN ROUTINE
6332							
6333							
6334							
6335	025026	005004			DEASGN:	CLR	
6336	025030	012703	000010		MOV	R4	:COUNTER
6337	025034	122715	000101		CMPB	#8,R3	:DEASSIGN ALL DRIVES ?
6338	025040	001403			BEQ	#A,(R5)	:BR IF YES
6339	025042	012703	000001		MOV	#BIT00,R3	:SET R3 FOR ONE UNIT

```

6340 025046 111504          MOV      (R5),R4          ;GET DRIVE NUMBER
6341 025050 136437 034606 001544 1$: BITB      ATABIT(R4),ASNLS1 ;DRIVE ASSIGNED ?
6342 025056 001414          BEQ        3$              ;BR IF NOT
6343 025060 146437 034606 001544          BICB      ATABIT(R4),ASNLS1 ;DELETE THE DRIVE FROM THE ASSIGNED LIST
6344 025066 006304          ASL        R4              ;MAKE ADDR INTO A WORD INDEX
6345 025070 015464 002102 001546          MOV      BLKADR(R4),DUNIT(R4) ;PUT ADDRESS IN DEASSIGN LIST
6346 025076 076204          ASR        R4
6347 025100 005303          2$: DEC        R3              ;ANY MORE DRIVES ?
6348 025102 001412          BEQ        4$              ;BR IF NOT
6349 025104 005204          INC        R4
6350 025106 000760          BR         1$
6351 025110 012737 053334 026754 3$: MOV      #UNTNOT,ASNMSG ;ADDR OF 'NOT ASSIGNED' MESSAGE
6352 025116 004737 026734          JSR      PC,ASNEAR ;REPORT IT
6353 025122 122715 000101          CMPB     #'A',(R5) ;ALL DRIVE ?
6354 025126 001764          BEQ        2$              ;YES
6355 025130 104401 001201          4$: TYPE     $SCLF
6356 025134 000207          RTS      PC
6357
6358 ;'S' COMMAND (ROUTINE TO TYPE DRIVE PERFORMANCE SUMMARY)
6359
6360 025136 005004          SCMD:  CLR      R4
6361 025140 012703 000010          MOV      #8,R3 ;COUNTER
6362 025144 122715 000101          CMPB     #'A',(R5) ;ALL STATISTICS ?
6363 025150 001421          BEQ        3$              ;BR IF YES
6364 025152 111504          MOV      (R5),R4          ;GET DRIVE NUMBER
6365 025154 136437 034606 001544          BITB      ATABIT(R4),ASNLS1 ;SEE IF DRIVE ASSIGNED
6366 025162 001406          BEQ        1$              ;BR IF NOT
6367 025164 006304          ASL        R4              ;MAKE DRIVE ADDR INTO WORD INDEX
6368 025166 016400 002102          MOV      BLKADR(R4),R0 ;ADDR OF BLOCK
6369 025172 004737 023122          JSR      PC,TYPEST ;TYPE DRIVE STATISTICS
6370 025176 000471          BR         8$              ;EXIT
6371 025200 012737 053334 026754 1$: MOV      #UNTNOT,ASNMSG ;ADDR OF 'NOT ASSIGNED' MSG
6372 025206 004737 026734          JSR      PC,ASNEAR ;TYPE ERROR MESSAGE
6373 025212 000463          BR         8$              ;EXIT
6374 025214 105737 001544          3$: TSTB     ASNLS1 ;ANY DRIVE ASSIGNED ?
6375 025220 001001          BNE        4$              ;YES
6376 025222 000457          BR         8$              ;RETURN
6377 025224 004737 023034          4$: JSR      PC,STATPR ;TYPE ALL STATISTICS
6378 025230 105737 001320          TSTB     DATE ;SEE IF 'DATE' ENTERED
6379 025234 001404          BEQ        11$             ;BR IF NOT
6380 025236 104401 054332          TYPE     ,DATEIS ;'DATE:'
6381 025242 104401 001320          TYPE     ,DATE ;THE OPERATOR ENTERED DATE
6382 025246 105737 001332          11$: TSTB     OPERID ;SEE IF OPERATOR I.D. ENTERED
6383 025252 001404          BEQ        12$             ;BR IF NOT
6384 025254 104401 054343          TYPE     ,IDIS ;'OPERATOR I.D.: '
6385 025260 104401 001332          TYPE     ,OPERID ;THE OPERATOR I.D.
6386 025264 104401 054365          TYPE     ,HEDLIN ;HEADER LINE
6387 025270 012737 043246 025336 12$: MOV      #DRIVED+$DRVID,6$ ;DRIVE I.D. FIELD ADDRESS
6388 025276 136437 034606 001544 5$: BITB      ATABIT(R4),ASNLS1 ;SEE IF DRIVE ASSIGNED
6389 025304 001417          BEQ        7$              ;BR IF NOT ASSIGNED
6390 025306 010446          MOV      R4,-(SP) ;SAVE R4 FOR TYPEOUT
6391
6392 025310 104403          TYPE     ,TYPE DRIVE NUMBER
6393 025312 002          TYPE     ,GO TYPE--OCTAL ASCII
6394 025313 000          TYPE     ,TYPE 2 DIGIT(S)
6395 025314 104401 053244          TYPE     ,SUPPRESS LEADING ZEROS
6396                                TYPE     ,4 SPACES

```

```

6396 025320 105777 000012      TSTB      26$      ;SEE IF DRIVE I.D. ENTERED
6397 025324 001003      BNE      10$      ;BR IF DRIVE I.D. PRESENT
6398 025326 104401 054410      TYPE      NONE      ;TYPE 'NONE'
6399 025332 000404      BR      7$      ;CONTINUE
6400 025334 104401      10$: TYPE      ;TYPE THE DRIVE I.D.
6401 025336 000000      6$: .WORD      0      ;ADDRESS OF DRIVE I.D. FIELD HERE
6402 025340 104401 001201      TYPE      SCRLF      ;CR-LF
6403 025344 005303      7$: DEC      R3      ;DECREMENT THE COUNTER
6404 025346 003405      BLE      8$      ;BR IF AT END
6405 025350 062737 000304 025336 ADD      SRMEC2+2,6$ ;INCREMENT THE MESSAGE FIELD ADDRESS
6406 025356 005204      INC      R4      ;INCREMENT DRIVE ADDRESS
6407 025360 000746      BR      5$      ;CONTINUE
6408 025362 104401 001201      8$: TYPE      SCRLF      ;CR-LF
6409 025366 000207      9$: RTS      PC
6410
6411      ;'W' COMMAND (ROUTINE TO WRITE A DATA PACK)
6412
6413 025370 012737 177777 001316 DATAPK: MOV      #-1,PACK      ;SET THE 'W' COMMAND INDICATOR
6414 025376 000137 024462      JMP      ASSIGN      ;ASSIGN REQUESTED DRIVE
6415
6416      ;'WT' COMMAND (TO WRITE A PACK AND FOLLOWED BY TEST PACK)
6417 025402 116515 000001      WATPAK: MOVB      1(R5), (R5) ;ADJUST DRIVE NUMBER ADDRESS
6418 025406 177777 001316      MOV      #-2,PACK      ;PACK WRITE COMMAND
6419 025414 000137 024462      JMP      ASSIGN      ;JUMP TO ASSIGN ROUTINE
6420
6421      ;'R' COMMAND (ROUTINE TO READ A DATA PACK)
6422 025420 012737 000001 001316 REDAPK: MOV      #1,PACK      ;SET THE 'READ' INDICATOR
6423 025426 000137 024462      JMP      ASSIGN      ;ASSIGN THE REQUESTED DRIVE
6424
6425      ;ROUTINE TO CLEAR THE DPB FOR THE ASSIGNED DRIVE
6426      CALL:
6427      MOV      #DPB,R0      ;DPB ADDRESS
6428      JSR      PC,CLRDPB
6429      RETURN
6430
6431      CLRDPB:
6432 025432 010146      MOV      R1,-(SP)      ;PUSH R1 ON STACK
6433 025434 010346      MOV      R3,-(SP)      ;PUSH R3 ON STACK
6434 025436 010446      MOV      R4,-(SP)      ;PUSH R4 ON STACK
6435 025440 010546      MOV      R5,-(SP)      ;PUSH R5 ON STACK
6436 025442 010004      MOV      R0,R4      ;GET THE DPB ADDRESS
6437 025444 062704 000002      ADD      #2,R4      ;ADDRESS OF FIRST LOCN TO BE CLEARED
6438 025450 012703 000005      MOV      #5,R3      ;NUMBER OF LOCNS TO BE CLEARED
6439 025454 005024      1$: CLR      (R4)+      ;CLEAR THE LOCATION
6440 025456 005303      DEC      R3      ;DECREMENT THE COUNTER
6441 025460 001375      BNE      1$      ;BR IF NOT FINISHED
6442 025462 062704 000002      ADD      #2,R4      ;MOVE THE ADDRESS PAST THE 'REG' ADDR
6443 025466 012703 000070      MOV      #NEXT-$REG,R3 ;NUMBER OF LOCNS TO BE CLEARED
6444 025472 005024      2$: CLR      (R4)+      ;CLEAR
6445 025474 162703 000002      SUB      #2,R3      ;DECREMENT THE LOCN COUNTER
6446 025500 001374      BNE      2$      ;BR IF NOT FINISHED
6447 025502 062704 000014      ADD      #12,R4      ;MOVE PAST ADDRESS LIMITS
6448 025506 012703 000162      MOV      #SRMEC2-MINSEC,R3 ;NUMBER OF LOCNS TO BE CLEARED
6449 025512 005024      3$: CLR      (R4)+      ;CLEAR A LOCATION
6450 025514 162703 000002      SUB      #2,R3      ;DECREMENT THE COUNTER
6451 025520 001374      BNE      3$      ;BR IF NOT DONE

```

6452	025522	113760	00151E	000024	MOV	BEGCOD,\$CODE(R0)	: INITIAL COMMAND CODE	
6453	025530	013701	001516		MOV	BEGCOD,R1	: GET THE ACTUAL OP CODE	
6454	025534	116160	002122	000002	MOV	COMTBL(R1),\$COMND(R0)	: OPERATION CODE	
6455	025542	113760	001514	000030	MOV	BEGPAT,\$PATTC(R0)	: PATTERN CODE	
6456	025550	106360	000030		ASLB	\$PATTC(R0)	: CONVERT CODE TO A TABLE INDEX	
6457	025554	013760	001520	000020	MOV	BEGSZ,\$SRDL(R0)	: BEGINNING RECORD SIZE	
6458	025562	013760	001520	000004	MOV	BEGSZ,\$SRDL(R0)	: VALUE FOR DATA TRANSFER	
6459	025570	005460	000004		NEG	\$SRDL(R0)	: MAKE IT INTO 2'S COMPLEMENT	
6460	025574	012760	000400	000022	MOV	#256,\$SSEC(R0)	: INITIAL VALUE OF SECTOR SIZE	
6461	025602	132760	000001	000024	BITB	#1,\$CODE(R0)	: HEADER ORDER ?	
6462	025610	001403			BEQ	4\$	: BR IF NOT	
6463	025612	062760	000002	000022	ADD	#2,\$SSEC(R0)	: ADD HEADER SIZE TO SECTOR SIZE	
6464	025620				4\$:			
6465	025620	012605			MOV	(SP)+,R5	: POP STACK INTO R5	
6466	025622	012604			MOV	(SP)+,R4	: POP STACK INTO R4	
6467	025624	012603			MOV	(SP)+,R3	: POP STACK INTO R3	
6468	025626	012601			MOV	(SP)+,R1	: POP STACK INTO R1	
6469	025630	000207			RTS	PC	: RETURN	
6470								
6471							: ROUTINE TO GET ADDRESS LIMITS FROM THE OPERATOR	
6472								
6473	025632	010346			DRVPRM: MOV	R3,-(SP)	: SAVE R3	
6474	025634	010446			MOV	R4,-(SP)	: SAVE R4	
6475	025636	005737	000042		TST	42	: RUNNING UNDER MONITOR CONTROL	
6476	025642	001010			BNE	3\$	: BR IF YES	
6477	025644	104401	054226		TYPE	ENTLMT	: 'ENTER ADDRESSES'	
6478	025650	006204			ASR	R4	: CONVERT INDEX TO DRIVE NUMBER	
6479	025652	010446			MOV	R4,-(SP)	: SAVE R4 FOR TYPEOUT	
6480							: TYPE DRIVE NUMBER	
6481	025654	104403			TYPOS		: GO TYPE--OCTAL ASCII	
6482	025656	002			.BYTE	2	: TYPE 2 DIGIT(S)	
6483	025657	000			.BYTE	0	: SUPPRESS LEADING ZEROS	
6484					:			
6485					TYPE	SLASH	: '/'	
6486					MOV	\$RM03A,2\$	: ADDRESS OF 'RM03' MESSAGE	
6487					BITB	#BIT02,DRVTP(R4)	: RM03 ?	
6488					BNE	1\$	: BR IF YES	
6489					HALT		: NOT RM03 PROGRAM SHOULD NOT HAVE	
6490							: REACHED THIS POINT	
6491					1\$:	TYPE	: TYPE THE MESSAGE WHICH FOLLOWS	
6492					2\$:	.WORD	: MESSAGE ADDRESS	
6493	025660	104401	001201		TYPE	\$CRLF	: CR-LF	
6494	025664	012737	001465	001446	3\$:	MOV	#821,CYLIMT	: ASSUME AN RM03
6495	025672	132764	000004	034502	BITB	#BIT02,DRVTP(R4)	: MAKE SURE RM03	
6496	025700	001001			BNE	4\$	: BR IF RM03	
6497	025702	000000			HALT		: HALT IF NOT	
6498	025704	116460	034502	000262	4\$:	MOV	DRVTP(R4),\$RMDT(R0)	: INITIATE \$RMDT
6499	025712	062760	177777	000122	ADD	#-1,\$FIRST(R0)	: SEE IF FIRST TIME STARTED	
6500	025720	103417			BCS	5\$	: BR IF NOT	
6501	025722	013760	001446	000106	MOV	CYLIMT,MAXCYL(R0)	: LOAD MAXIMUM CYLINDER	
6502	025730	005060	000110		CLR	MINCYL(R0)	: CLEAR MINIMUM CYLINDER	
6503	025734	013760	001444	000112	MOV	TRKIMT,MAXTRK(R0)	: LOAD MAXIMUM TRACK	
6504	025742	005060	000114		CLR	MINTRK(R0)	: CLEAR MINIMUM TRACK	
6505	025746	013760	001442	000116	MOV	SECLMT,MAXSEC(R0)	: LOAD MAXIMUM SECTOR	
6506	025754	005060	000120		CLR	MINSEC(R0)	: CLEAR MINIMUM SECTOR	
6507	025760	006304			5\$:	ASL	R4	: SETUP TO ADDRESS WORDS
6508	025762	016403	055252		MOV	TABLE(R4),R3	: PARAMETER TABLE ADDRESS	

```

6508 025766 013763 001446 000002      MOV      CYLIMT,2(R3)      ;LOAD CYLINDER LIMIT FOR LAST CYLINDER
6509 025774 013763 001446 000010      MOV      CYLIMT,10(R3)     ;LOAD CYLINDER LIMIT FOR STARTING CYLINDER
6510 026002 005737 000042      TST      42              ;UNDER MONITOR CONTROL ?
6511 026006 001002 000042      BNE      6$              ;BR IF YES
6512 026010 004737 026610      JSR      PC,PARENT       ;GET THE DRIVE'S PARAMETERS
6513 026014 116060 000120 000010 6$:  MOV      MINSEC(R0),SSEC(R0) ;INITIAL SECTOR VALUE
6514 026022 116060 000114 000011      MOV      MINTRK(R0),STAK(R0) ;INITIAL TRACK VALUE
6515 026030 016060 000110 000012      MOV      MINCYL(R0),SCYL(R0) ;INITIAL CYLINDER VALUE
6516 026036 012604      MOV      (SP)+,R4       ;RESTORE R4
6517 026040 012603      MOV      (SP)+,R3       ;RESTORE R3
6518 026042 000207      RTS      PC            ;RETURN
6519
6520      ;ROUTINE TO GET THE DRIVE I.D. FROM THE OPERATOR
6521
6522 026044 010546      GETID:  MOV      R5,-(SP)      ;SAVE R5
6523 026046 005737 000042      TST      42              ;UNDER MONITOR CONTROL ?
6524 026052 001036      BNE      2$              ;BR IF NOT
6525 026054 005037 001360 1$:  CLR      CFLAG          ;CLEAR THE 'CONTROL C' FLAG
6526 026060 104401 054201      TYPE     ,ENTDRV        ;'ENTER DRV I.D.:'
6527 026064 005046      CLR      -(SP)         ;CLEAR THE STACK
6528 026066 111016      MOV      (R0),(SP)      ;PUT THE DRIVE NUMBER ON THE STACK
6529 026070 104403      TYPPOS    ;TYPE THE DRIVE NUMBER
6530 026072 002      .BYTE     2          ;TYPE 2 DIGITS
6531 026073 000      .BYTE     0          ;SUPPRESS LEADING ZEROS
6532 026074 104401 001201      TYPE     ,SCRLF        ;CR-LF
6533 026100 104411      RDLIN     ;READ THE ENTRY
6534 026102 012605      MOV      (SP)+,R5      ;GET THE ENTRY ADDRESS
6535 026104 005737 001360      TST      CFLAG          ;'CONTROL C' ENTERED ?
6536 026110 001361      BNE      1$              ;BR IF IT WAS
6537 026112 121527 000056      CMPB     (R5),#'.      ;PERIOD ENTERED ?
6538 026116 001414      BEQ      2$              ;BR IF YES
6539 026120 112560 000224      MOV      (R5)+,$DRVID(R0) ;STORE THE I.D.
6540 026124 112560 000225      MOV      (R5)+,$DRVID+1(R0) ;STORE THE I.D.
6541 026130 112560 000226      MOV      (R5)+,$DRVID+2(R0) ;STORE THE I.D.
6542 026134 112560 000227      MOV      (R5)+,$DRVID+3(R0) ;STORE THE I.D.
6543 026140 112560 000230      MOV      (R5)+,$DRVID+4(R0) ;STORE THE I.D.
6544 026144 112560 000231      MOV      (R5)+,$DRVID+5(R0) ;STORE THE I.D.
6545 026150 012605 2$:  MOV      (SP)+,R5      ;RESTORE R5
6546 026152 000207      RTS      PC            ;RETURN
6547
6548      ;ROUTINE TO GET THE ADDRESSES OF ANY BAD SECTORS (UP TO A MAX OF 16)
6549      ;CALL SEQ
6550      JSR      R5,GETADR
6551      RET1     ERROR RET
6552      RET2     NORMAL RET
6553
6554      ;RD DPB ADDRESS
6555
6556 026154      GETADR:  MOV      R1,-(SP)      ;PUSH R1 ON STACK
6557 026154 010146      MOV      R2,-(SP)      ;PUSH R2 ON STACK
6558 026156 010246      MOV      R3,-(SP)      ;PUSH R3 ON STACK
6559 026160 010346      MOV      R4,-(SP)      ;PUSH R4 ON STACK
6560 026162 010446      MOV      R0,R1         ;DPB ADDRESS
6561 026164 010001      ADD      #SBDSEC,R1     ;ADDRESS OF BAD SECTOR TABLE
6562 026166 062701 000124      MOV      #32.,R2       ;WORD CTR
6563 026172 012702 000040

```

6564	026176	012721	177777		1\$:	MOV	#-1,(R1)+	; INITIALIZE ALL LOCS TO -1
6565	026202	005302				DEC	R2	; DECREMENT WORD CTR
6566	026204	001374				BNE	1\$	; BRANCH IF NOT DONE
6567	026206	112737	000171	046064		MOV	#RDAT,GENDPB+\$COMND	; READ DATA COMMAND
6568	026214	012737	001466	046074		MOV	#B22,GENDPB+\$CYL	; LAST CYLINDER
6569	026222	112737	000004	046073		MOV	#4,GENDPB+\$TRK	; LAST TRACK
6570	026230	012737	177400	046066		MOV	#-256,GENDPB+\$WRDM	; ONE SECTOR WORD CTR
6571	026236	111037	046062			MOV	(R0),GENDPB	; DRIVE NUMBER
6572	026242	005037	001264			CLR	\$CDW1	; ERROR FLAG
6573	026246	005046				CLR	-(SP)	; DUMMY PAIR
6574	026250	005046				CLR	-(SP)	; DUMMY PAIR
6575	026252	012746	000012			MOV	#12,-(SP)	; STARTING SECTOR 10.
6576	026256	012746	000036			MOV	#30,-(SP)	; LAST SECTOR
6577	026262	012737	000010	001266		MOV	#10,\$CDW2	; LAST SECTOR
6578	026270	105037	046072			CLRB	GENDPB+\$SEC	; STARTING SECTOR
6579	026274	004037	035364		2\$:	JSR	R0,RM03	; READ CURRENT SECTOR
6580	026300	046062				GENDPB		; BRANCH IF QUEUE FAILS
6581	026302	000774				BR	2\$	
6582	026304	005737	046100		3\$:	TST	GENDPB+\$STATUS	; DONE YET ?
6583	026310	001775				BEQ	3\$	; BRANCH IF NOT
6584	026312	100013				BPL	4\$	; BRANCH IF DONE WITHOUT ERROR
6585	026314	062737	000002	046072		ADD	#2,GENDPB+\$SEC	; INCREMENT TO NEXT 2ND SECTOR
6586	026322	123737	001266	046072		CMPB	\$CDW2,GENDPB+\$SEC	; ALL SECTORS ARE TRIED ?
6587	026330	103361				BHIS	2\$	; BRANCH IF NOT
6588	026332	052737	000001	001264		BIS	#BIT0,\$CDW1	; FLAG-RETRIEVE FAILS
6589	026340	000451				BR	9\$	; NEXT SETS
6590	026342	010001			4\$:	MOV	R0,R1	; BAD SECTOR TABLE
6591	026344	062701	000124			ADD	#80SEC,R1	; BAD SECTOR TABLE ADDRESS
6592	026350	012702	000040			MOV	#32,R2	; WORD CTR OF BAD SECTOR TABLE
6593	026354	012703	056512			MOV	#CYLDER,R3	; BUFFER ADDRESS
6594	026360	062703	000010			ADD	#10,R3	; STARTS AT THE 5TH WORD
6595	026364	012704	000374			MOV	#252,R4	; WORD CTR FOR BAD SPOT FILE
6596	026370	022713	177777		5\$:	CMP	#-1,(R3)	; END OF BAD SPOT FILE ?
6597	026374	001433				BEQ	9\$	; BRANCH IF SO
6598	026376	022711	177777		6\$:	CMP	#-1,(R1)	; ENTRY OF THE TABLE ?
6599	026402	001406				BEQ	7\$	; BRANCH IF SO
6600	026404	062701	000004			ADD	#4,R1	; ADJUST TABLE ENTRY
6601	026410	162702	000002			SUB	#2,R2	; DECREMENT WORD CTR
6602	026414	002420				BLT	8\$	; OVER 16 BAD SECTOR DETECTED
6603	026416	000767				BR	6\$	; LOOP BACK
6604	026420	011311			7\$:	MOV	(R3),(R1)	; LOAD THE TABLE
6605	026422	016361	000002	000002		MOV	2(R3),2(R1)	; TRK AND SECTOR
6606	026430	162704	000002			SUB	#2,R4	; DECREMENT WORD CTR FOR FILE
6607	026434	003413				BLE	9\$	; BRANCH IF DONE
6608	026436	062703	000004			ADD	#4,R3	; ADJUST FILE ADDRESS
6609	026442	162702	000002			SUB	#2,R2	; DECREMENT TABLE WORD CTR
6610	026446	003403				BLE	8\$	; TABLE ENTRY EXHAUSTED
6611	026450	062701	000004			ADD	#4,R1	; ADJUST TABLE ENTRY
6612	026454	000745				BR	5\$	; LOOP BACK
6613	026456	052737	000002	001264	8\$:	BIS	#BIT1,\$CDW1	; ERROR FLAG-OVER 16 BAD SECTORS
6614	026464	012637	001266		9\$:	MOV	(SP)+,\$CDW2	; UPDATE LAST SECTOR
6615	026470	112637	046072			MOV	(SP)+,GENDPB+\$SEC	; STARTING SECTOR
6616	026474	001277				BNE	2\$	; BRANCH IF NOT DUMMY PAIR
6617	026476	032737	000001	001264		BIT	#BIT0,\$CDW1	; RETRIEVE FAILS ?
6618	026504	001412				BEQ	10\$	; BRANCH IF NOT
6619	026506	104401	054575			TYPE	,MERR1	; FAILS TO RETRIEVE BAD SPOT FILE


```

6620 026512 104401 053305      TYPE      UNTMSG      ; ON DRIVE
6621 026516 111046      MOVVB      (R0),-(SP)      ;
6622 026520 104403      TYPDS      ;
6623 026522      002      .BYTE      2
6624 026523      000      .BYTE      0
6625 026524 104401 001201      TYPE      $CRLF      ; CR-LF
6626 026530 000415      BR      11$      ; EXIT
6627 026532 032737 000002 001264 10$: BIT      $BIT1,$CDW1      ; OVER 16 BAD SECTORS ?
6628 026540 001411      BEQ      11$      ; BRANCH IF NOT
6629 026542 104401 054643      TYPE      $ERR2      ; OVER 16 BAD SECTORS DETECTED
6630 026546 104401 053305      TYPE      UNTMSG      ; ON DRIVE
6631 026552 111046      MOVVB      (R0),-(SP)      ;
6632 026554 104403      TYPDS      ;
6633 026556      002      .BYTE      2
6634 026557      000      .BYTE      0
6635 026560 104401 001201      TYPE      $CRLF      ;
6636 026564 005737 001264      11$: TST      $CDW1      ; ERROR FLAG SET ?
6637 026570 001002      BNE      12$      ; BRANCH IS SET
6638 026572 062705 000002      ADD      $2,R5      ; ADJUST FOR NORMAL RET
6639 026576      12$:
6640 026576 012604      MOV      (SP)+,R4      ; POP STACK INTO R4
6641 026600 012603      MOV      (SP)+,R3      ; POP STACK INTO R3
6642 026602 012602      MOV      (SP)+,R2      ; POP STACK INTO R2
6643 026604 012601      MOV      (SP)+,R1      ; POP STACK INTO R1
6644 026606 000205      RTS      R5      ; EXIT
6645
6646      ;PARAMETER ENTRY ROUTINE
6647      ;CALL
6648      ;
6649      ;      MOV      $ADR,R3      ;PARAMETER TABLE ADDRESS
6650      ;      JSR      PC,PARENT      ;GET THE PARAMETERS
6651 026610 010346      PARENT: MOV      R3,-(SP)      ;SAVE THE PARAMETER TABLE ADDRESS
6652 026612 005037 001360      CLR      CFLAG      ;CLEAR THE 'CONTROL C' FLAG
6653 026616 012337 026626      1$: MOV      (R3)+,$3      ;ADDRESS OF PARAMETER NAME
6654 026622 001442      BEQ      9$      ;BR IF AT END OF TABLE
6655 026624 104401      TYPE      ;TYPE THE PARAMETER NAME
6656 026626 000000      .WORD      0      ;ADDRESS OF PARAMETER NAME TEXT
6657 026630 012302      MOV      (R3)+,R2      ;MAXIMUM PARAMETER VALUE
6658 026632 012305      MOV      (R3)+,R5      ;ADDRESS OF PARAMETER
6659 026634 011546      MOV      (R5),-(SP)      ;CURRENT VALUE OF PARAMETER
6660 026636 104405      TYPDS      ;TYPE THE CURRENT VALUE OF THE PARAMETER
6661 026640 104401 055036      TYPE      .SLASH      ; '/'
6662 026644 104411      RDLIN      ;READ THE KEYBOARD
6663 026646 012601      MOV      (SP)+,R1      ;INPUT ASCII STRING ADDRESS
6664 026650 005737 001360      TST      CFLAG      ;'CONTROL C' ENTERED ?
6665 026654 001021      BNE      8$      ;BR IF IT WAS
6666 026656 004537 030300      JSR      R5,CK.DIG      ;CHECK THE DIGIT(S)
6667 026662 026616      1$      ;CARRIAGE RETURN ONLY ENTERED
6668 026664 026730      9$      ;PERIOD ONLY ENTERED
6669 026666 026702      6$      ;ILLEGAL INPUT
6670 026670 026676      5$      ;TERMINATED WITH A CARRIAGE RETURN
6671 026672 026702      6$      ;TERMINATED WITH A "."
6672 026674 026714      7$      ;TERMINATED WITH A "!"
6673 026676 010215      5$: MOV      R2,(R5)      ;MOVE NEW VALUE TO PARAMETER LOCATION
6674 026700 000746      BR      1$      ;GET MORE PARAMETERS
6675 026702 104401 054417      6$: TYPE      .BADENT      ;'BAD ENTRY'

```

```

6676 026706 162703 000006      SUB      #6,R3      ;DECREMENT THE TABLE POINTER
6677 026712 000741              BR      1$          ;TRY AGAIN
6678 026714 010215      7$:      MOV      R2,(R5)      ;NEW VALUE
6679 026716 000404              BR      9$          ;EXIT
6680 026720 005037 001360      8$:      CLR      CFLAG      ;CLEAR THE 'CONTROL C' FLAG
6681 026724 011603              MOV      (SP),R3      ;RELOAD THE PARAMETER TABLE ADDRESS
6682 026726 000733              BR      1$          ;TRY AGAIN
6683 026730 005726      9$:      TST      (SP)+      ;CORRECT THE STACK POINTER
6684 026732 000207              RTS      PC          ;RETURN

;TYPEOUT ASSIGN/DEASSIGN ERROR MESSAGE
;CALL
;      MOV      #MESADR,ASNMSG ;ERROR MESSAGE ADDRESS
;      JSR      PC,ASNERR
;      RETURN

6692 026734 104401 054130      ASNERR: TYPE      ,QUES      ;QUESTION MARK
6693 026740 104401 053305      TYPE      ,UNMSG      ;TYPE 'DRIVE'
6694 026744 010446              MOV      R4,-(SP)      ;SAVE R4 FOR TYPEOUT
6695                          ;TYPE DRIVE NUMBER
6696 026746 104403              TYP0S      ;GO TYPE--OCTAL ASCII
6697 026750 002              .BYTE      2      ;TYPE 2 DIGIT(S)
6698 026751 000              .BYTE      0      ;SUPPRESS LEADING ZEROS
6699 026752 104401              TYPE      ;TYPE SPECIFIC MESSAGE
6700 026754 000000      ASNMSG: .WORD      0      ;MESSAGE ADDRESS
6701 026756 104401 001201      TYPE      ,%CRLF
6702 026762 000207      RTS      PC

;DEASSIGN DRIVE IF A FATAL ERROR OCCURS
;CALL
;      JSR      PC,DROP
;      RETURN

6709 026764 005004      DROP:   CLR      R4          ;CLEAR R4 FOR DRIVE NUMBER
6710 026766 111004      MOV8      (R0),R4      ;MOVE DRIVE NUMBER TO R4
6711 026770 146437 034606 001544 BICB      ATABIT(R4),ASNLS1 ;REMOVE DRIVE FROM ASSIGNED LIST
6712 026776 006304      ASL      R4          ;MAKE DRIVE NUMBER INTO A TABLE INDEX
6713 027000 010064      MOV      R0,DUNIT(R4) ;PUT DRIVE IN DROP LIST
6714 027004 104401 001201      TYPE      ,%CRLF
6715 027010 104401 001201      TYPE      ,%CRLF
6716 027014 104401 053703      TYPE      ,DROPNG      ;TYPE 'DROPPING DRIVE'
6717 027020 104401 054014      TYPE      ,DRNUM      ;'DRIVE #'
6718 027024 006204      ASR      R4          ;DRIVE NUMBER
6719 027026 010446      MOV      R4,-(SP)      ;SAVE R4 FOR TYPEOUT
6720                          ;TYPE DRIVE NUMBER
6721 027030 104403      TYP0S      ;GO TYPE--OCTAL ASCII
6722 027032 002              .BYTE      2      ;TYPE 2 DIGIT(S)
6723 027033 000              .BYTE      0      ;SUPPRESS LEADING ZEROS
6724 027034 104401 001201      TYPE      ,%CRLF
6725 027040 105737 001544      TSTB      ASNLS1      ;MORE DRIVES ACTIVE
6726 027044 001006      BNE      1$          ;YES
6727 027046 005737 000042      TST      0#42      ;ANY MONITOR ?
6728 027052 001403      BEQ      1$          ;NO
6729 027054 005726      TST      (SP)+      ;CLEAR STACK
6730 027056 000137 027440      JMP      $GET42      ;GIVE CONTROL TO MONITOR
6731 027062 000207      1$:      RTS      PC

```



```

6732
6733 ;ROUTINE TO DEASSIGN DRIVE IF ERRORS BECOMES EXCESSIVE
6734
6735 027064 032777 000020 152062 ABNRML: BIT #SW04,2SWR ;SEE IF SWITCH 4 SET
6736 027072 001006 000056 1$ BNE 1$ ;BR IF IT'S SET
6737 027074 023760 001464 000056 CMP MAXER,$TOTAL(R0) ;CHECK TOTAL ERROR VALUE
6738 027102 103002 026764 BHS 1$ ;BR IF ERRORS DONOT EXCEED MAX
6739 027104 000137 026764 JMP DROP ;DEASSING THE DRIVE
6740 027110 000207 1$: RTS PC ;RETURN
6741
6742 ;ROUTINE TO CHECK FOR END OF PASS AND END OF TEST
6743
6744 .SBTTL END OF PASS ROUTINE
6745
6746 ;*****
6747 ;*INCREMENT THE PASS NUMBER ($PASS)
6748 ;*IF THERES A MONITOR GO TO IT
6749 ;*IF THERE ISN'T JUMP TO FISK
6750
6751 027112 $EOP:
6752 027112 005737 001506 EOP: TST ENDET ;END OF PASS DETERMINED BY SEEKS OR WORDS ?
6753 027116 001412 BEQ EOP1 ;BR IF SEEKS
6754 027120 026037 000054 001452 CMP $READ+2(R0),ENDCON+2 ;CHECK MSW OF WORDS READ COUNT
6755 027126 101017 BHI EOP2 ;BR IF MSW GREATER THAN LIMIT
6756 027130 103527 BLO EOPX ;BR IF MSW LESS THAN LIMIT
6757 027132 026037 000052 001450 CMP $READ(R0),ENDCON ;CHECK LSW AGAINST LIMIT
6758 027140 103012 BHS EOP2 ;BR IF EQUAL OR GREATER
6759 027142 000522 BR EOPX ;EXIT
6760 027144 026037 000044 001456 EOP1: CMP $POSIT+2(R0),ENDSEK+2 ;CHECK MSW OF SEEK COUNT
6761 027152 101005 BHI EOP2 ;BR IF MSW GREATER THAN LIMIT
6762 027154 103515 BLO EOPX ;EXIT IF MSW LESS THAN LIMIT
6763 027156 026037 000042 001454 CMP $POSIT(R0),ENDSEK ;CHECK LSW OF SEEK COUNT
6764 027164 103511 BLO EOPX ;EXIT IF LSW LESS THAN LIMIT
6765 027166 104401 001201 EOP2: TYPE , $CRLF ;CR-LF
6766 027172 104401 053737 TYPE ,ENDPAS ;END OF PASS FOR THE DRIVE
6767 027176 016046 000070 MOV $PASSC(R0),-(SP) ;PUT PASS COUNT ON THE STACK
6768 027202 104405 TYPDS ;CONVERT PASS COUNT TO DECIMAL AND TYPE IT
6769 027204 111037 001344 MOV8 (R0),UNIT ;STORE THE DRIVE NUMBER
6770 027210 032777 000020 151736 BIT #SW04,2SWR ;SWITCH 4 SET ?
6771 027216 001017 BNE 1$ ;BR IF SET
6772 027220 026037 000070 001460 CMP $PASSC(R0),PASCNT ;SEE IF AT END OF TEST
6773 027226 103413 BLO 1$ ;BR IF NOT
6774 027230 104401 053753 TYPE ,ENDTST ;TYPE 'END OF TEST'
6775 027234 104401 054014 TYPE ,DRNUM ;'DRIVE #'
6776 027240 013746 001344 MOV UNIT,-(SP) ;SAVE UNIT FOR TYPEOUT
6777
6778 027244 104403 TYPDS ;TYPE DRIVE NUMBER
6779 027246 002 .BYTE 2 ;GO TYPE--OCTAL ASCII
6780 027247 000 .BYTE 0 ;TYPE 2 DIGIT(S)
6781 027250 104401 001201 TYPE , $CRLF ;SUPPRESS LEADING ZEROS
6782 027254 000431 BR 3$ ;CR-LF
6783 027256 104401 054014 1$: TYPE ,DRNUM ;DEASSIGN THE DRIVE
6784 027262 013746 001344 MOV UNIT,-(SP) ;'DRIVE #'
6785
6786 027266 104403 TYPDS ;SAVE UNIT FOR TYPEOUT
6787 027270 002 .BYTE 2 ;TYPE DRIVE NUMBER
;GO TYPE--OCTAL ASCII
;TYPE 2 DIGIT(S)

```

```

6788 027271 000 .BYTE 0 ; SUPPRESS LEADING ZEROS
6789 027272 104401 001201 TYPE $SCRLF ; CR-LF
6790 027276 004737 023122 JSR PC,TYPEST ; TYPE THE DRIVE'S STATISTICS
6791 027302 010346 MOV R3,-(SP) ; SAVE R3
6792 027304 010446 MOV R4,-(SP) ; SAVE R4
6793 027306 010004 MOV R0,R4 ; DRIVE'S BLOCK ADDRESS
6794 027310 062704 000036 ADD $OPERC,R4 ; ADD THE STARTING ADDR OF SECTIONS TO CLEAR
6795 027314 012703 000010 MOV $B.,R3 ; NUMBER OF LOCNS TO BE CLEARED
6796 ; (ERROR COUNTERS NOT CLEARED)
6797 027320 005024 2$: CLR (R4)+ ; CLEAR THE LOCN
6798 027322 005303 DEC R3 ; DECREMENT THE LOCATION COUNTER
6799 027324 001375 BNE 2$ ; BR IF MORE TO GO
6800 027326 012604 MOV (SP)+,R4 ; RESTORE R4
6801 027330 012603 MOV (SP)+,R3 ; RESTORE R3
6802 027332 005260 000070 INC $PASSC(R0) ; INCREMENT THE PASS COUNT
6803 027336 000424 BR EOPX ; EXIT
6804 027340 104401 001201 3$: TYPE $SCRLF
6805 027344 005004 CLR R4 ; CLEAR R4 FOR DRIVE NUMBER
6806 027346 111004 MOV R4 (R0),R4 ; MOVE DRIVE NUMBER
6807 027350 146437 034606 001544 BICB ATABIT(R4),ASNLS ; DELETE DRIVE FROM ASSIGNED LIST
6808 027356 006304 ASL R4 ; MAKE DRIVE NUMBER INTO TABLE INDEX
6809 027360 010064 001546 MOV R0,DUNIT(R4) ; PUT BLOCK ADDRESS INTO DROP LIST
6810 027364 105737 001544 TSTB ASNLS ; ALL DRIVES ARE DESIGNED ?
6811 027370 001007 BNE EOPX ; BRANCH IF NOT
6812 027372 005237 001214 INC $DEVCT ; INCREMENT DEVICE COUNT
6813 027376 005237 001212 INC $PASS ; INCREMENT THE PASS NUMBER FOR APT
6814 027402 042737 100000 001212 BIC $100000,$PASS ; AVOID NEGATIVE NUMBER
6815 027410 000207 EOPX: RTS PC ; RETURN
6816 027412 005237 001212 INC $PASS ; INCREMENT THE PASS NUMBER
6817 027416 042737 100000 001212 BIC $100000,$PASS ; DON'T ALLOW A NEG. NUMBER
6818 027424 005327 DEC (PC)+ ; LOOP?
6819 027426 000001 $EOPCT: .WORD 1
6820 027430 003013 BGT $DOAGN ; YES
6821 027432 012737 MOV (PC)+,$(PC)+ ; RESTORE COUNTER
6822 027434 000001 $ENDCT: .WORD 1
6823 027436 027426 $EOPCT
6824 027440 013700 000042 $GET42: MOV $42,R0 ; GET MONITOR ADDRESS
6825 027444 001405 BEQ $DOAGN ; BRANCH IF NO MONITOR
6826 027446 000005 RESET ; CLEAR THE WORLD
6827 027450 004710 $ENDAD: JSR PC,(R0) ; GO TO MONITOR
6828 027452 000240 NOP ; SAVE ROOM
6829 027454 000240 NOP ; FOR
6830 027456 000240 NOP ; ACT11
6831 027460 $DOAGN:
6832 027460 000137 JMP $(PC)+ ; RETURN
6833 027462 027464 $RTNAD: .WORD FISK
6834 027464 005237 001210 FISK: INC $TESTN ; INCREMENT THE TEST NUMBER IN THE MAIL BOX
6835 027470 000137 005574 JMP MAIN1 ; RETURN TO LOOP
6836
6837 ; ROUTINE TO GET THE REMAINDER OF THE RANDOM NUMBER
6838 ; CALL
6839 : MOV NUMBER,R5 ; DIVISOR INTO R5
6840 : JSR PC,GETREM
6841 : ; REMAINDER IS IN R5
6842
6843 027474 013746 033752 GETREM: MOV $LONUM,-(SP) ; STORE RANDOM NUMBER ON THE STACK FOR DIVIDE

```

6844 027500 013746 033750
6845 027504 010546
6846 027506 004737 027522
6847 027512 012605
6848 027514 005726
6849 027516 000240
6850 027520 000207

MOV \$HINUM, -(SP) ; UPPER PART
MOV R5, -(SP) ; PUT THE DIVISOR ONTO THE STACK
JSR PC, LINKDV ; DIVIDE THE RANDOM NUMBERS
MOV (SP)+, R5 ; PUT THE REMAINDER INTO R5
TST (SP)+ ; ADJUST THE STACK POINTER
NOP ; FOR DEBUGGING HALT
RTS PC

; LINK ROUTINE TO THE DIVISION UTILITY SUBROUTINE
; THIS ROUTINE ALLOWS THE 'SYSMAC' DIVIDE ROUTINE
; CALLING SEQUENCE TO BE USED

6851
6852
6853
6854
6855
6856 027522 104412
6857 027524 016605 000026
6858 027530 005004
6859 027532 016602 000030
6860 027536 016603 000032
6861 027542 005000
6862 027544 005001
6863 027546 004737 027570
6864 027552 010166 000030
6865 027556 010366 000032
6866 027562 104413
6867 027564 012616
6868 027566 000207
6869

LINKDV: SAVREG ; STORE R0 - R5
MOV 26(SP), R5 ; DIVISOR
CLR R4 ; OTHER DIVISOR WORD
MOV 30(SP), R2 ; UPPER DIVIDEND WORD
MOV 32(SP), R3 ; LOWER DIVIDEND WORD
CLR R0 ; CLEAR OTHER DIVIDEND REGISTERS
CLR R1
JSR PC, M.DPID ; GO TO THE DIVIDE ROUTINE
MOV R1, 30(SP) ; REMAINDER ON THE STACK
MOV R3, 32(SP) ; QUOTIENT ON THE STACK
RESREG ; RESTORE R0 - R5
MOV (SP)+, (SP) ; MOVE RETURN UP THE STACK
RTS PC

; DIVISION UTILITY SUBROUTINE
; R0-R1-R2-R3=DIVIDEND
; R4-R5=DIVISOR
; R0-R1=REMAINDER AFTER DIVISION
; R2-R3=QUOTIENT AFTER DIVISION
; ENTER WITH JSR PC, M.DPID

6870
6871
6872
6873
6874
6875
6876
6877 027570 012746 000040
6878 027574 010446
6879 027576 010546
6880 027600 005466 000002
6881 027604 005416
6882 027606 005666 000002
6883 027612 061601
6884 027614 005500
6885 027616 066600 000002
6886 027622 103445
6887 027624 005046
6888 027626 006103
6889 027630 006102
6890 027632 006101
6891 027634 006100
6892 027636 005716
6893 027640 001410
6894 027642 005016
6895 027644 066601 000002
6896 027650 005500
6897 027652 005516
6898 027654 066600 000004
6899 027660 000404

M.DPID: MOV #40, -(SP) ; COUNTER FOR DIVISION CYCLES
MOV R4, -(SP) ; HIGH ORDER
MOV R5, -(SP) ; LOW ORDER DIVISOR TO THE STACK
NEG 2(SP) ; FORM NEGATIVE
NEG 2SP ; VERSION OF THE DIVISOR
SBC 2(SP)
ADD 2SP, R1
ADC R0 ; PERFORM THE INITIAL SUBTRACTION
ADD 2(SP), R0
BCS M.DP50 ; IF CARRY THEN OVERFLOW HAS OCCURRED
CLR -(SP) ; THIS IS A LONGER LASTING CARRY BIT
M.DP40: ROL R3
ROL R2
ROL R1
ROL R0
TST 2SP ; TEST "CARRY" INDICATOR
BEQ M.DP41 ; IF NO "CARRY" THEN ADD ELSE SUBTRACT
CLR 2SP ; CLEAR UP FOR NEXT TIME
ADD 2(SP), R1
ADC R0 ; ADD -(DIVISOR)
ADC 2SP ; SET "CARRY"
ADD 4(SP), R0 ;
BR M.DP42

6900	027662	060501		M.DP41:	ADD	R5,R1	
6901	027664	005500			ADC	R0	;ADD +(DIVISOR)
6902	027666	005516			QSP		;SET "CARRY"
6903	027670	060400			ADD	R4,R0	;I
6904	027672	005516		M.DP42:	ADC	QSP	;SET "CARRY"
6905	027674	005716			TST	QSP	;TEST THE UPDATE INDICATOR
6906	027676	001401			BEQ	+4	;IF ZERO FORGET IT
6907	027700	005203			INC	R3	;NO CARRY POSSIBLE HERE
6908	027702	005366	000006		DEC	6(SP)	;DECREMENT COUNTER
6909	027706	003347			BGT	M.DP40	;BRANCH IF MORE TO DO
6910	027710	006003			ROR	R3	
6911	027712	103404			BCS	M.DP44	
6912	027714	060501			ADD	R5,R1	
6913	027716	005500			ADC	R0	
6914	027720	060400			ADD	R4,R0	
6915	027722	000241			CLC		
6916	027724	006103		M.DP44:	ROL	R3	
6917	027726	062706	000010		ADD	#10,SP	;ADJUST STACK BY 4 WORDS
6918	027732	000242			CLV		
6919	027734	000207			RTS	PC	
6920	027736	062706	000006	M.DP50:	ADD	#6,SP	
6921	027742	000262			SEV		
6922	027744	000207			RTS	PC	
6923							
6924							
6925							
6926							
6927							
6928							
6929							
6930							
6931	027746	010046					
6932	027750	012746	000012				
6933	027754	162516					
6934	027756	016600	000006				
6935	027762	122710	000060				
6936	027766	001004					
6937	027770	112710	000040				
6938	027774	005200					
6939	027776	000771					
6940	030000	105710					
6941	030002	001003					
6942	030004	005300					
6943	030006	112710	000060				
6944	030012	016637	000006	030026			
6945	030020	062637	030026				
6946	030024	104401					
6947	030026	000000					
6948	030030	012600					
6949	030032	012616					
6950	030034	000205					
6951							
6952							
6953							
6954							
6955							


```

;ROUTINE TO REPLACE LEADING ZEROS IN A NUMERIC STRING WITH SPACES
;CALL
;      MOV      #ADR, -(SP)      ;ADDRESS OF NUMBER (IN ASCII)
;      JSR      RS, REPLZ
;      .WORD    N                ;'N' IS NUMBER OF DIGITS TO BE TYPED
REPLZ:  MOV      R0, -(SP)        ;SAVE R0
        MOV      #10, -(SP)      ;MAXIMUM NUMBER OF DIGITS TO BE TYPED
        SUB      (R5)+, (SP)     ;SUBTRACT DIGITS TO FORM INDEX
        MOV      6(SP), R0       ;ADDRESS OF NUMBER TO R0
1$:     C' B      #0, (R0)        ;BYTE EQUAL TO ASCII '0' ?
        BNE      2$              ;BR IF NOT
        MOVB     #40, (R0)       ;REPLACE THE ZERO WITH A SPACE
        INC      R0              ;INCREMENT THE BYTE ADDRESS
        BR       1$              ;GO BACK AND LOOK FOR MORE LEADING ZEROS
2$:     TSTB     (R0)             ;SEE IF ZERO BYTE TERMINATOR
        BNE      3$              ;BR IF NOT
        DEC      R0              ;BACKUP STRING POINTER
        MOVB     #0, (R0)        ;PUT A ZERO BACK IN
        MOV      6(SP), 4$       ;PUT ADDRESS IN LOCATION FOR TYPEOUT
        ADD      (SP)+, 4$       ;BEGINNING OF SIGNIFICANT DIGITS
        TYPE     .WORD          ;TYPE THE NUMBER
4$:     .WORD    0                ;ADDRESS OF NUMBER
        MOV      (SP)+, R0       ;RESTORE R0
        MOV      (SP)+, (SP)     ;MOVE RETURN ADDRESS
        RTS      RS              ;RETURN

;TYPE NUMERICAL ASCIZ STRING SUPPRESS LEADING ZEROS
;CALL
;      MOV      #NUMADR, -(SP)   ;FIRST ADDRESS OF ASCIZ STRING

```

6956						; JSR PC,\$SUPRS	
6957						\$SUPRS: MOV RO, -(SP)	; SAVE RO
6958	030036	010046				MOV 4(SP), RO	; PICKUP THE POINTER
6959	030040	016600	000004			1\$: TSTB (RO)	; TERMINATOR ?
6960	030044	105710				BEQ 2\$	; BR IF YES
6961	030046	001403				CMPB #'0, (RO)+	; IS THIS AN ASCII '0' ?
6962	030050	122720	000060			BEQ 1\$	; BR IF YES
6963	030054	001773				2\$: DEC RO	; BACKUP BY '1'
6964	030056	005300				MOV RO, 3\$	; SAVE FOR TYPING
6965	030060	010037	030066			DISPLY	; GO PRINT
6966	030064	104414				3\$: .WORD 0	; ASCII POINTER GOES HERE
6967	030066	000000				MOV (SP)+, RO	; RESTORE RO
6968	030070	012600				MOV (SP)+, (SP)	; RESTORE THE STACK
6969	030072	012616				RTS PC	; RETURN
6970	030074	000207					
6971							
6972						; ROUTINE TO TYPE AT PRIORITY 4	
6973							
6974	030076	013746	177776			TYPRI4: MOV 2\$PS, -(SP)	; SAVE THE PRESENT STATUS
6975	030102	012737	000200	177776		MOV 200, 2\$PS	; CHANGE THE PRIORITY TO 4
6976	030110	012537	030120			MOV (R5)+, 1\$	; MESSAGE ADDRESS
6977	030114	004737	031516			JSR PC, \$TYPE	; TYPE THE MESSAGE
6978	030120	000000				1\$: .WORD 0	; MESSAGE ADDRESS GOES HERE
6979	030122	000205				RTS R5	; RETURN
6980							
6981						; ROUTINE TO TYPE ERRORS	
6982						CALL	
6983						DISPLY	; MUST DEFINED IN 'TRAP' TABLE
6984						MESADR	; ADDRESS OF MESSAGE
6985						RETURN	
6986							
6987	030124	032777	020000	151022		\$DISPLY: BIT #BIT13, 2\$SWR	; INHIBIT ERROR TYPEOUT ?
6988	030132	001004				BNE 1\$	; BR IF YES
6989	030134	005037	177776			CLR 2\$PS	; SET PRIORITY TO ZERO
6990	030140	000137	031516			JMP \$TYPE	; TYPE THE MESSAGE
6991	030144	062716	000002			1\$: ADD #2, (SP)	; INCREMENT THE RETURN
6992	030150	000002				RTI	; RETURN
6993							
6994						; THIS ROUTINE IS USED TO CHECK IF AN	
6995						ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.	
6996						CALL	
6997						MOV #ADR, R1	; ADDRESS OF ASCII CHARACTER
6998						JSR R5, CK.OCT	; CHECK THE CHARACTER
6999						RETURN1	; CHARACTER IS NOT BETWEEN 0-7
7000						RETURN2	; CHARACTER IS IN R2 AS A
7001							; OCTAL DIGIT
7002							
7003	030152	121127	000060			CK.OCT: CMPB (R1), #'0	; LESS THAN ZERO?
7004	030156	103407				BLO 1\$	; YES -- BRANCH
7005	030160	121127	000067			CMPB (R1), #'7	; GREATER THAN SEVEN?
7006	030164	101004				BHI 1\$	; YES -- BRANCH
7007	030166	111102				MOVB (R1), R2	; GET THE CHARACTER
7008	030170	042702	177770			BIC #C7, R2	; STRIP AWAY THE ASCII
7009	030174	005725				TST (R5)+	; ADJUST FOR RETURN
7010	030176	000205				1\$: RTS R5	; RETURN
7011							

B11

7012
7013
7014
7015
7016
7017
7018
7019
7020
7021 030200 121127 000060
7022 030204 103407
7023 030206 121127 000071
7024 030212 101004
7025 030214 111102
7026 030216 042702 000060
7027 030222 005725
7028 030224 006205
7029
7030
7031
7032
7033
7034
7035
7036
7037
7038
7039
7040
7041
7042
7043 030226 105711
7044 030230 001417
7045 030232 121127 000054
7046 030236 001413
7047 030240 121127 000056
7048 030244 001407
7049 030246 004537 030200
7050 030252 000410
7051 030254 004537 030152
7052 030260 005725
7053 030262 005725
7054 030264 005725
7055 030266 005725
7056 030270 005725
7057 030272 005201
7058 030274 011505
7059 030276 000205
7060
7061
7062
7063
7064
7065
7066
7067

```

; THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER
; AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.
CALL
      MOV      #ADR,R1      ; ADDRESS OF ASCII CHARACTER
      JSR      R5,CK.DEC    ; CHECK THE CHARACTER
      RETURN1   ; NOT BETWEEN 0 AND 9
      RETURN2   ; BETWEEN 0 AND 9
      ; R2 = DIGIT

CK.DEC: CMPB    (R1),#'0    ; LESS THAN ZERO?
      BLO      1$           ; YES -- BRANCH
      CMPB    (R1),#'9    ; GREATER THAN NINE?
      BHI      1$           ; YES -- BRANCH
      MOVB    (R1),R2      ; GET THE CHARACTER
      BIC     #'0,R2       ; STRIP AWAY THE ASCII
      TST     (R5)+        ; ADJUST FOR RETURN
1$:    RTS      R5          ; RETURN

; THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO
; DETERMINE WHAT IT IS.
CALL
      MOV      #ADR,R1      ; ADDRESS OF ASCII CHARACTER
      JSR      R5,CK.CHR    ; CHECK CHARACTER
      RETURN   ADR1         ; UNKNOWN CHARACTER
      RETURN   ADR2         ; CARRIAGE RETURN * (R1)=ADR+1
      RETURN   ADR3         ; COMMA * (R1)=ADR+1
      RETURN   ADR4         ; PERIOD * (R1)=ADR+1
      RETURN   ADR5         ; DIGIT BETWEEN 0 AND 7.
      RETURN   ADR6         ; DIGIT BETWEEN 8 AND 9.
      ; R2 = DIGIT * (R1)=ADR+1

CK.CHR: TSTB    (R1)        ; "CARRIAGE RETURN"?
      BEQ      3$           ; YES -- BRANCH
      CMPB    (R1),#',     ; "COMMA"?
      BEQ      2$           ; YES -- BRANCH
      CMPB    (R1),#'.     ; "PERIOD"?
      BEQ      1$           ; YES -- BRANCH
      JSR      R5,CK.DEC    ; "DIGIT"?
      BFA      4$           ; NO -- BRANCH
      JSR      R5,CK.OCT    ; OCTAL ?
      TST     (R5)+        ; DIGIT BETWEEN 8-9
      TST     (R5)+        ; DIGIT BETWEEN 0-7
1$:    TST     (R5)+        ; PERIOD
2$:    TST     (R5)+        ; COMMA
3$:    TST     (R5)+        ; CARRIAGE RETURN
      INC     R1           ; MOVE POINTER TO NEXT CHARACTER
4$:    MOV     (R5),R5      ; UNKNOWN CHARACTER
      RTS      R5          ; RETURN

; THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL
; CHARACTERS AND FORMS A DECIMAL VALUE BINARY NUMBER IN R2.
CALL
      MOV      #ADR,R1      ; ADDRESS OF ASCII STRING
      MOV      #NUM,R2      ; MAX. MAGNITUDE OF INPUT NUMBER
      JSR      R5,CK.DIG    ; CHECK DIGITS
      RETURN   ADR1         ; "CR" ONLY ENTERED -- R2=0

```

C11

C11

7068			:	RETURN	ADR2	:	"PERIOD" ONLY ENTERED -- R2=0
7069			:	RETURN	ADR3	:	ILLEGAL CHARACTER OR INPUT TOO LARGE -- R2=?
7070			:	RETURN	ADR4	:	"CR" -- R2 = NUMBER
7071			:	RETURN	ADR5	:	"COMMA" -- R2 = NUMBER
7072			:	RETURN	ADR6	:	"PERIOD" -- R2 = NUMBER
7073							
7074	030300	010446	CK.DIG:	MOV	R4, -(SP)	:	SAVE R4
7075	030302	010346		MOV	R3, -(SP)	:	SAVE R3
7076	030304	010246		MOV	R2, -(SP)	:	SAVE THE MAX. SIZE ON THE STACK
7077	030306	005002		CLR	R2	:	START WITH 0
7078	030310	005003		CLR	R3		
7079	030312	005004		CLR	R4		
7080	030314	004537	030226	JSR	R5, CK.CHR	:	CHECK ONE CHARACTER
7081	030320	030414		6\$		:	ILLEGAL CHARACTER
7082	030322	030422		9\$		:	CARRIAGE RETURN
7083	030324	030414		6\$		:	"
7084	030326	030416		7\$		:	"
7085	030330	030334		1\$		:	DIGIT 0-7
7086	030332	030334		1\$		:	DIGIT 8-9
7087	030334	062705	000004	1\$:	ADD	#4, R5	STEP RETURN POINTER PAST "CR" & "PERIOD" RETURNS
7088	030340	006303		2\$:	ASL	R3	INPUT NUMBER *2
7089	030342	010346			MOV	R3, -(SP)	SAVE *2
7090	030344	006303			ASL	R3	*4
7091	030346	006303			ASL	R3	*8
7092	030350	062603			ADD	(SP)+, R3	(*2)+(*8) = *10
7093	030352	060203			ADD	R2, R3	UPDATE THE INPUT NUMBER
7094	030354	004537	030226		JSR	R5, CK.CHR	CHECK ONE CHARACTER
7095	030360	030420			8\$		ILLEGAL CHARACTER
7096	030362	030404			5\$		CARRIAGE RETURN
7097	030364	030402			4\$		"
7098	030366	030374			3\$		"
7099	030370	030340			2\$		DIGIT 0-7
7100	030372	030340			2\$		DIGIT 8-9
7101	030374	105711		3\$:	TSTB	(R1)	DOES A "CR" FOLLOW THE "PERIOD"
7102	030376	001010			BNE	8\$	BR IF NOT
7103	030400	005724			TST	(R4)+	INCREMENT THE RETURN
7104	030402	005724		4\$:	TST	(R4)+	INCREMENT THE RETURN
7105	030404	005724		5\$:	TST	(R4)+	INCREMENT THE RETURN
7106	030406	020316			CMP	R3, (SP)	CHECK THE MAGNITUDE OF THE NUMBER
7107	030410	101004			BHI	9\$	BR IF ENTERED NUMBER TOO LARGE
7108	030412	000402			BR	8\$	BYPASS INCREMENT
7109	030414	005725		6\$:	TST	(R5)+	INCREMENT RETURN PAST INVALID RETURN
7110	030416	005725		7\$:	TST	(R5)+	INCREMENT RETURN
7111	030420	060405		8\$:	ADD	R4, R5	SETUP RETURN POINTER
7112	030422	010302		9\$:	MOV	R3, R2	ENTERED VALUE
7113	030424	005726			TST	(SP)+	CLEAN MAX. SIZE OFF OF STACK
7114	030426	012603			MOV	(SP)+, R3	RESTORE R3
7115	030430	012604			MOV	(SP)+, R4	RESTORE R4
7116	030432	011505			MOV	(R5), R5	GET RETURN ADDRESS
7117	030434	000205			RTS	R5	RETURN
7118							
7119							
7120							
7121							
7122							
7123							

THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
UNSIGNED DECIMAL ASCII NUMBER.

:	CALL						
:	MOV	NUMBER, -(SP)	:	PUT THE NUMBER ON THE STACK			
:	JSR	PC, \$SB2D	:	CALL			

```

7124      ; RETURN ; ADDRESS OF THE 1ST ASCIZ CHAR IS ON THE STACK
7125
7126      ; NOTE: THE PROGRAM REQUIRES THIS FORM OF '$SB20', NOT THE VERSION ON
7127      ; THE SYSMAC LIBRARY, REV C AND LATER
7128
7129      030436 016637 000002 030462 $SB20: MOV 2(SP),1$ ;SAVE THE BINARY NUMBER
7130      030444 012746 030462      MOV 1$,-(SP) ;SET THE POINTER
7131      030450 004737 034050      JSR PC,$DB20 ;CALL THE DOUBLE LENGTH CONVERT
7132      030454 012666 000002      MOV (SP)+,2(SP) ;PICKUP THE POINTER
7133      030460 000207      RTS PC ;RETURN
7134      030462 000000 000000      1$: .WORD 0,0
7135
7136      ; THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
7137      ; UNSIGNED OCTAL ASCIZ NUMBER.
7138      ; CALL
7139      ; MOV NUMBER,-(SP) ;PUT THE NUMBER ON THE STACK
7140      ; JSR PC,$SB20 ;CALL
7141      ; RETURN ; ADDRESS OF THE 1ST ASCIZ CHAR IS ON THE STACK
7142
7143      ; NOTE: THE PROGRAM REQUIRES THIS FORM OF '$SB20', NOT THE VERSION ON
7144      ; THE SYSMAC LIBRARY, REV C AND LATER
7145
7146      030466 016637 000002 030512 $SB20: MOV 2(SP),1$ ;SAVE THE BINARY NUMBER
7147      030474 012746 030512      MOV 1$,-(SP) ;SET THE POINTER
7148      030500 004737 034244      JSR PC,$DB20 ;CALL THE DOUBLE LENGTH CONVERT
7149      030504 012666 000002      MOV (SP)+,2(SP) ;PICKUP THE POINTER
7150      030510 000207      RTS PC ;RETURN
7151      030512 000000 000000      1$: .WORD 0,0
7152
7153      ; KEYBOARD INTERRUPT INITIALIZATION ROUTINE
7154      ; CALL
7155      ; JSR PC,$TKINT
7156      ; RETURN
7157
7158      030516 012737 030546 000060 $TKINT: MOV $TKSRV,TKVEC ;SETUP VECTOR
7159      030524 012737 000100 000062      MOV $100,TKVEC+2 ;PRIORITY TO 4
7160      030532 005777 150424      TST $TKB ;CLEAR THE BUFFER
7161      030536 012777 000100 150414      MOV $BIT06,$TKS ;SET INTERRUPT ENABLE
7162      030544 000207      RTS PC ;RETURN
7163
7164      ; KEYBOARD INTERRUPT SERVICE ROUTINE
7165      ; CALL
7166      ; ENTER VIA INTERRUPT
7167
7168      030546 104410      $TKSRV: RDCHR ;READ THE KEYBOARD
7169      030550 112637 030676      MOVB (SP)+,5$ ;GET THE CHARACTER
7170      030554 023727 030676 000003      CMP 5$,#3 ;'CONTROL C' ?
7171      030562 001012      BNE 1$ ;BR IF NOT
7172      030564 104401 001201      TYPE $,SCRLF ;CR-LF
7173      030570 104401 031170      TYPE $,SCNTLC ;'C'
7174      030574 012737 177777 001360      MOV 1-1,CFLAG ;SET THE 'CONTROL C' FLAG
7175      030602 005077 150352      CLR $TKS ;CLEAR THE TTY INTERRUPT
7176      030606 000432      BR 4$ ;EXIT
7177      030610 023727 001154 000176      1$: CMP SWR,$SWREG ;SOFTWARE SWITCH REGISTER IN USE ?
7178      030616 001024      BNE 3$ ;BR IF NOT
7179      030620 023727 030676 000007      CMP 5$,#7 ;'CONTROL G' ?

```


7180	030626	001020			BNE	3\$		;BR IF NOT
7181	030630	104401	001201		TYPE	,SCRLF		;CR-LF
7182	030634	104401	033623		TYPE	,SCNTLG		; 'IG'
7183	030640	013746	177776		MOV	PS, -(SP)		;PUT THE STATUS WORD ON THE STACK
7184	030644	012746	030660		MOV	#2\$, -(SP)		;RETURN ADDRESS
7185	030650	005077	150304		CLR	2\$TKS		;CLEAR THE TTY INTERRUPT ENABLE
7186	030654	000137	033264		JMP	\$GTSWR		;GET THE SWITCH REGISTER ENTRY
7187	030660	012777	000100	150272	MOV	#100, 2\$TKS		;ENABLE TTY KEYBOARD INTERRUPT
7188	030666	000402			BR	4\$		;EXIT
7189	030670	104401	030676		3\$: TYPE	,5\$		;ECHO THE CHARACTER
7190	030674	000002			4\$: RTI			;RETURN
7191								
7192	030676	000000			5\$: .WORD	0		;ENTERED CHARACTER
7193								
7194								;THIS ROUTINE WILL INPUT A STRING FROM THE TTY
7195					CALL:			
7196								
7197					RDLIN			;INPUT A STRING FROM THE TTY
7198					RETURN HERE			;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
7199								;TERMINATOR WILL BE A BYTE OF ALL 0'S
7200	030700	010346			\$RDLIN: MOV	R3, -(SP)		;SAVE R3
7201	030702	005046			CLR	-(SP)		;CLEAR THE RUBOUT KEY
7202	030704	012703	031156		1\$: MOV	#STTYIN, R3		;GET ADDRESS
7203	030710	022703	031170		2\$: CMP	#STTYIN+10., R3		;BUFFER FULL?
7204	030714	101467				4\$		;BR IF YES
7205	030716	104410			BLOS			;GO READ ONE CHARACTER FROM THE TTY
7206	030720	112613			RDCHR			;GET CHARACTER
7207	030722	122713	000177		MOVB	(SP)+, (R3)		;IS IT A RUBOUT
7208	030726	001022			CMPB	#177, (R3)		;BR IF NO
7209	030730	005716			BNE	5\$		;IS THIS THE FIRST RUBOUT?
7210	030732	001007			TST	(SP)		;BR IF NO
7211	030734	112737	000134	031154	BNE	6\$		;TYPE A BACK SLASH
7212	030742	104401	031154		MOVB	#'\, 9\$		
7213	030746	012716	177777		TYPE	, 9\$		
7214	030752	005303			MOV	#-1, (SP)		;SET THE RUBOUT KEY
7215	030754	020327	031156		6\$: DEC	R3		;BACKUP BY ONE
7216	030760	103445			CMP	R3, #STTYIN		;STACK EMPTY?
7217	030762	111337	031154		BLO	4\$		;BR IF YES
7218	030766	104401	031154		MOVB	(R3), 9\$		;SETUP TO TYPEOUT THE DELETED CHAR.
7219	030772	000746			TYPE	, 9\$		;GO TYPE
7220	030774	005716			BR	2\$		;GO READ ANOTHER CHAR.
7221	030776	001406			5\$: TST	(SP)		;RUBOUT KEY SET?
7222	031000	112737	000134	031154	BEQ	7\$		;BR IF NO
7223	031006	104401	031154		MOVB	#'\, 9\$		;TYPE A BACK SLASH
7224	031012	005016			TYPE	, 9\$		
7225	031014	122713	000025		CLR	(SP)		;CLEAR THE RUBOUT KEY
7226	031020	001003			7\$: CMPB	#25, (R3)		;IS CHARACTER A CTRL U?
7227	031022	104401	033616		BNE	10\$		;BR IF NO
7228	031026	000726			TYPE	, SCNTLU		;TYPE A CONTROL "U"
7229	031030	122713	000003		BR	1\$		;GO START OVER
7230	031034	001006			10\$: CMPB	#3, (R3)		;IS CHARACTER A CTRL C ?
7231	031036	012737	177777	001360	BNE	8\$		;BR IF NOT
7232	031044	104401	031170		MOV	#-1, CFLAG		;SET CNTRL C FLAG
7233	031050	000427			TYPE	, SCNTLC		;ECHO IT
7234	031052	122713	000012		BR	11\$		;EXIT
7235	031056	001011			8\$: CMPB	#12, (R3)		;IS CHARACTER A "LF"?
					BNE	3\$		;BRANCH IF NO

7236	031060	105013				CLRB	(R3)		; CLEAR THE CHARACTER
7237	031062	104401	001201			TYPE	,SCRLF		; TYPE A "CR" & "LF"
7238	031066	104401	031156			TYPE	,STTYIN		; TYPE THE INPUT STRING
7239	031072	000706				BR	2\$		; GO PICKUP ANOTHER CHARACTER
7240	031074	104401	001200		4\$:	TYPE	\$QUES		; TYPE A '?'
7241	031100	000701				BR	1\$		; CLEAR THE BUFFER AND LOOP
7242	031102	111337	031154		3\$:	MOVB	(R3),9\$		; ECHO THE CHARACTER
7243	031106	104401	031154			TYPE	9\$		
7244	031112	122723	000015			CMPB	#15,(R3)+		; CHECK FOR RETURN
7245	031116	001274				BNE	2\$		; LOOP IF NOT RETURN
7246	031120	105063	177777			CLRB	-1(R3)		; CLEAR RETURN (THE 15)
7247	031124	104401	001202			TYPE	\$LF		; TYPE A LINE FEED
7248	031130	005726			11\$:	TST	{SP}+		; CLEAN RUBOUT KEY FROM THE STACK
7249	031132	012603				MOV	(SP)+,R3		; RESTORE R3
7250	031134	011646				MOV	(SP)-,(SP)		; ADJUST THE STACK AND PUT ADDRESS OF THE
7251	031136	016666	000004	000002		MOV	4(SP),2(SP)		; FIRST ASCII CHARACTER ON IT
7252	031144	012766	031156	000004		MOV	#STTYIN,4(SP)		
7253	031152	000002				RTI			; RETURN
7254	031154	000			9\$:	.BYTE	0		; STORAGE FOR ASCII CHAR. TO TYPE
7255	031155	000				.BYTE	0		; TERMINATOR
7256	031156	000012			STTYIN:	.BLKB	10.		; RESERVE 10 BYTES FOR TTY INPUT
7257	031170	041536	005015	000	SCNTLC:	.ASCIZ	/C/<CR><LF>		; CONTROL "C"
7258									
7259		031176				.EVEN			
7260									
7261									
7262									
7263									
7264									
7265						.SBTTL	MACRO ROUTINES		
7266						.SBTTL	ERROR HANDLER ROUTINE		
7267									
7268									
7269									
7270									
7271									
7272									
7273									
7274									
7275									
7276									
7277									
7278									
7279									
7280	031176					\$ERROR:			
7281	031176	104407				CKSWR			; TEST FOR CHANGE IN SOFT-SWR
7282	031200	010337	001342			MOV	R3,ATTN		; SAVE THE ATTENTION REGISTER CONTENTS
7283	031204	010137	001216			MOV	R1,DRIVE		; DRIVE NUMBER
7284	031210	032777	020000	147736		BIT	#SW13,\$SWR		; INHIBIT PRINTOUTS ?
7285	031216	001002				BNE	+6		; BR IF YES
7286	031220	004737	023670			JSR	PC \$TIME		; TYPE THE TIME
7287	031224	105237	001117		7\$:	INCB	\$ERFLG		; SET THE ERROR FLAG
7288	031230	001775				BEQ	7\$		; DON'T LET THE FLAG GO TO ZERO
7289	031232	013777	001116	147716		MOV	\$STNM,\$DISPLAY		; DISPLAY TEST NUMBER AND ERROR FLAG
7290	031240	032777	002000	147706		BIT	#BIT10,\$SWR		; BELL ON ERROR?
7291	031246	001402				BEG	1\$		; NO - SKIP

```

7292 031250 104401 001174
7293 031254 005237 001126
7294 031260 011637 001132
7295 031264 162737 000002 001132
7296 031272 117737 147634 001130
7297 031300 032777 020000 147646
7298 031306 001004
7299 031310 004737 031362
7300 031314 104401 001201
7301 031320
7302 031320 122737 000001 001224
7303 031326 001007
7304 031330 113737 001130 031342
7305 031336 004737 032016
7306 031342 000
7307 031343 000
7308 031344 000777
7309 031346 005777 147602
7310 031352 100002
7311 031354 000000
7312 031356 104407
7313 031360
7314 031360 000002
7315
7316
7317
7318
7319
7320
7321
7322
7323 031362
7324 031362 104401 001201
7325 031366 010046
7326 031370 005000
7327 031372 153700 001130
7328 031376 001004
7329
7330 031400 013746 001132
7331
7332 031404 104402
7333 031406 000426
7334 031410 005300
7335 031412 006300
7336 031414 006300
7337 031416 006300
7338 031420 062700 003542
7339 031424 012037 031434
7340 031430 001404
7341 031432 104401
7342 031434 000000
7343 031436 104401 001201
7344 031442 012037 031452
7345 031446 001404
7346 031450 104401
7347 031452 000000

```

```

1$: TYPE $BELL ;;RING BELL
INC $ERTTL ;;COUNT THE NUMBER OF ERRORS
MOV (SP), $ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
SUB #2, $ERRPC
MOVB $ERRPC, $ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
BIT $BIT13, $SWR ;;SKIP TYPEOUT IF SET
BNE 20$ ;;SKIP TYPEOUTS
JSR PC, $ERRTYP ;;GO TO USER ERROR ROUTINE
TYPE $CRLF

20$: CMPB $APTENV, $ENV ;;RUNNING IN APT MODE
BNE 2$ ;;NO SKIP APT ERROR REPORT
MOVB $ITEMB, 21$ ;;SET ITEM NUMBER AS ERROR NUMBER
JSR PC, $ATY4 ;;REPORT FATAL ERROR TO APT

21$: .BYTE 0
.BYTE 0

22$: BR 22$ ;;APT ERROR LOOP
2$: TST $SWR ;;HALT ON ERROR
BPL 3$ ;;SKIP IF CONTINUE
HALT ;;HALT ON ERROR!
CKSWR ;;TEST FOR CHANGE IN SOFT-SWR

3$: RTI ;;RETURN

```

.SBTTL ERROR MESSAGE TYPEOUT ROUTINE

```

;*****
;THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
;ERROR IS TO BE REPORTED. IT THEN OBTAINS FROM THE "ERROR TABLE" ($ERRTB),
;AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.

```

```

$ERRTYP:
TYPE $CRLF ;; "CARRIAGE RETURN" & "LINE FEED"
MOV RO, -(SP) ;;SAVE RO
CLR RO ;;PICKUP THE ITEM INDEX
BISB $ITEMB, RO
BNE 1$ ;;IF ITEM NUMBER IS ZERO, JUST
TYPE THE PC OF THE ERROR
MOV $ERRPC, -(SP) ;;SAVE $ERRPC FOR TYPEOUT
ERROR ADDRESS
GO TYPE--OCTAL ASCII(ALL DIGITS)
GET OUT
ADJUST THE INDEX SO THAT IT WILL
WORK FOR THE ERROR TABLE

1$: BR 6$
DEC RO
ASL RO
ASL RO
ASL RO
ADD $ERRTB, RO ;;FORM TABLE POINTER
MOV (RO)+, 2$ ;;PICKUP "ERROR MESSAGE" POINTER
BEQ 3$ ;;SKIP TYPEOUT IF NO POINTER
TYPE THE "ERROR MESSAGE"
"ERROR MESSAGE" POINTER GOES HERE
" "CARRIAGE RETURN" & "LINE FEED"
PICKUP "DATA HEADER" POINTER
SKIP TYPEOUT IF 0
TYPE THE "DATA HEADER"
"DATA HEADER" POINTER GOES HERE

2$: .WORD 0
TYPE $CRLF

3$: MOV (RO)+, 4$
BEQ 5$

4$: .WORD 0
5$:

```

CZRM880 RM03/2 PERF EXER
CZRM88.P11 21-NOV-77 15:34

MACY11 30(1046) 22-NOV-77 10:06 PAGE 138
ERROR MESSAGE TYPEOUT ROUTINE

SEQ 0137

7348 031454 104401 001201
7349 031460 011000
7350 031462 001004
7351 031464 012600
7352 031466 104401 001201
7353 031472 000207
7354 031474
7355 031474 013046
7356 031476 104402
7357 031500 005710
7358 031502 001770
7359 031504 104401 031512
7360 031510 000771
7361 031512 020040 000
7362 031516
7363
7364
7365
7366
7367
7368
7369
7370
7371
7372
7373
7374
7375
7376
7377
7378
7379
7380
7381 031516 105737 001173
7382 031522 100002
7383 031524 000000
7384 031526 000430
7385 031530 010046
7386 031532 017600 000002
7387 031536 122737 000001 001224
7388 031544 001011
7389 031546 132737 000100 001225
7390 031554 001405
7391 031556 010037 031566
7392 031562 004737 032006
7393 031566 000000
7394 031570 132737 000040 001225
7395 031576 001003
7396 031600 112046
7397 031602 001005
7398 031604 005726
7399 031606 012600
7400 031610 062716 000002
7401 031614 000002
7402 031616 122716 000011
7403 031622 001430

```

      TYPE      ,SCRLF      ; "CARRIAGE RETURN" & "LINE FEED"
5$:  MOV      (RO),RO      ; PICKUP "DATA TABLE" POINTER
      BNE      7$          ; GO TYPE THE DATA
6$:  MOV      (SP)+,RO      ; RESTORE RO
      TYPE      ,SCRLF      ; "CARRIAGE RETURN" & "LINE FEED"
      RTS      PC          ; RETURN
7$:  MOV      2(RO)+,-(SP)  ; SAVE 2(RO)+ FOR TYPEOUT
      TYPOC      (RO)      ; GO TYPE--OCTAL ASCII(ALL DIGITS)
      TST      (RO)        ; IS THERE ANOTHER NUMBER?
      BEQ      6$          ; BR IF NO
      TYPE      ,8$        ; TYPE TWO(2) SPACES
      BR       7$          ; LOOP
8$:  .ASCIZ    / /         ; TWO(2) SPACES
      .EVEN

```

.SBTTL TYPE ROUTINE

```

;*****
;ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
;THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
;NOTE1:      $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
;NOTE2:      $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
;NOTE3:      $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
;
;CALL:
;1) USING A TRAP INSTRUCTION
;
;      TYPE      ,MESADR      ; MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
;OR
;      TYPE      MESADR
;

```

```

$TYPE:  TSTB    $TPFLG      ; IS THERE A TERMINAL?
        BPL     1$          ; BR IF YES
        HALT    3$          ; HALT HERE IF NO TERMINAL
        BR      3$          ; LEAVE
1$:  MOV      RO,-(SP)      ; SAVE RO
        MOV      22(SP),RO  ; GET ADDRESS OF ASCIZ STRING
        CMPB    #APTENV,$ENV ; RUNNING IN APT MODE
        BNE     62$        ; NO GO CHECK FOR APT CONSOLE
        BITB    #APTPOOL,$ENVM ; SPOOL MESSAGE TO APT
        BEQ     62$        ; NO GO CHECK FOR CONSOLE
        MOV      RO,61$     ; SETUP MESSAGE ADDRESS FOR APT
        JSR     PC,$ATY3    ; SPOOL MESSAGE TO APT
        .WORD   0          ; MESSAGE ADDRESS
62$:  BITB    #APTCSUP,$ENVM ; APT CONSOLE SUPPRESSED
        BNE     60$        ; YES, SKIP TYPE OUT
        MOVB    (RO)+,-(SP) ; PUSH CHARACTER TO BE TYPED ONTO STACK
        BNE     4$          ; BR IF IT ISN'T THE TERMINATOR
        TST     (SP)+      ; IF TERMINATOR POP IT OFF THE STACK
60$:  MOV      (SP)+,RO      ; RESTORE RO
3$:  ADD      #2,(SP)        ; ADJUST RETURN PC
        RTI
4$:  CMPB     #HT,(SP)      ; BRANCH IF 'HT'
        BEG     8$

```

```

7404 031624 122716 000200      CMPB    #CRLF,(SP)      ;;BRANCH IF NOT 'CRLF'
7405 031630 001006      BNE      5$
7406 031632 005726      TST      (SP)+      ;;POP 'CR'<LF' EQUIV
7407 031634 104401      TYPE      ;;TYPE A CR AND LF
7408 031636 001201      SCRLF
7409 031640 105037 031774      CLRB     $CHARCNT      ;;CLEAR CHARACTER COUNT
7410 031644 000755      BR      2$      ;;GET NEXT CHARACTER
7411 031646 004737 031730      5$:     JSR     PC,$TYPEC      ;;GO TYPE THIS CHARACTER
7412 031652 123726 001172      6$:     CMPB    $FILLC,(SP)+      ;;IS IT TIME FOR FILLER CHARS.?
7413 031656 001350      BNE      2$      ;;IF NO GO GET NEXT CHAR.
7414 031660 013746 001170      MOV      $NULL,-(SP)      ;;GET # OF FILLER CHARS. NEEDED
7415                                AND THE NULL CHAR.
7416 031664 105366 000001      7$:     DECB     1(SP)      ;;DOES A NULL NEED TO BE TYPED?
7417 031670 002770      BLT      6$      ;;BR IF NO--GO POP THE NULL OFF OF STACK
7418 031672 004737 031730      JSR     PC,$TYPEC      ;;GO TYPE A NULL
7419 031676 105337 031774      DECB     $CHARCNT      ;;DO NOT COUNT AS A COUNT
7420 031702 000770      BR      7$      ;;LOOP
7421
7422      ;HORIZONTAL TAB PROCESSOR
7423
7424 031704 112716 000040      8$:     MOVB     #'',(SP)      ;;REPLACE TAB WITH SPACE
7425 031710 004737 031730      9$:     JSR     PC,$TYPEC      ;;TYPE A SPACE
7426 031714 132737 000007 031774      BITB     #7,$CHARCNT      ;;BRANCH IF NOT AT
7427 031722 001372      BNE      9$      ;;TAB STOP
7428 031724 005726      TST      (SP)+      ;;POP SPACE OFF STACK
7429 031726 000724      BR      2$      ;;GET NEXT CHARACTER
7430 031730 105777 147230      $TYPEC:  TSTB     $STPS      ;;WAIT UNTIL PRINTER IS READY
7431 031734 100375      BPL      $TYPEC
7432 031736 116677 000002 147222      MOVB     2(SP),$STPB      ;;LOAD CHAR TO BE TYPED INTO DATA REG.
7433 031744 122766 000015 000002      CMPB     #CR,2(SP)      ;;IS CHARACTER A CARRIAGE RETURN?
7434 031752 001003      BNE      1$      ;;BRANCH IF NO
7435 031754 105037 031774      CLRB     $CHARCNT      ;;YES--CLEAR CHARACTER COUNT
7436 031760 000406      BR      $TYPEX      ;;EXIT
7437 031762 122766 000012 000002      1$:     CMPB     #LF,2(SP)      ;;IS CHARACTER A LINE FEED?
7438 031770 001402      BEQ      $TYPEX      ;;BRANCH IF YES
7439 031772 105227      INCB     (PC)+      ;;COUNT THE CHARACTER
7440 031774 000000      $CHARCNT: .WORD 0      ;;CHARACTER COUNT STORAGE
7441 031776 000207      $TYPEX:  RTS      PC
7442
7443      .SBTTL  APT COMMUNICATIONS ROUTINE
7444
7445      ;*****
7446 032000 112737 000001 032244      $ATY1:  MOVB     #1,$FFLG      ;;TO REPORT FATAL ERROR
7447 032006 112737 000001 032242      $ATY3:  MOVB     #1,$MFLG      ;;TO TYPE A MESSAGE
7448 032014 000403      BR      $ATYC
7449 032016 112737 000001 032244      $ATY4:  MOVB     #1,$FFLG      ;;TO ONLY REPORT FATAL ERROR
7450 032024      $ATYC:
7451 032024 010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
7452 032026 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
7453 032030 105737 032242      TSTB     $MFLG      ;;SHOULD TYPE A MESSAGE?
7454 032034 001450      BEQ      5$      ;;IF NOT: BR
7455 032036 122737 000001 001224      CMPB     #APTENV,$ENV      ;;OPERATING UNDER APT?
7456 032044 001031      BNE      3$      ;;IF NOT: BR
7457 032046 132737 000100 001225      BITB     #APTSPOOL,$ENVM      ;;SHOULD SPOOL MESSAGES?
7458 032054 001425      BEQ      3$      ;;IF NOT: BR

```

```

7460 032056 017600 000004      MOV      24(SP),RO      ;;GET MESSAGE ADDR.
7461 032062 062766 000002      ADD      #2,4(SP)      ;;BUMP RETURN ADDR.
7462 032070 005737 001204      1$:      TST      $MSGTYPE      ;;SEE IF DONE W/ LAST XMISSION?
7463 032074 001375              BNE      1$              ;;IF NOT: WAIT
7464 032076 010037 001220      MOV      RO,$MSGAD      ;;PUT ADDR IN MAILBOX
7465 032102 105720              2$:      TSTB     (RO)+      ;;FIND END OF MESSAGE
7466 032104 001376              BNE      2$              ;;
7467 032106 163700 001220      SUB      $MSGAD,RO      ;;SUB START OF MESSAGE
7468 032112 006200              ASR      RO              ;;GET MESSAGE LGTH IN WORDS
7469 032114 010037 001222      MOV      RO,$MSGGLT      ;;PUT LENGTH IN MAILBOX
7470 032120 012737 000004      MOV      #4,$MSGTYPE      ;;TELL APT TO TAKE MSG.
7471 032126 000413              BR       5$              ;;
7472 032130 017637 000004      3$:      MOV      24(SP),4$      ;;PUT MSG ADDR IN JSR LINKAGE
7473 032136 062766 000002      ADD      #2,4(SP)      ;;BUMP RETURN ADDRESS
7474 032144 013746 177776      MOV      177776,-(SP)      ;;PUSH 177776 ON STACK
7475 032150 004737 001516      JSR      PC,$TYPE      ;;CALL TYPE MACRO
7476 032154 000000              4$:      .WORD     0              ;;
7477 032156              5$:              ;;
7478 032156 105737 032244      10$:     TSTB     $FFLG      ;;SHOULD REPORT FATAL ERROR?
7479 032162 001416              BEQ      12$      ;;IF NOT: BR
7480 032164 005737 001224      TST      $ENV      ;;RUNNING UNDER APT?
7481 032170 001413              BEQ      12$      ;;IF NOT: BR
7482 032172 005737 001204      11$:     TST      $MSGTYPE      ;;FINISHED LAST MESSAGE?
7483 032176 001375              BNE      11$      ;;IF NOT: WAIT
7484 032200 017637 000004      MOV      24(SP),$FATAL      ;;GET ERROR #
7485 032206 062766 000002      ADD      #2,4(SP)      ;;BUMP RETURN ADDR.
7486 032214 005237 001204      INC      $MSGTYPE      ;;TELL APT TO TAKE ERROR
7487 032220 105037 032244      12$:     CLRB     $FFLG      ;;CLEAR FATAL FLAG
7488 032224 105037 032243      CLRB     $LFLG      ;;CLEAR LOG FLAG
7489 032230 105037 032242      CLRB     $MFLG      ;;CLEAR MESSAGE FLAG
7490 032234 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
7491 032236 012600      MOV      (SP)+,RO      ;;POP STACK INTO RO
7492 032240 000207      RTS      PC      ;;RETURN
7493 032242 000      $MFLG: .BYTE     0      ;;MESSG. FLAG
7494 032243 000      $LFLG: .BYTE     0      ;;LOG FLAG
7495 032244 000      $FFLG: .BYTE     0      ;;FATAL FLAG
7496 032246 000      .EVEN
7497 000200      APTSIZE=200
7498 000001      APTENV=001
7499 000100      APTSP00L=100
7500 000040      APTCSUP=040
7501
7502      .SBTTL  POWER DOWN AND UP ROUTINES
7503
7504      ;*****
7505      ;POWER DOWN ROUTINE
7506 032246 012737 032412 000024 $PWDRN: MOV      $SILLUP,$PWAVEC      ;;SET FOR FAST UP
7507 032254 012737 000340 000026      MOV      #340,$PWAVEC+2      ;;PRIO:7
7508 032262 010046      MOV      RO,-(SP)      ;;PUSH RO ON STACK
7509 032264 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
7510 032266 010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
7511 032270 010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
7512 032272 010446      MOV      R4,-(SP)      ;;PUSH R4 ON STACK
7513 032274 010546      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
7514 032276 017746 146652      MOV      2$WR,-(SP)      ;;PUSH 2$WR ON STACK
7515 032302 010637 032416      MOV      SP,$SAVR6      ;;SAVE SP

```



```

7516 032306 012737 032320 000024      MOV      #SPWRUP, @PWRVEC ;; SET UP VECTOR
7517 032314 000000                      HALT
7518 032316 000776                      BR      .-2                      ;; HANG UP
7519
7520      ;*****
7521      ; POWER UP ROUTINE
7522 032320 012737 032412 000024 $PWRUP: MOV      $SILLUP, @PWRVEC ;; SET FOR FAST DOWN
7523 032326 013706 032416          MOV      $SAVR6, SP      ;; GET SP
7524 032332 005037 032416          CLR      $SAVR6          ;; WAIT LOOP FOR THE TTY
7525 032336 005237 032416 1$:      INC      $SAVR6          ;; WAIT FOR THE INC
7526 032342 001375          BNE      1$                      ;; OF WORD
7527 032344 012677 146604          MOV      (SP)+, @SWR      ;; POP STACK INTO @SWR
7528 032350 012605          MOV      (SP)+, R5          ;; POP STACK INTO R5
7529 032352 012604          MOV      (SP)+, R4          ;; POP STACK INTO R4
7530 032354 012603          MOV      (SP)+, R3          ;; POP STACK INTO R3
7531 032356 012602          MOV      (SP)+, R2          ;; POP STACK INTO R2
7532 032360 012601          MOV      (SP)+, R1          ;; POP STACK INTO R1
7533 032362 012600          MOV      (SP)+, R0          ;; POP STACK INTO R0
7534 032364 012737 032246 000024      MOV      #SPWRDN, @PWRVEC ;; SET UP THE POWER DOWN VECTOR
7535 032372 012737 000340 000026      MOV      #340, @PWRVEC+2 ;; PRIO:7
7536 032400 104401          TYPE                      ;; REPORT THE POWER FAILURE
7537 032402 032420          $PWRMG: .WORD      $POWER      ;; POWER FAIL MESSAGE POINTER
7538 032404 012716          MOV      (PC)+, (SP)          ;; RESTART AT SATPOW
7539 032406 032430          $PWRAD: .WORD      SATPOW      ;; RESTART ADDRESS
7540 032410 000002          RTI
7541 032412 000000          $SILLUP: HALT
7542 032414 000776          BR      .-2                      ;; THE POWER UP SEQUENCE WAS STARTED
7543 032416 000000          $SAVR6: 0                      ;; BEFORE THE POWER DOWN WAS COMPLETE
7544 032420 005015 047520 042527      $POWER: .ASCIZ  <15><12>"POWER" ;; PUT THE SP HERE
7545 032426 000122
7546
7547
7548
7549
7550
7551 032430 000005          SATPOW: RESET
7552 032432 005037 177776          CLR      @PS
7553 032436 005037 001364          CLR      HOUR
7554 032442 005037 001366          CLR      MINUTE
7555 032446 005037 001370          CLR      SECOND
7556 032452 005037 001470          CLR      INTRVL+2
7557 032456 004737 022632          JSR      PC, CKCLK
7558 032462 022737 000005 001470 1$:  CMP      #5, INTRVL+2
7559 032470 101374          BHI      1$
7560 032472 005037 001470          CLR      INTRVL+2
7561 032476 005037 001370          CLR      SECOND
7562 032502 005037 001366          CLR      MINUTE
7563 032506 012737 000400 001356          MOV      #400, CHGADR
7564 032514 012705 001522          MOV      #ORDERQ, R5
7565 032520 005025          2$:      CLR      (R5)+
7566 032522 022705 002102          CMP      #BLKADR, R5
7567 032526 001374          BNE      2$
7568 032530 012737 007070 001264          MOV      #7070, $CDW1
7569 032536 000137 004410          JMP      $IZMEM
7570
7571

```

```

; POW UP ROUTINE WAIT
; FIVE MINUTS, THEN AUTO STARTS AT SETVEC

```

```

; CLEAR THE BUS
; CLEAR PSW
; RESET THE HOUR COUNT
; RESET THE MINUTS COUNT
; RESET THE SECOND COUNT
; RESET THE INTERVAL COUNT
; CHECK THE CLOCK
; FIVE MINUTES YET ?
; WAIT IF NOT
; RESET INTERVAL COUNT
; RESET TIMER
; FORGE THE AUTO START
; CLEAR UP THE QUEUE AND BUFFER
; ALL DONE ?
; BRANCH IF NOT
; FLAG FROM POWER FAIL
; LOOP BACK

```

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPOS      ;;CALL FOR TYPEOUT
*      .BYTE      N      ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
*      .BYTE      M      ;;M=1 OR 0
*                               ;;1=TYPE LEADING ZEROS
*                               ;;0=SUPPRESS LEADING ZEROS
*$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*$TYPOS OR $TYPOC
*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPON      ;;CALL FOR TYPEOUT
*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPOC      ;;CALL FOR TYPEOUT
*$TYPOS: MOV      2(SP),-(SP)      ;;PICKUP THE MODE
*      MOVB      1(SP),SOFILL      ;;LOAD ZERO FILL SWITCH
*      MOVB      (SP)+,SOMODE+1      ;;NUMBER OF DIGITS TO TYPE
*      ADD      #2,(SP)      ;;ADJUST RETURN ADDRESS
*      BR      $TYPON
*$TYPOC: MOVB      #1,SOFILL      ;;SET THE ZERO FILL SWITCH
*      MOVB      #6,SOMODE+1      ;;SET FOR SIX(6) DIGITS
*$TYPON: MOVB      #5,SOCNT      ;;SET THE ITERATION COUNT
*      MOV      R3,-(SP)      ;;SAVE R3
*      MOV      R4,-(SP)      ;;SAVE R4
*      MOV      R5,-(SP)      ;;SAVE R5
*      MOVB      SOMODE+1,R4      ;;GET THE NUMBER OF DIGITS TO TYPE
*      NEG      R4      ;;SUBTRACT IT FOR MAX. ALLOWED
*      ADD      #6,R4      ;;SAVE IT FOR USE
*      MOVB      R4,SOMODE      ;;GET THE ZERO FILL SWITCH
*      MOVB      SOFILL,R4      ;;PICKUP THE INPUT NUMBER
*      MOV      12(SP),R5      ;;CLEAR THE OUTPUT WORD
*      CLR      R3      ;;ROTATE MSB INTO "C"
*      1$: ROL      R5      ;;GO DO MSB
*      BR      3$      ;;FORM THIS DIGIT
*      2$: ROL      R5
*      ROL      R5
*      ROL      R5
*      MOV      R5,R3
*      3$: ROL      R3      ;;GET LSB OF THIS DIGIT
*      DECB      SOMODE      ;;TYPE THIS DIGIT?
*      BPL      7$      ;;BR IF NO
*      BIC      #177770,R3      ;;GET RID OF JUNK
*      BNE      4$      ;;TEST FOR 0
*      TST      R4      ;;SUPPRESS THIS 0?
*      BEQ      5$      ;;BR IF YES

```

```

7572
7573
7574
7575
7576
7577
7578
7579
7580
7581
7582
7583
7584
7585
7586
7587
7588
7589
7590
7591
7592
7593
7594
7595
7596
7597 032542 017646 000000
7598 032546 116637 000001 032765
7599 032554 112637 032767
7600 032560 062716 000002
7601 032564 000406
7602 032566 112737 000001 032765
7603 032574 112737 000006 032767
7604 032602 112737 000005 032764
7605 032610 010346
7606 032612 010446
7607 032614 010546
7608 032616 113704 032767
7609 032622 005404
7610 032624 062704 000006
7611 032630 110437 032766
7612 032634 113704 032765
7613 032640 016605 000012
7614 032644 005003
7615 032646 006105
7616 032650 000404
7617 032652 006105
7618 032654 006105
7619 032656 006105
7620 032660 010503
7621 032662 006103
7622 032664 105337 032766
7623 032670 100016
7624 032672 042703 177770
7625 032676 001002
7626 032700 005704
7627 032702 001403

```



```

7628 032704 005204      4$: INC      R4      ;; DON'T SUPPRESS ANYMORE 0'S
7629 032706 052703 000060      BIS      #'0,R3      ;; MAKE THIS DIGIT ASCII
7630 032712 052703 000040      5$: BIS      #' ,R3      ;; MAKE ASCII IF NOT ALREADY
7631 032716 110337 032762      MOV      R3,R5      ;; SAVE FOR TYPING
7632 032722 104401 032762      TYPE     8$      ;; GO TYPE THIS DIGIT
7633 032726 105337 032764      7$: DECB     $OCNT      ;; COUNT BY 1
7634 032732 003347      BGT      2$      ;; BR IF MORE TO DO
7635 032734 002402      BLT      6$      ;; BR IF DONE
7636 032736 005204      INC      R4      ;; INSURE LAST DIGIT ISN'T A BLANK
7637 032740 000744      BR       2$      ;; GO DO THE LAST DIGIT
7638 032742 012605      6$: MOV      (SP)+,R5      ;; RESTORE R5
7639 032744 012604      MOV      (SP)+,R4      ;; RESTORE R4
7640 032746 012603      MOV      (SP)+,R3      ;; RESTORE R3
7641 032750 016666 000002 000004      MOV      2(SP),4(SP)      ;; SET THE STACK FOR RETURNING
7642 032756 012616      MOV      (SP)+,(SP)
7643 032760 000002      RTI      ;; RETURN
7644 032762 000      8$: .BYTE     0      ;; STORAGE FOR ASCII DIGIT
7645 032763 000      .BYTE     0      ;; TERMINATOR FOR TYPE ROUTINE
7646 032764 000      $OCNT: .BYTE    0      ;; OCTAL DIGIT COUNTER
7647 032765 000      $OFILL: .BYTE   0      ;; ZERO FILL SWITCH
7648 032766 000000      $OMODE: .WORD   0      ;; NUMBER OF DIGITS TO TYPE
7649
7650      .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
7651
7652      ;;*****
7653      ;;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
7654      ;;SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
7655      ;;NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
7656      ;;BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
7657      ;;REPLACED WITH SPACES.
7658      ;;CALL:
7659      ;;      MOV      NUM,-(SP)      ;; PUT THE BINARY NUMBER ON THE STACK
7660      ;;      TYPDS      ;; GO TO THE ROUTINE
7661
7662      $TYPDS:
7663      MOV      R0,-(SP)      ;; PUSH R0 ON STACK
7664      MOV      R1,-(SP)      ;; PUSH R1 ON STACK
7665      MOV      R2,-(SP)      ;; PUSH R2 ON STACK
7666      MOV      R3,-(SP)      ;; PUSH R3 ON STACK
7667      MOV      R5,-(SP)      ;; PUSH R5 ON STACK
7668      MOV      #20200,-(SP)      ;; SET BLANK SWITCH AND SIGN
7669      MOV      20(SP),R5      ;; GET THE INPUT NUMBER
7670      BPL      1$      ;; BR IF INPUT IS POS.
7671      NEG      R5      ;; MAKE THE BINARY NUMBER POS.
7672      MOV      #'-,1(SP)      ;; MAKE THE ASCII NUMBER NEG.
7673      CLR      R0      ;; ZERO THE CONSTANTS INDEX
7674      MOV      $DBLK,R3      ;; SETUP THE OUTPUT POINTER
7675      MOV      #' ,R3)+      ;; SET THE FIRST CHARACTER TO A BLANK
7676      CLR      R2      ;; CLEAR THE BCD NUMBER
7677      MOV      $DTBL(R0),R1      ;; GET THE CONSTANT
7678      SUB      R1,R5      ;; FORM THIS BCD DIGIT
7679      BLT      4$      ;; BR IF DONE
7680      INC      R2      ;; INCREASE THE BCD DIGIT BY 1
7681      BR       3$
7682      4$: ADD      R1,R5      ;; ADD BACK THE CONSTANT
7683      TST      R2      ;; CHECK IF BCD DIGIT=0

```

```

7684 033060 001002      BNE      5$      ; FALL THROUGH IF 0
7685 033062 105716      TSTB     (SP)      ; STILL DOING LEADING 0'S?
7686 033064 100407      BMI      7$      ; BR IF YES
7687 033066 106316      5$: ASLB     (SP)      ; MSD?
7688 033070 103003      BCC      6$      ; BR IF NO
7689 033072 116663 000001 177777      MOVB     1(SP),-1(R3) ; YES--SET THE SIGN
7690 033100 052702 000060      BIS      #'0,R2 ; MAKE THE BCD DIGIT ASCII
7691 033104 052702 000040      6$: BIS      #' ,R2 ; MAKE IT A SPACE IF NOT ALREADY A DIGIT
7692 033110 110223      MOVB     R2,R3)+ ; PUT THIS CHARACTER IN THE OUTPUT BUFFER
7693 033112 005720      TST      (R0)+ ; JUST INCREMENTING
7694 033114 020027 000010      CMP      R0,#10 ; CHECK THE TABLE INDEX
7695 033120 002746      BLT      2$      ; GO DO THE NEXT DIGIT
7696 033122 003002      BGT      8$      ; GO TO EXIT
7697 033124 010502      MOV      R5,R2 ; GET THE LSD
7698 033126 000764      BR       6$      ; GO CHANGE TO ASCII
7699 033130 105726      8$: TSTB     (SP)+ ; WAS THE LSD THE FIRST NON-ZERO?
7700 033132 100003      BPL      9$      ; BR IF NO
7701 033134 116663 177777 177776      MOVB     -1(SP),-2(R3) ; YES--SET THE SIGN FOR TYPING
7702 033142 105013      9$: CLRB     (R3) ; SET THE TERMINATOR
7703 033144 012605      MOV      (SP)+,R5 ; POP STACK INTO R5
7704 033146 012603      MOV      (SP)+,R3 ; POP STACK INTO R3
7705 033150 012602      MOV      (SP)+,R2 ; POP STACK INTO R2
7706 033152 012601      MOV      (SP)+,R1 ; POP STACK INTO R1
7707 033154 012600      MOV      (SP)+,R0 ; POP STACK INTO R0
7708 033156 104401 033204      TYPE     $DBLK ; NOW TYPE THE NUMBER
7709 033162 016666 000002 000004      MOV      2(SP),4(SP) ; ADJUST THE STACK
7710 033170 012616      MOV      (SP)+,(SP)
7711 033172 000002      RTI ;,RETURN TO USER
7712 033174 023420      $DTBL: 10000.
7713 033176 001750      1000.
7714 033200 000144      100.
7715 033202 000012      10.
7716 033204 000004      $DBLK: .BLKW 4
7717
7718      .SBTTL  TTY INPUT ROUTINE
7719
7720      ;*****
7721      .ENABL  LSB
7722
7723      ;*****
7724      ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
7725      ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
7726      ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
7727      ;*WHEN OPERATING IN TTY FLAG MODE.
7728 033214 022737 000176 001154 $CKSWR: CMP      #SWREG,SWR ; IS THE SOFT-SWR SELECTED?
7729 033222 001074      BNE      15$      ; BRANCH IF NO
7730 033224 105777 145730      TSTB     2$TKS ; CHAR THERE?
7731 033230 100071      BPL      15$      ; IF NO, DON'T WAIT AROUND
7732 033232 117746 145724      MOVB     2$TKB,-(SP) ; SAVE THE CHAR
7733 033236 042716 177600      BIC      #177,(SP) ; STRIP-OFF THE ASCII
7734 033242 022726 000007      CMP      #7,(SP)+ ; IS IT A CONTROL G?
7735 033246 001062      BNE      15$      ; NO, RETURN TO USER
7736 033250 123727 001150 000001      CMPB     $AUTOB,#1 ; ARE WE RUNNING IN AUTO-MODE?
7737 033256 001456      BEQ      15$      ; BRANCH IF YES
7738
7739 033260 104401 033623      TYPE     , $CNTLG ;,ECHO THE CONTROL-G (↑G)

```

B12

```
7740 033264 104401 033630      SGT SWR: TYPE      SMSWR      :: TYPE CURRENT CONTENTS
7741 033270 013746 000176      MOV      SWREG, -(SP)  :: SAVE SWREG FOR TYPEOUT
7742 033274 104402              TYPOC              :: GO TYPE--OCTAL ASCII(ALL DIGITS)
7743 033276 104401 033641      TYPE      , SMNEW      :: PROMPT FOR NEW SWR
7744 033302 005046              CLR      -(SP)        :: CLEAR COUNTER
7745 033304 005046              CLR      -(SP)        :: THE NEW SWR
7746 033306 105777 145646      7$:      TSTB      2$TKS      :: CHAR THERE?
7747 033312 100375              BPL      7$           :: IF NOT TRY AGAIN
7748
7749 033314 117746 145642      MOVB      2$TKB, -(SP)  :: PICK UP CHAR
7750 033320 042716 177600      BIC      #1C177, (SP) :: MAKE IT 7-BIT ASCII
7751
7752
7753
7754 033324 021627 000025      9$:      CMP      (SP), #25      :: IS IT A CONTROL-U?
7755 033330 001005              BNE      10$          :: BRANCH IF NOT
7756 033332 104401 033616      TYPE      $CNTLU      :: YES, ECHO CONTROL-U (↑U)
7757 033336 062706 000006      20$:     ADD      #6, SP      :: IGNORE PREVIOUS INPUT
7758 033342 000757              BR      19$          :: LET'S TRY IT AGAIN
7759
7760
7761 033344 021627 000015      10$:     CMP      (SP), #15      :: IS IT A <CR>?
7762 033350 001022              BNE      16$          :: BRANCH IF NO
7763 033352 005766 000004      TST      4(SP)          :: YES, IS IT THE FIRST CHAR?
7764 033356 001403              BEQ      11$          :: BRANCH IF YES
7765 033360 016677 000002 145566 MOV      2(SP), 2$SWR      :: SAVE NEW SWR
7766 033366 062706 000006      11$:     ADD      #6, SP      :: CLEAR UP STACK
7767 033372 104401 001201      14$:     TYPE      $CRLF      :: ECHO <CR> AND <LF>
7768 033376 123727 001151 000001 CMPB      $INTAG, #1      :: RE-ENABLE TTY KBD INTERRUPTS?
7769 033404 001003              BNE      15$          :: BRANCH IF NOT
7770 033406 012777 000100 145544 MOV      #100, 2$TKS      :: RE-ENABLE TTY KBD INTERRUPTS
7771 033414 000002              RTI                    :: RETURN
7772 033416 004737 031730      16$:     JSR      PC, $TYPEC      :: ECHO CHAR
7773 033422 021627 000060      CMP      (SP), #60      :: CHAR < L
7774 033426 002420              BLT      18$          :: BRANCH IF YES
7775 033430 021627 000067      CMP      (SP), #67      :: CHAR > ?
7776 033434 003015              BGT      18$          :: BRANCH IF YES
7777 033436 042726 000060      BIC      #60, (SP)+      :: STRIP-OFF ASCII
7778 033442 005766 000002      TST      2(SP)          :: IS THIS THE FIRST CHAR
7779 033446 001403              BEQ      17$          :: BRANCH IF YES
7780 033450 006316              ASL      (SP)          :: NO, SHIFT PRESENT
7781 033452 006316              ASL      (SP)          :: CHAR OVER TO MAKE
7782 033454 006316              ASL      (SP)          :: ROOM FOR NEW ONE.
7783 033456 005266 000002      17$:     INC      2(SP)          :: KEEP COUNT OF CHAR
7784 033462 056616 177776      BIS      -2(SP), (SP)  :: SET IN NEW CHAR
7785 033466 000707              BR      7$           :: GET THE NEXT ONE
7786 033470 104401 001200      18$:     TYPE      $QUES      :: TYPE ?<CR><LF>
7787 033474 000720              BR      20$          :: SIMULATE CONTROL-U
7788 .DSABL L5B
7789
7790
7791
7792
7793
7794
7795
```

```
:: *****
:: *THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
:: *CALL:
:: *      RDCHR      :: INPUT A SINGLE CHARACTER FROM THE TTY
:: *      RETURN HERE :: CHARACTER IS ON THE STACK
```

```

7796          ;*                               ;: WITH PARITY BIT STRIPPED OFF
7797          ;
7798          ;
7799 033476 011646          $RDCHR: MOV      (SP), -(SP)          ;: PUSH DOWN THE PC
7800 033500 016666 000004 000002      MOV      4(SP), 2(SP)      ;: SAVE THE PS
7801 033506 105777 145446      1$:      TSTB      2$TKS          ;: WAIT FOR
7802 033512 100375          BPL      1$          ;: A CHARACTER
7803 033514 117766 145442 000004      MOVB      2$TKB, 4(SP)      ;: READ THE TTY
7804 033522 042766 177600 000004      BIC      4(C<177>, 4(SP)    ;: GET RID OF JUNK IF ANY
7805 033530 026627 000004 000023      CMP      4(SP), 23          ;: IS IT A CONTROL-S?
7806 033536 001013          BNE      3$          ;: BRANCH IF NO
7807 033540 105777 145414      2$:      TSTB      2$TKS          ;: WAIT FOR A CHARACTER
7808 033544 100375          BPL      2$          ;: LOOP UNTIL ITS THERE
7809 033546 117746 145410      MOVB      2$TKB, -(SP)          ;: GET CHARACTER
7810 033552 042716 177600          BIC      4(C177, (SP)          ;: MAKE IT 7-BIT ASCII
7811 033556 022627 000021      CMP      (SP)+, 21          ;: IS IT A CONTROL-Q?
7812 033562 001366          BNE      2$          ;: IF NOT DISCARD IT
7813 033564 000750          BR      1$          ;: YES, RESUME
7814 033566 026627 000004 000140      3$:      CMP      4(SP), 140      ;: IS IT UPPER CASE?
7815 033574 002407          BLT      4$          ;: BRANCH IF YES
7816 033576 026627 000004 000175      CMP      4(SP), 175          ;: IS IT A SPECIAL CHAR?
7817 033604 003003          BGT      4$          ;: BRANCH IF YES
7818 033606 042766 000040 000004      BIC      40, 4(SP)          ;: MAKE IT UPPER CASE
7819 033614 000002          RTI          ;: GO BACK TO USER
7820 033616 052536 005015 000      $CNTLU: .ASCIZ /?U<15><12>      ;: CONTROL "U"
7821 033623 136 006507 000012      $CNTLG: .ASCIZ /?G<15><12>      ;: CONTROL "G"
7822 033630 005015 053523 020122      $MSWR: .ASCIZ <15><12>/SWR = /
7823 033636 020075 000
7824 033641 040 047040 053505      $MNEW: .ASCIZ / NEW = /
7825 033646 036440 000040

```

.SBTTL RANDOM NUMBER GENERATOR ROUTINE

```

7826
7827
7828
7829 ;: *****
7830 ;: *THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
7831 ;: *WITH A RANGE OF 0 TO 2(+33)-1.
7832 ;: *CALL:
7833 ;: *      JSR      PC, $RAND          ;: CALL THE ROUTINE
7834 ;: *      RETURN          ;: RETURN HERE THE RANDOM
7835 ;: *                          ;: NUMBER WILL BE IN
7836 ;: *                          ;: $HINUM, $LONUM
7837 ;:
7838 $RAND:
7839 033652 010046          MOV      R0, -(SP)          ;: PUSH R0 ON STACK
7840 033654 010146          MOV      R1, -(SP)          ;: PUSH R1 ON STACK
7841 033656 010246          MOV      R2, -(SP)          ;: PUSH R2 ON STACK
7842 033660 013700 033752      MOV      $LONUM, R0          ;: SET R0 WITH LOW
7843 033664 013701 033750      MOV      $HINUM, R1          ;: SET R1 WITH HIGH
7844 033670 012702 177771      MOV      #-7, R2          ;: SET SHIFT COUNT
7845 033674 006300      1$:      ASL      R0          ;: SHIFT R0 LEFT AND
7846 033676 006101          ROL      R1          ;: ROTATE CARRY INTO R1 AND
7847 033700 005202          INC      R2          ;: CHECK FOR DONE
7848 033702 001374          BNE      1$          ;: CONTINUE SHIFT LOOP
7849 033704 063700 033752      ADD      $LONUM, R0          ;: ADD NUMBER TO MAKE X 129
7850 033710 005501          ADC      R1          ;: PROPOGATE CARRY
7851 033712 063701 033750      ADD      $HINUM, R1          ;: ADD NUMBER TO MAKE X 129

```

7852 033716 062700 001057
7853 033722 005501
7854 033724 062701 047401
7855 033730 010037 033752
7856 033734 010137 033750
7857 033740 012602
7858 033742 012601
7859 033744 012600
7860 033746 000207
7861 033750 176543
7862 033752 123456
7863
7864
7865
7866
7867
7868
7869
7870
7871
7872
7873
7874
7875
7876
7877
7878
7879
7880
7881 033754
7882 033754 010046
7883 033756 010146
7884 033760 010246
7885 033762 010346
7886 033764 010446
7887 033766 010546
7888 033770 016646 000022
7889 033774 016646 000022
7890 034000 016646 000022
7891 034004 016646 000022
7892 034010 000002
7893
7894
7895
7896
7897 034012
7898 034012 012666 000022
7899 034016 012666 000022
7900 034022 012666 000022
7901 034026 012666 000022
7902 034032 012605
7903 034034 012604
7904 034036 012603
7905 034040 012602
7906 034042 012601
7907 034044 012600

```

ADD      #1057,R0      ;; ADD LOW CONSTANT
ADC      R1             ;; PROPOGATE CARRY
ADD      #47401,R1     ;; ADD HIGH CONSTANT
MOV      R0,$LONUM     ;; SAVE R0
MOV      R1,$SHINUM    ;; SAVE R1
MOV      (SP)+,R2      ;; POP STACK INTO R2
MOV      (SP)+,R1      ;; POP STACK INTO R1
MOV      (SP)+,R0      ;; POP STACK INTO R0
RTS      PC            ;; RETURN
$SHINUM: .WORD 176543
$LONUM:  .WORD 123456

.SBTTL   SAVE AND RESTORE R0-R5 ROUTINES

;*****
;SAVE R0-R5
;CALL:
;*      SAVREG
;UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
;*TOP---(+16)
;*+2---(+18)
;*+4---R5
;*+6---R4
;*+8---R3
;*+10---R2
;*+12---R1
;*+14---R0

$SAVREG:
MOV      R0,-(SP)      ;; PUSH R0 ON STACK
MOV      R1,-(SP)      ;; PUSH R1 ON STACK
MOV      R2,-(SP)      ;; PUSH R2 ON STACK
MOV      R3,-(SP)      ;; PUSH R3 ON STACK
MOV      R4,-(SP)      ;; PUSH R4 ON STACK
MOV      R5,-(SP)      ;; PUSH R5 ON STACK
MOV      22(SP),-(SP)  ;; SAVE PS OF MAIN FLOW
MOV      22(SP),-(SP)  ;; SAVE PC OF MAIN FLOW
MOV      22(SP),-(SP)  ;; SAVE PS OF CALL
MOV      22(SP),-(SP)  ;; SAVE PC OF CALL
RTI

;RESTORE R0-R5
;CALL:
;*      RESREG
$RESREG:
MOV      (SP)+,22(SP)  ;; RESTORE PC OF CALL
MOV      (SP)+,22(SP)  ;; RESTORE PS OF CALL
MOV      (SP)+,22(SP)  ;; RESTORE PC OF MAIN FLOW
MOV      (SP)+,22(SP)  ;; RESTORE PS OF MAIN FLOW
MOV      (SP)+,R5      ;; POP STACK INTO R5
MOV      (SP)+,R4      ;; POP STACK INTO R4
MOV      (SP)+,R3      ;; POP STACK INTO R3
MOV      (SP)+,R2      ;; POP STACK INTO R2
MOV      (SP)+,R1      ;; POP STACK INTO R1
MOV      (SP)+,R0      ;; POP STACK INTO R0

```

```

7908 034046 000002 RTI
7909
7910 .SBTTL DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
7911
7912 ;*****
7913 ;*THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
7914 ;*DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
7915 ;*POSITIVE.
7916 ;*CALL
7917 ;*      MOV      #PNTR, -(SP)      ;; POINTER TO LOW WORD OF BINARY NUMBER
7918 ;*      JSR      PC, @#SDB2D      ;; THE FIRST ADDRESS OF ASCII
7919 ;*      RETURN                      ;; IS ON THE STACK
7920
7921
7922
7923 034050 104412 $SDB2D: SAVREG      ;; SAVE REGISTERS
7924 034052 016602 000002 MOV      2(SP), R2      ;; PICKUP THE DATA POINTER
7925 034056 012700 034230 MOV      #SDECVL, R0      ;; GET ADDRESS OF "SDECVL" STRING
7926 034062 010066 000002 MOV      R0, 2(SP)      ;; PUT ADDRESS OF ASCII STRING ON STACK
7927 034066 012201 MOV      (R2)+, R1      ;; PICKUP THE BINARY NUMBER
7928 034070 012202 MOV      (R2)+, R2
7929 034072 012737 000012 034146 MOV      #10, R4      ;; SET UP TO DO 10 CONVERSIONS
7930 034100 012704 034160 MOV      #STNPWR, R4      ;; ADDRESS OF TEN POWER
7931 034104 012705 034162 MOV      #STNPWR+2, R5
7932 034110 005003 1$: CLR      R3      ;; CLEAR PARTIAL
7933 034112 161401 2$: SUB      (R4), R1      ;; SUBTRACT TEN POWER
7934 034114 005602 SBC      R2
7935 034116 161502 SUB      (R5), R2
7936 034120 002402 BLT      3$      ;; BR IF TEN POWER TO LARGE
7937 034122 005203 INC      R3      ;; ADD 1 TO PARTIAL
7938 034124 000772 BR      2$      ;; LOOP
7939 034126 062401 3$: ADD      (R4)+, R1      ;; RESTORE SUBTRACTED VALUE
7940 034130 005502 ADC      R3
7941 034132 062402 ADD      (R4)+, R2
7942 034134 022525 CMP      (R5)+, (R5)+      ;; MOVE TO NEXT TEN POWER
7943 034136 052703 000060 BIS      #0, R3      ;; CHANGE PARTIAL TO ASCII
7944 034142 110320 MOV      R3, (R0)+      ;; SAVE IT
7945 034144 005327 DEC      (PC)+      ;; DONE?
7946 034146 000000 4$: .WORD 0
7947 034150 001357 BNE      1$      ;; BR IF NO
7948 034152 105020 CLRB      (R0)+      ;; TERMINATOR
7949 034154 104413 RESREG      ;; RESTORE REGISTERS
7950 034156 000207 RTS      PC      ;; RETURN
7951 034160 145000 $STNPWR: 145000      ;; 1.0E09
7952 034162 035632 35632
7953 034164 160400      ;; 1.0E08
7954 034166 002765 2765
7955 034170 113200 113200      ;; 1.0E07
7956 034172 000230 230
7957 034174 041100 041100      ;; 1.0E06
7958 034176 000017 17
7959 034200 103240 103240      ;; 1.0E05
7960 034202 000001 1
7961 034204 023420 23420      ;; 1.0E04
7962 034206 000000 0
7963 034210 001750 1750      ;; 1.0E03

```


7964 034212 000000
7965 034214 000144
7966 034216 000000
7967 034220 000012
7968 034222 000000
7969 034224 000001
7970 034226 000000
7971 034230 000014

0
144 ;:1.0E02
0
12 ;:1.0E01
0
1 ;:1.0E00
0
\$DECVL: .BLKB 12. ;:RESERVE STORAGE FOR ASCII STRING

.SBTTL DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE

;;*****
;;THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
;;UNSIGNED OCTAL ASCII NUMBER.

;;CALL

;;* MOV #PNTR, -(SP, ;: POINTER TO LOW WORD OF BINARY NUMBER
;;* JSR PC, @SDB20 ;: CALL THE ROUTINE
;;* RETURN ;: THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK

7984 034244 104412
7985 034246 016601 000002
7986 034252 012705 034363
7987 034256 012704 000014
7988 034262 012703 177770
7989 034266 012100
7990 034270 012101
7991 034272 005002
7992 034274 110245
7993 034276 010002
7994 034300 005304
7995 034302 003007
7996 034304 001405
7997 034306 005205
7998 034310 010566 000002
7999 034314 104413
8000 034316 000207
8001 034320 006203
8002 034322 006001
8003 034324 006000
8004 034326 006001
8005 034330 006000
8006 034332 006001
8007 034334 006000
8008 034336 040302
8009 034340 062702 000060
8010 034344 000753
8011 034346 000016

\$SDB20: SAVREG ;: SAVE ALL REGISTERS
MOV 2(SP), R1 ;: PICKUP THE POINTER TO LOW WORD
MOV #SDB20+13., R5 ;: POINTER TO DATA TABLE
MOV #12, R4 ;: DO ELEVEN CHARACTERS
MOV #C7, R3 ;: MASK
MOV (R1)+, R0 ;: LOWER WORD
MOV (R1)+, R1 ;: HIGH WORD
CLR R2 ;: TERMINATOR
1\$: MOV R2, -(R5) ;: PUT CHARACTER IN DATA TABLE
MOV R0, R2 ;: GET THIS DIGIT
DEC R4 ;: COUNT THIS CHARACTER
BGT 3\$;: BR IF NOT THE LAST DIGIT
BEQ 2\$;: BR IF IT IS THE LAST DIGIT
INC R5 ;: ALL DIGITS DONE-ADJUST POINTER FOR FIRST
MOV R5, 2(SP) ;: ASCII CHAR. & PUT IT ON THE STACK
RESREG ;: RESTORE ALL REGISTERS
RTS PC ;: RETURN TO USER
2\$: ASR R3 ;: POSITION THE MASK FOR THE LAST DIGIT
3\$: ROR R1 ;: POSITION THE BINARY NUMBER FOR
ROR R0 ;: THE NEXT OCTAL DIGIT
ROR R1
ROR R0
ROR R1
ROR R0
BIC R3, R2 ;: MASK OUT ALL JUNK
ADD #0, R2 ;: MAKE THIS CHAR. ASCII
BR 1\$;: GO PUT IT IN THE DATA TABLE
\$SDB20: .BLKB 14. ;: RESERVE DATA TABLE

.SBTTL TRAP DECODER

;;*****
;;THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
;;AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
;;OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
;;GO TO THAT ROUTINE.

```

8020
8021 034364 010046
8022 034366 016600 000002
8023 034372 005740
8024 034374 111000
8025 034376 006300
8026 034400 016000 034420
8027 034404 000200
8028
8029
8030
8031
8032 034406 011646
8033 034410 016666 000004 000002
8034 034416 000002
8035
8036
8037
8038
8039
8040
8041
8042
8043 034420 034406
8044 034422 031516
8045 034424 032566
8046 034426 032542
8047 034430 032602
8048 034432 032770
8049
8050 034434 033264
8051
8052 034436 033214
8053 034440 033476
8054 034442 030700
8055 034444 033754
8056 034446 034012
8057 034450 030124
8058 000032
8059
8060
8061
8062
8063
8064
8065
8066
8067
8068
8069
8070
8071
8072
8073
8074
8075

```

STRAP: MOV RO, -(SP) ;: SAVE RO
 MOV 2(SP), RO ;: GET TRAP ADDRESS
 TST -(RO) ;: BACKUP BY 2
 MOVB (RO), RO ;: GET RIGHT BYTE OF TRAP
 ASL RO ;: POSITION FOR INDEXING
 MOV STRPAD(RO), RO ;: INDEX TO TABLE
 RTS RO ;: GO TO ROUTINE

;: THIS IS USE TO HANDLE THE "GETPRI" MACRO

STRAP2: MOV (SP), -(SP) ;: MOVE THE PC DOWN
 MOV 4(SP), 2(SP) ;: MOVE THE PSW DOWN
 RTI ;: RESTORE THE PSW

.SBTTL TRAP TABLE

;: *THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
 ;: *BY THE "TRAP" INSTRUCTION.

ROUTINE	STARTING ADDRESS
STRAP2	034406
STRAP2	031516
STRAP2	032566
STRAP2	032542
STRAP2	032602
STRAP2	032770
STRAP2	033264
STRAP2	033214
STRAP2	033476
STRAP2	030700
STRAP2	033754
STRAP2	034012
STRAP2	030124
STRAP2	000032

STRAP2: .WORD STRAP2
 STRAP2: .CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
 STRAP2: .CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
 STRAP2: .CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
 STRAP2: .CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
 STRAP2: .CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
 STRAP2: .CALL=GTSWR TRAP+6(104406) GET SOFT-SWR SETTING
 STRAP2: .CALL=CKSWR TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
 STRAP2: .CALL=RDCHR TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
 STRAP2: .CALL=RDLIN TRAP+11(104411) TTY TYPEIN STRING ROUTINE
 STRAP2: .CALL=SAVREG TRAP+12(104412) SAVE RO-R5 ROUTINE
 STRAP2: .CALL=RESREG TRAP+13(104413) RESTORE RO-R5 ROUTINE
 STRAP2: .CALL=DISPLY TRAP+14(104414) ROUTINE TO TYPE ERROR MESSAGES

\$TERM=.-STRAP2

;: *****

.SBTTL SINGLE/DUAL PORT RM70/RM03 DRIVER (REV 5.1)-24-AUG-77
 ;: NEW DRIVE TYPE ID FOR RM02 *****
 ;: 10-AUG-77 *****

;: COPYRIGHT (C) 1977
 ;: DIGITAL EQUIPMENT CORP.
 ;: MAYNARD, MA 01754
 ;: AUTHOR(S): JIM LACEY/CHUCK HESS/C. CHEN

;: *****

;: STORAGE FOR RMD5, RMER1, RMER2, AND RMMR2 ON AN ERROR "2"


```

8076      ;RMERRS = RMD5
8077      ;RMERRS+2 = RMER1
8078      ;RMERRS+4 = RMER2
8079      ;RMERRS+6 = RMMR2
8080
8081      034452 000000 000000 000000 RMERRS: .WORD 0,0,0,0
8082      034460 000000
8083
8084      ;TABLE OF DRIVE ACTIVE INDICATORS (DRVACT=8 BYTES)
8085      ;DRVACT=0 IF DRIVE IS IDLE
8086      ;DRVACT>0 IF DRIVE IS ACTIVE WITH A COMMAND
8087      ;DRVACT<0 IF DRIVE IS ACTIVE WITH AN ERROR RECOVERY OPERATION
8088
8089      034462 000      DRVACT: .BYTE 0      ;DRIVE 0
8090      034463 000      .BYTE 0      ;DRIVE 1
8091      034464 000      .BYTE 0      ;DRIVE 2
8092      034465 000      .BYTE 0      ;DRIVE 3
8093      034466 000      .BYTE 0      ;DRIVE 4
8094      034467 000      .BYTE 0      ;DRIVE 5
8095      034470 000      .BYTE 0      ;DRIVE 6
8096      034471 000      .BYTE 0      ;DRIVE 7
8097
8098      ;TABLE OF DRIVE STATUS INDICATORS (DRVSTA=8 BYTES)
8099      ;DRVSTA=0 IF DRIVE IS OFFLINE OR NONEXISTENT
8100      ;DRVSTA>0 IF DRIVE IS ONLINE
8101      ;DRVSTA<0 IF DRIVE IS UNSAFE
8102
8103      034472 000      DRVSTA: .BYTE 0      ;DRIVE 0
8104      034473 000      .BYTE 0      ;DRIVE 1
8105      034474 000      .BYTE 0      ;DRIVE 2
8106      034475 000      .BYTE 0      ;DRIVE 3
8107      034476 000      .BYTE 0      ;DRIVE 4
8108      034477 000      .BYTE 0      ;DRIVE 5
8109      034500 000      .BYTE 0      ;DRIVE 6
8110      034501 000      .BYTE 0      ;DRIVE 7
8111
8112      ;TABLE OF DRIVE TYPES (DRVTYPE=8 BYTES)
8113      ;DRVTYPE=0 IF DRIVE IS NONEXISTENT (DRVSTA=0, ALSO)
8114      ;DRVTYPE=5 IF DRIVE IS RM02 *****
8115      ;DRVTYPE=4 IF DRIVE IS RM03
8116      ;DRVTYPE=-1 IF NOT RM03
8117
8118      034502 000      DRVTYPE: .BYTE 0      ;DRIVE 0
8119      034503 000      .BYTE 0      ;DRIVE 1
8120      034504 000      .BYTE 0      ;DRIVE 2
8121      034505 000      .BYTE 0      ;DRIVE 3
8122      034506 000      .BYTE 0      ;DRIVE 4
8123      034507 000      .BYTE 0      ;DRIVE 5
8124      034510 000      .BYTE 0      ;DRIVE 6
8125      034511 000      .BYTE 0      ;DRIVE 7
8126
8127      ;TABLE OF DUAL PORT INITIALIZATION INDICATORS
8128      ;DPINT=0 IF INITIALIZATION IS NOT ACTIVE ON THE DRIVE
8129      ;DPINT<0 IF INITIALIZATION IS IN PROGRESS
8130
8131      034512 000      DPINT: .BYTE 0      ;DRIVE 0

```

0132 034513 000
 0133 034514 000
 0134 034515 000
 0135 034516 000
 0136 034517 000
 0137 034520 000
 0138 034521 000
 0139
 0140
 0141
 0142
 0143
 0144 034522 000
 0145 034523 000
 0146 034524 000
 0147 034525 000
 0148 034526 000
 0149 034527 000
 0150 034530 000
 0151 034531 000
 0152
 0153
 0154
 0155
 0156
 0157 034532 000000
 0158
 0159
 0160
 0161
 0162
 0163
 0164
 0165 034534 000000
 0166
 0167
 0168
 0169
 0170
 0171 034536 000
 0172
 0173
 0174
 0175
 0176
 0177 034537 000
 0178
 0179
 0180
 0181
 0182
 0183
 0184 034540 000
 0185 034541 000
 0186 034542 000
 0187 034543 000

.BYTE 0 ;DRIVE 1
 .BYTE 0 ;DRIVE 2
 .BYTE 0 ;DRIVE 3
 .BYTE 0 ;DRIVE 4
 .BYTE 0 ;DRIVE 5
 .BYTE 0 ;DRIVE 6
 .BYTE 0 ;DRIVE 7

 ;TABLE OF PENDING DUAL PORT REQUESTS
 ;DPRQS=0 IF THAT A DUAL PORT REQUEST IS NOT PENDING FOR THAT DRIVE
 ;DPRQS<0 IF THAT A DUAL PORT REQUEST IS PENDING FOR THAT DRIVE

 DPRQS: .BYTE 0 ;DRIVE 0
 .BYTE 0 ;DRIVE 1
 .BYTE 0 ;DRIVE 2
 .BYTE 0 ;DRIVE 3
 .BYTE 0 ;DRIVE 4
 .BYTE 0 ;DRIVE 5
 .BYTE 0 ;DRIVE 6
 .BYTE 0 ;DRIVE 7

 ;TRANSFER WAIT FLAG (TRNSWT=1 WORD)
 ;THIS IS A ONE WORD QUEUE. IT WILL CONTAIN THE ADDRESS OF
 ;"DPB" OF THE I/O OPERATION.

 TRNSWT: .WORD 0

 ;SEARCH WAIT KEYS (SRCHWT=1 WORD)
 ;THIS IS A ONE WORD QUEUE THAT WILL CONTAIN A KEY FOR EACH OF
 ;THE DRIVES THAT ARE PERFORMING A SEARCH COMMAND FOR THE I/O
 ;REQUEST THAT IS AT THE TOP OF THEIR REQUEST QUEUE.
 ;EACH DRIVE IS ASSIGNED ONE BIT, STARTING AT BIT00 FOR DRIVE 0.

 SRCHWT: .WORD 0

 ;RM03 DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
 ;ACTDRV=0 IF DRIVER IS INACTIVE
 ;ACTDRV>0 IF DRIVER IS ACTIVE

 ACTDRV: .BYTE 0

 ;SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)
 ;ACTSTR=0 IF SOFTWARE TIMER ROUTINE IS INACTIVE
 ;ACTSTR>0 IF SOFTWARE TIMER ROUTINE IS ACTIVE

 ACTSTR: .BYTE 0

 ;UNLOAD FLAG (ULDFLG=8 BYTES)
 ;ULDFLG=0 IF NO UNLOAD COMMAND
 ;ULDFLG>0 IF UNLOAD COMMAND IN PROGRESS
 ;ULDFLG<0 IF UNLOAD COMMAND IN WAIT QUEUE

 ULDFLG: .BYTE 0 ;DRIVE 0
 .BYTE 0 ;DRIVE 1
 .BYTE 0 ;DRIVE 2
 .BYTE 0 ;DRIVE 3

CZRM880 RM03/2 PERF EXER
CZRM88.P11 21-NOV-77 15:34

MACY11 30(1046) 22-NOV-77 10:06 PAGE 153
SINGLE/DUAL PORT RH70/RM03 DRIVER (REV 5.1)-24-AUG-77

SEQ 0152

8188	034544	000	.BYTE	0	;DRIVE 4
8189	034545	000	.BYTE	0	;DRIVE 5
8190	034546	000	.BYTE	0	;DRIVE 6
8191	034547	000	.BYTE	0	;DRIVE 7
8192					
8193			;LOOK AHEAD COUNT (LACNT=8 BYTES)		
8194			;LACNT WILL INDICATE THE NUMBER OF LOOK AHEADS PERFORMED		
8195					
8196	034550	000	LACNT: .BYTE	0	;DRIVE 0
8197	034551	000	.BYTE	0	;DRIVE 1
8198	034552	000	.BYTE	0	;DRIVE 2
8199	034553	000	.BYTE	0	;DRIVE 3
8200	034554	000	.BYTE	0	;DRIVE 4
8201	034555	000	.BYTE	0	;DRIVE 5
8202	034556	000	.BYTE	0	;DRIVE 6
8203	034557	000	.BYTE	0	;DRIVE 7
8204					
8205			;SAVE REGISTERS FLAG (SAVEFG=1 WORD)		
8206			;SAVEFG<0 IF SAVE THE RH70/RM03 REGISTERS WHEN THE		
8207			;OPERATION IS COMPLETED AS PER (DPB+14).		
8208			;SAVEFG=0 IF SAVE THE RH70/RM03 REGISTERS, AS PER		
8209			;(DPB+14), AFTER AN ERROR.		
8210					
8211	034560	000000	SAVEFG: .WORD	0	
8212					
8213			;SEEK FLAG (SEEKFG=1 WORD)		
8214			;SEEKFG=0 IF WHEN THE DISK ADDRESS ISN'T IN THE WINDOW		
8215			;FOR A DATA TRANSFER START A SEARCH COMMAND		
8216			;SEEKFG<0 IF DATA TRANSFER WILL DO IMPLIED SEEKS,		
8217			;DISREGARD THE WINDOW		
8218					
8219	034562	177777	SEEKFG: .WORD	-1	
8220					
8221			;TIMEOUT TABLE (TIMER=8 WORDS)		
8222			;THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION		
8223					
8224	034564	177777	TIMER: .WORD	-1	;DRIVE 0
8225	034566	177777	.WORD	-1	;DRIVE 1
8226	034570	177777	.WORD	-1	;DRIVE 2
8227	034572	177777	.WORD	-1	;DRIVE 3
8228	034574	177777	.WORD	-1	;DRIVE 4
8229	034576	177777	.WORD	-1	;DRIVE 5
8230	034600	177777	.WORD	-1	;DRIVE 6
8231	034602	177777	.WORD	-1	;DRIVE 7
8232					
8233			;DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)		
8234			;DTUW<0 IF NO DATA TRANSFER UNDERWAY		
8235			;DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE N		
8236					
8237	034604	177777	DTUW: .WORD	-1	
8238					
8239			;ATTENTION BITS TABLE (ATABIT=8 BYTES)		
8240			;THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES		
8241			;ATTENTION BIT		
8242					
8243	034606	001	ATABIT: .BYTE	1	;DRIVE 0

8244	034607	002	.BYTE	2	;DRIVE 1
8245	034610	004	.BYTE	4	;DRIVE 2
8246	034611	010	.BYTE	10	;DRIVE 3
8247	034612	020	.BYTE	20	;DRIVE 4
8248	034613	040	.BYTE	40	;DRIVE 5
8249	034614	100	.BYTE	100	;DRIVE 6
8250	034615	200	.BYTE	200	;DRIVE 7
8251					
8252			;FSRM03 TO RH70 "MASSBUS CONTROL BUS PARITY ERRORS" (MCPE) ALLOWED BEFORE		
8253			;CALLING IT FATAL (MCPEM=1 WORD)		
8254					
8255	034616	000003	MCPEM:	.WORD	3
8256					
8257			;STORAGE FOR RMADR (THE FIRST ADDRESS (776700) OF THE RH70/RM03),		
8258			;RMVEC (THE VECTOR ADDRESS (254)), AND RMVEC+2 (THE BR LEVEL (5)).		
8259					
8260	034620	176700	RMADR:	.WORD	176700
8261	034622	000254	RMVEC:	.WORD	254,5*32.
8262					
8263			;MAXIMUM NUMBER OF LOOK AHEADS ALLOWED IS 4 (MXLACT=1 WORD)		
8264					
8265	034626	000004	MXLACT:	.WORD	4
8266			;MAXIMUM DELTA DELAY IS 8 SECTORS (MXDLTA=1 WORD)		
8267					
8268	034630	001000	MXDLTA:	.WORD	8*64.
8269			;MINIMUM DELTA DELAY IS 2 SECTORS (MNDLTA=1 WORD)		
8270					
8271	034632	000200	MNDLTA:	.WORD	2*64.
8272			;MAXIMUM SEARCH FOR I/O WINDOW IS 5 SECTORS (MXWNDW=1 WORD)		
8273					
8274	034634	000005	MXWNDW:	.WORD	5
8275					
8276			;DEFINITIONS OF THE RH70/RM03 ADDRESS INDEXES		
8277					
8278	000000		RMCS1=0		;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00)
8279	000002		RMWC=2		;WORD COUNT REGISTER (NOT A DRIVE REG)
8280	000004		RMBA=4		;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
8281	000006		RMDA=6		;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 05)
8282	000010		RMCS2=10		;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
8283	000012		RMDS=12		;DRIVE STATUS REGISTER (DRIVE REG 01)
8284	000014		RMER1=14		;ERROR REGISTER #1 (DRIVE REG. 02)
8285	000016		RMA5=16		;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 04)
8286	000020		RMLA=20		;LOOK AHEAD REGISTER (DRIVE REG. 07)
8287	000022		RMOB=22		;DATA BUFFER REGISTER (NOT A DRIVE REG.)
8288	000024		RMMR1=24		;MAINTAINABILITY REGISTER (DRIVE REG. 03)
8289	000026		RMDT=26		;DRIVE TYPE REGISTER (DRIVE REG. 06)
8290	000030		RMSN=30		;SERIAL NUMBER REGISTER (DRIVE REG. 10)
8291	000032		RMOF=32		;OFFSET REGISTER (DRIVE REG. 11)
8292	000034		RMDC=34		;DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
8293	000036		RMHR=36		;DUMMY ADDRESS REGISTER (DRIVE REG. 13)
8294	000040		RMMR2=40		;MAINTENANCE REGISTER #2
8295	000042		RMER2=42		;ERROR REGISTER #2 (DRIVE REG. 15)
8296	000044		RMEC1=44		;ECC POSITION REGISTER (DRIVE REG. 16)
8297	000046		RMEC2=46		;ECC PATTERN REGISTER (DRIVE REG. 17)
8298					
8299			;RH70/RM03 DRIVER INITIALIZATION CODE		

```

8300      ; THIS ROUTINE WILL DETERMINE WHICH RM03 DRIVES ARE
8301      ; AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
8302      ; TO THE PROPER STATE FOR EACH DRIVE.
8303      ; NOTE: THIS ROUTINE CALLS DRVINT
8304
8305      : CALL
8306
8307      JSR    PC,RMINIT
8308      RETURN
8309
8310      : NOTE: THE 'P' OR 'L' CLOCK MUST BE STARTED
8311
8312      RMINIT: SAVREG
8313      MOV     2#PS, -(SP)      ; SAVE R0 - R5
8314      MOV     2#PS, 2#PS      ; SAVE THE PRESENT PROCESSOR STATUS
8315      JSR     PC,CLIQUE      ; CHANGE THE PRIORITY TO 5
8316      MOV     2#RMERRS,R1    ; CLEAR ALL REQUEST QUEUES
8317      MOV     2#SEEKFG,R2    ; FIRST ADDRESS TO BE CLEARED
8318      MOV     2#SEEKFG,R2    ; LAST ADDRESS TO BE CLEARED
8319      1$:    CLR     (R1)+    ; CLEAR
8320      CMP     R1,R2          ; ARE WE DONE?
8321      BLO     1$            ; BRANCH IF NO
8322      MOV     2#DTUW,R2      ; LAST ADDRESS
8323      2$:    MOV     2#-1,(R1)+ ; INITIALIZE
8324      CMP     R1,R2          ; DONE?
8325      BLOS    2$            ; LOOP IF NO
8326      CLR     DRVSTA        ; SET ALL DRIVES TO OFFLINE
8327      CLR     DRVSTA+2
8328      CLR     DRVSTA+4
8329      CLR     DRVSTA+6
8330      MOV     RMVEC,R3      ; SETUP THE RH70/RM03 VECTOR
8331      MOV     2#ISR,(R3)+
8332      MOV     RMVEC+2,(R3)
8333      MOV     RMADR,R4      ; FIRST ADDRESS OF RH70/RM03
8334      MOV     2#BIT05,RMCS2(R4) ; MASSBUS INIT
8335      CLR     R1            ; START WITH DRIVE 0
8336      3$:    JSR     R0,DRVINT ; INIT THE DRIVE
8337      BR      4$            ; 'DVA' NOT SET OR PARITY ERROR
8338      BR      5$            ; NORMAL RETURN
8339      CLR     DRVSTA(R1)    ; SET DRIVE STATUS TO OFFLINE
8340      INC     R1            ; GO TO NEXT DRIVE
8341      BIC     2#107,R1      ; MASK OUT UNUSED BITS
8342      BNE     3$            ; BR IF MORE DRIVES TO GO
8343      MOV     2#7,R1        ; START WITH DRIVE 7
8344      CLR     2#PS          ; CLEAR THE PROCESSOR STATUS
8345      6$:    TSTB    DPINT(R1) ; WAITING FOR DRIVE TO SWITCH PORTS ?
8346      BEQ     8$            ; BR NOT WAITING
8347      JSR     PC,SET.IE     ; SET INTERRUPT
8348      TSTB    DPINT(R1)    ; DRIVE SWITCHED PORTS ?
8349      BNE     7$            ; BR IF NOT
8350      DEC     R1            ; GO TO THE NEXT DRIVE
8351      BPL     6$            ; CHECK NEXT DRIVE
8352      MOV     (SP)+, 2#PS    ; RESTORE THE PROCESSOR STATUS
8353      RESREG
8354      RTS     PC            ; RESTORE R0 - R5
8355      ; BYE-BYE

```

;DRIVE INITILIZATION ROUTINE

```

8356      ; THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
8357      ; AN RM03. IF IT IS A "READ-IN PRESET" IS ISSUED AND FM*22
8358      ; IS SET TO A "1". THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
8359      ; INSURE THEY ARE ALL ON A "1". AND DEPENDING ON THEIR STATE,
8360      ; DRVSTA IS SET TO THE PROPER CONDITION.
8361      :CALL
8362      :MOV      #DRVNUM,R1      ; DRIVE NUMBER TO R1
8363      :MOV      RMADR,R4      ; UNIBUS ADDRESS OF RH70/RM03 (RMCS1)
8364      :JSR      RO,DRVINT      ; CALLED BY A JSR
8365      :RETURN1      ; ERROR OCCURRED (PARITY)
8366      :RETURN2      ; NORMAL RETURN
8367
8368
8369      035050 010546      DRVINT: MOV      R5,-(SP)      ; SAVE R5
8370      035052 105061 034472      DULP:  CLRB     DRVSTA(R1)      ; START DRIVE STATUS AS OFFLINE
8371      035056 105061 034502      CLRB     DRVSTYP(R1)      ; CLEAR THE DRIVE TYPE INDICATOR
8372      035062 105061 034540      CLRB     ULDFLG(R1)      ; CLEAR THE UNLOAD FLAG
8373      035066 010164 000010      MOV      R1,RMCS2(R4)      ; SELECT A DRIVE
8374      035072 112764 000111 000000      MOVVB   #111,RMCS1(R4)      ; DO A DRIVE CLEAR COMMAND (& SEIZE DRIVE)
8375      035100 032764 010000 000010      BIT      #BIT12,RMCS2(R4)      ; NONEXISTENT DRIVE?
8376      035106 001403      BEQ      1$      ; NO---BRANCH
8377      035110 004737 042214      JSR      PC,SET.IE      ; GO SET "IE" WITHOUT A "TRE"
8378      035114 000507      BR      6$      ; LEAVE THIS ROUTINE
8379      035116 105061 034472      1$:  CLRB     DRVSTA(R1)      ; SET DRIVE STATUS TO OFFLINE
8380      035122 032764 004000 000000      BIT      #BIT11,RMCS1(R4)      ; SEE IF DRIVE AVAILABLE
8381      035130 001750      BEQ      DULP      ; BR IF DRIVE NOT AVAILABLE
8382      035132 004037 041534      JSR      RO,RD.RM      ; READ THE DRIVE TYPE REG.
8383      035136 000026      RMDT      8$      ; ERROR RETURN ADDRESS
8384      035140 035360      8$      MOV      (SP)+,R5      ; PUT DRIVE TYPE IN R5
8385      035142 012605      MOVVB   #4,DRVSTYP(R1)      ; SET RM03 INDICATOR
8386      035144 112761 000004 034502      CMP      #20024,R5      ; SINGLE PORT RM03 ?
8387      035152 022705 020024      BEQ      2$      ; BR IF YES
8388      035156 001420      CMP      #24024,R5      ; DUAL PORT RM03 ?
8389      035160 022705 024024      BEQ      2$      ; BR IF YES
8390      035164 001415      MOVVB   #5,DRVSTYP(R1)      ; SET RM02 INDICATOR
8391      035166 112761 000005 034502      CMP      #20025,R5      ; SINGLE PRPT RM02 ?
8392      035174 022705 020025      BEQ      2$      ; BRANCH IF SO
8393      035200 001407      CMP      #24025,R5      ; DUAL PORT RM02 ?
8394      035202 022705 024025      BEQ      2$      ; BRANCH IF SO
8395      035206 001404      MOVVB   #-1,DRVSTYP(R1)      ; SET INDICATOR TO 'OTHER'
8396      035210 112761 177777 034502      BR      6$      ; EXIT
8397      035216 000446      2$:  MOV      #121,-(SP)      ; DO A "READ-IN PRESET"
8398      035220 012746      JSR      RO,WRT.RM
8399      035224 004037 041710      RMCS1
8400      035230 000000      8$
8401      035232 035360      MOV      #BIT12,-(SP)      ; SET FMT22=1
8402      035234 012746      JSR      RO,WRT.RM
8403      035240 004037 041710      RMOF
8404      035244 000032      8$
8405      035246 035360      JSR      RO,RD.RM      ; READ RMDS
8406      035250 004037 041534      RMD5
8407      035254 000012      8$
8408      035256 035360      MOV      (SP)+,R5      ; AND SAVE IT IN R5
8409      035260 012605      BPL      4$      ; BRANCH IF ATA=0
8410      035262 100015      MOVVB   ATABIT(R1),RMAS(R4) ; CLEAR ATTENTION SIT
8411      035264 116164 034606 000016

```



```

8412 035272 004037 041534      JSR      RO,RO,RM      ;FIND OUT WHY ATA=1
8413 035276 000014      RMER1
8414 035300 035360      B$
8415 035302 006126      ROL      (SP)+      ;IS IT UNSAFE?
8416 035304 100004      BPL      4$      ;BR IF NOT
8417 035306 112761 177777 034472  MOVB    #-1,DRVSTA(R1) ;SET UNSAFE INDICATOR
8418 035314 000407      BR      6$      ;EXIT
8419 035316 005105      4$:  COM      R5      ;CHECK MOL, DPR, DRY, AND VV
8420 035320 042705 167077      BIC      #1<BIT12!BIT08!BIT07!BIT06>,R5 ;BRANCH IF MOL, DPR, DRY, OR VV IS CLEAR
8421 035324 001003      BNE      6$      ;SET DRIVE STATUS TO ONLINE
8422 035326 112761 000001 034472  MOVB    #1,DRVSTA(R1) ;STEP OVER THE ERROR RETURN
8423 035334 005720      6$:  TST      (RO)+      ;EXIT
8424 035336 000410      BR      8$      ;CHANGE INDEX TO ADDRESS WORDS
8425 035340 006301      7$:  ASL      R1      ;START 2 SEC TIMER
8426 035342 012761 060000 034564  MOV      #60000,TIMER(R1) ;RESTORE R1
8427 035350 006201      ASR      R1      ;SET PORT INITIALIZE INDICATOR
8428 035352 112761 177777 034512  MOVB    #-1,DPINT(R1) ;RESTORE R5
8429 035360 012605      8$:  MOV      (SP)+,R5 ;EXIT
8430 035362 000200      RTS      RO
8431
8432      ;REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
8433
8434      ;CALL
8435
8436      JSR      RO,@#RM03      ;CALL THE RM03 DRIVER
8437      PNTADR      ;ADDRESS OF POINTER OF DRIVES PARAMETER BLOCK
8438      RETURN1      ;RETURN HERE IF QUEUE IS FULL
8439      RETURN2      ;RETURN HERE IF REQUEST IS IN QUEUE OR THERE
8440      ;IS AN ERROR CONDITION
8441
8442 035364 013746 177776      RM03:  MOV      @#PS, -(SP)      ;SAVE THE CALLING STATUS
8443 035370 013737 034624 177776  MOV      RMVEC+2,@#PS      ;DON'T ALLOW ANY RM03 INTERRUPTS
8444 035376 112737 000001 034536  MOVB    #1,ACTDRV      ;SET "ACTIVE DRIVER" FLAG
8445 035404 104412      SAVREG      ;SAVE RO - R5
8446 035406 011002      MOV      (RO),R2      ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
8447 035410 005062 000016      CLR      16(R2)      ;CLEAR THE STATUS/ERROR INDICATOR
8448 035414 111201      MOVB    (R2),R1      ;PICKUP THE DRIVE NUMBER
8449 035416 013704 034620      MOV      RMADR,R4      ;UNIBUS ADDRESS OF RMCS1
8450 035422 105761 034472      TSTB    DRVSTA(R1)      ;CHECK DRIVES STATUS
8451 035426 003014      BGT      1$      ;BRANCH IF ONLINE
8452 035430 105761 034540      TSTB    ULDFLG(R1)      ;UNLOAD COMMAND IN QUEUE?
8453 035434 001036      BNE      3$      ;BRANCH IF YES
8454 035436 105761 034512      TSTB    DPINT(R1)      ;TRYING TO INIT THE DRIVE
8455 035442 001042      BNE      5$      ;BR IF YES
8456 035444 004037 035050      JSR      RO,DRVINT      ;GO INIT. THE DRIVE
8457 035450 000434      BR      4$      ;ERROR RETURN
8458 035452 105761 034472      TSTB    DRVSTA(R1)      ;IS DRIVE STATUS ONLINE?
8459 035456 003445      BLE      6$      ;BR IF NOT
8460 035460 105761 034522      1$:  TSTB    DPRQS(R1)      ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
8461 035464 001031      BNE      5$      ;BR IF YES
8462 035466 010164 000010      MOV      R1,RMCS2(R4)      ;SELECT THE DRIVE
8463 035472 004037 042656      JSR      RO,DRVQUE      ;PUT THIS REQUEST IN QUEUE
8464 035476 000460      BR      9$      ;QUEUE IS FULL
8465 035500 122762 000103 000002  CMPB    #103,2(R2)      ;IS THIS REQ. FOR AN UNLOAD?
8466 035506 001003      BNE      2$      ;BR IF NO
8467 035510 112761 177777 034540  MOVB    #-1,ULDFLG(R1) ;SET THE "UNLOAD IN QUEUE" FLAG

```

```

8468 035516 105761 034462      2$: TSTB   DRVACT(R1)      ;IS THIS DRIVE ACTIVE?
8469 035522 001043      BNE      8$          ;BR IF YES
8470 035524 004737 035656      JSR      PC,OPT      ;CALL THE OPTIMIZER
8471 035530 000440      BR       8$
8472 035532 012762 120000 000016 3$: MOV     #BIT15:BIT13,16(R2) ;SET THE "UNLOAD IN QUEUE" ERROR FLAG
8473 035540 000434      BR       8$          ;EXIT
8474 035542 004737 036736      4$: JSR      PC,C17      ;GO HANDLE THE PARITY ERROR
8475 035546 000431      BR       8$
8476 035550 004037 042656      5$: JSR      RD,DRVQUE     ;PUT REQUEST IN QUEUE
8477 035554 000431      BR       9$          ;QUEUE IS FULL
8478 035556 032714 000100      BIT      #BIT06,(R4)    ;IE BIT SET?
8479 035562 001023      BNE      8$          ;YES
8480 035564 004737 042214      JSR      PC,SET.IE    ;SET THE INTERRUPT
8481 035570 000420      BR       8$          ;RETURN
8482 035572 105761 034472      6$: TSTB   DRVSTA(R1)    ;SEE IF DRIVE OFFLINE OR UNSAFE
8483 035576 002412      BLT      7$          ;BR IF UNSAFE
8484 035600 012762 140000 000016  MOV     #BIT15:BIT14,16(R2) ;SET OFFLINE ERROR INDICATOR
8485 035606 105761 034502      TSTB   DRVTP(R1)    ;SEE IF OFFLINE OR NONEXISTENT
8486 035612 001007      BNE      8$          ;BR IF OFFLINE
8487 035614 012762 100002 000016  MOV     #BIT15:BIT01,16(R2) ;REPORT DRIVE NONEXISTENT
8488 035622 000403      BR       8$          ;GO TO EXIT
8489 035624 012762 110000 000016 7$: MOV     #BIT15:BIT12,16(R2) ;DRIVE IS UNSAFE
8490 035632 104413      8$: RESREG      ;RESTORE R0 - R5
8491 035634 005720      TST      (R0)+      ;SETUP FOR NORMAL RETURN
8492 035636 000401      BR       10$     ;FINISH UP, THEN EXIT
8493 035640 104413      9$: RESREG      ;RESTORE R0 - R5
8494 035642 005720      10$: TST      (R0)+      ;CORRECT THE RETURN ADDRESS
8495 035644 105037 034536      CLRB     ACTDRV     ;CLEAR "ACTIVE DRIVER" FLAG
8496 035650 012637 177776      MOV     (SP)+,2#PS    ;RETURN "PS" TO USER LEVEL
8497 035654 000200      RTS      R0      ;RETURN TO CALLER
8498
8499 ;OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
8500 ;CALL
8501 ;
8502 ;
8503 ;
8504 ;
8505 035656 104412      OPT: SAVREG      ;SAVE R0 - R5
8506 035660 013746 177776      MOV     2#PS, -(SP)    ;SAVE PROC. STATUS
8507 035664 146137 034606 034534  BICB     ATABIT(R1),SRCHWT ;CLEAR LA SEARCH FLAG
8508 035672 105061 034522      CLRB     DPRQS(R1)    ;RESET THE PORT REQ FLAG ****
8509 035676 004737 042732      JSR      PC,GETREQ    ;GET "DPB" POINTER OF REQUEST
8510 035702 005702      TST      R2          ;IS THERE A REQUEST IN QUEUE?
8511 035704 001472      BEQ      7$          ;NO--BRANCH TO EXIT
8512 035706 010164 000010      MOV     R1,RMCS2(R4) ;LOAD THE DRIVE ADDRESS *****
8513 035712 012764 000111 000000  MOV     #11,RMCS1(R4) ;CLEAR THE DRIVE
8514 035720 032764 004000 000000  BIT      #BIT11,RMCS1(R4) ;DVA SET?
8515 035726 001446      BEQ      5$          ;TO PROT REQUEST, IF NOT
8516 035730 105761 034472      10$: TSTB   DRVSTA(R1)    ;IS DRIVE ONLINE?
8517 035734 003014      BGT      1$          ;YES--BRANCH
8518 035736 004737 042754      JSR      PC,POPQUE    ;NO--REMOVE REQUEST FROM QUEUE
8519 035742 012762 140000 000016  MOV     #BIT15:BIT14,16(R2) ;SET OFFLINE STATUS/ERROR INDICATOR
8520 035750 105761 034472      TSTB   DRVSTA(R1)    ;IS DRIVE UNSAFE?
8521 035754 100053      BPL      8$          ;BR TO EXIT IF NOT
8522 035756 012762 110000 000016  MOV     #BIT15:BIT12,16(R2) ;SET UNSAFE STATUS ERROR INDICATOR
8523 035764 000447      BR       8$          ;BRANCH TO EXIT

```


8524	035766			15:	MOV	#111, -(SP)	: LOAD COMMAND ONTO THE STACK
8525				:	JSR	RO, WRT.RM	: LOAD THE REGISTER
8526				:	RMCS1		: REGISTER INCREMENT
8527				:	BS		: ERROR RETURN ADDRESS
8528				:	BIT	#BIT11, (R4)	: DRIVE AVAILABLE ?
8529				:	REQ	95	: BR IF NOT
8530				:	CMPB	#150, 2(R2)	: IS THE REQUEST FOR I/O?
8531	035766	122762	000150	000002	BLT	25	: YES--BRANCH
8532	035774	002403			JSR	PC, CI4	: CALL THE COMMAND INITIATOR
8533	035776	004737	036322		BR	85	: BRANCH TO EXIT
8534	036002	000440			TST	DTUW	: DATA TRANSFER UNDERWAY?
8535	036004	005737	034604	25:	BGE	45	: YES--GO START A SEARCH
8536	036010	002012			TST	SEEKFG	: DO IMPLIED SEEKS?
8537	036012	005737	034562		BMI	35	: YES---BRANCH
8538	036016	100404			JSR	RO, LA	: NO--DO LOOK AHEAD
8539	036020	004037	037274		BR	85	: RETURN HERE ON A PARITY ERROR
8540	036024	000427			BR	45	: GO START A SEARCH
8541	036026	000403			JSR	PC, CI1	: START A DATA TRANSFER
8542	036030	004737	036114	35:	BR	85	
8543	036034	000423			JSR	PC, CI3	: START A SEARCH
8544	036036	004737	036222	45:	BR	85	: GO TO THE EXIT
8545	036042	000420			MOVB	#-1, DPRQS(R1)	: SET PORT REQUEST INDICATOR
8546	036044	112761	177777	55:	MOV	R1, R3	: SET UP TO ADDRESS WORDS
8547	036052	010103			ASL	R3	: CONVERT TO WORD INDEX
8548	036054	006303			MOV	#60000, TIMER(R3)	: START 10 SEC TIMER
8549	036056	012763	060000	034564	BR	75	: EXIT
8550	036064	000402			JSR	PC, CI7	: PROCESS THE PARITY ERROR
8551	036066	004737	036736	65:	BIT	#BIT06, (R4)	: SEE IF 'IE' ALREADY SET
8552	036072	032714	000100	75:	BNE	85	: BR IF SET
8553	036076	001002			JSR	PC, SET.IE	: SET "IE" WITHOUT A "TRE"
8554	036100	004737	042214		MOV	(SP)+, 2#PS	: RESTORE PROC. STATUS
8555	036104	012637	177776	85:	RESREG		: RESTORE RO - R5
8556	036110	104413			RTS	PC	
8557	036112	000207					
8558							
8559							
8560							
8561							
8562							
8563							
8564							
8565							
8566							
8567							
8568							
8569							
8570	036114	004737	042754	CI1:	JSR	PC, POPQUE	: REMOVE REQUEST FROM "DRIVES WAIT" QUEUE
8571	036120	010237	034532		MOV	R2, TRANSW	: PUT REQ. IN TRANSFER WAIT QUEUE
8572	036124	010203			MOV	R2, R3	: DPB ADDRESS TO R3
8573	036126	013704	034620		MOV	RMADR, R4	: RMCS1 ADDRESS
8574	036132	010164	000010		MOV	R1, RMCS2(R4)	: SELECT DRIVE
8575	036136	062703	000004		ADD	#4, R3	: DESIRED WORD COUNT
8576	036142	062704	000002		ADD	#2, R4	: RMWC ADDRESS
8577	036146	012324			MOV	(R3)+, (R4)+	: LOAD WORD COUNT
8578	036150	012324			MOV	(R3)+, (R4)+	: LOAD BUFFER ADDRESS
8579	036152	012346			MOV	(R3)+, -(SP)	: LOAD SECTOR AND TRACK

CZRM880 RM03/2 PERF EXER
CZRM88.P11 21-NOV-77 15:34

MACY11 30(1046) 22-NOV-77 10:06 PAGE 160
SINGLE/DUAL PORT RH70/RM03 DRIVER (REV 5.1)-24-AUG-77

SEQ 0159

8580	036154	004037	041710		JSR	RO,WRT.RM	;CALL THE LOAD(WRITE) ROUTINE
8581	036160	000006			RMDA		;INDEX OF REGISTER TO LOAD
8582	036162	036736			CI7		;ERROR RETURN ADDRESS
8583	036164	012346			MOV	(R3)+-(SP)	;LOAD CYLINDER ADDRESS
8584	036166	004037	041710		JSR	RO,WRT.RM	
8585	036172	000034			RMDC		
8586	036174	036736			CI7		
8587	036176	016246	000002		MOV	2(R2)-(SP)	;LOAD "COMMAND+GO", "A17&A16", AND "PSEL"
8588	036202	004037	041710		JSR	RO,WRT.RM	
8589	036206	000000			RMCS1		
8590	036210	036736			CI7		
8591	036212	010137	034604		MOV	R1,DTUW	;SET "DATA TRANSFER UNDERWAY"
8592	036216	000137	036700		JMP	CI5	
8593	036222	013704	034620		MOV	RMADR,R4	;RMCS1 ADDRESS
8594	036226	010164	000010		MOV	R1,RMCS2(R4)	;SELECT DRIVE
8595	036232	016246	000012		MOV	12(R2)-(SP)	;DESIRED CYLINDER ADDRESS
8596	036236	004037	041710		JSR	RO,WRT.RM	
8597	036242	000034			RMDC		
8598	036244	036736			CI7		
8599	036246	116203	000010		MOV	10(R2),R3	;PICKUP SECTOR ADDRESS
8600					SUB	MXWNOU,R3	;BACKUP BY MAX. SEARCH FOR I/O WINDOW
8601					BGE	1\$	
8602					ADD	#32,R3	
8603	036252	010246			MOV	R3-(SP)	;COMBINE THE ADJUSTED SECTOR WITH
8604	036254	042716	177740		BIC	#177740,(SP)	;CHOP OF ALL EXTENDED BITS, IF ANY
8605	036260	116266	000011	000001	MOV	11(R2),1(SP)	;THE DESIRED TRACK
8606	036266	004037	041710		JSR	RO,WRT.RM	;LOAD DESIRED TRACK & SECTOR
8607	036272	000006			RMDA		
8608	036274	036736			CI7		
8609	036276	012746	000131		MOV	#131-(SP)	;START A SEARCH
8610	036302	004037	041710		JSR	RO,WRT.RM	
8611	036306	000000			RMCS1		
8612	036310	036736			CI7		
8613	036312	156137	034606	034534	BISB	ATABIT(R1),SRCHWT	;SET "SEARCH WAIT" KEY
8614	036320	000567			BR	CI5	
8615	036322	013704	034620		MOV	RMADR,R4	;RMCS1 ADDRESS
8616	036326	010164	000010		MOV	R1,RMCS2(R4)	;SELECT DRIVE
8617	036332	116203	000002		MOV	2(R2),R3	;PICKUP THE REQUESTED COMMAND
8618	036336	122703	000131		CMPB	#131,R3	;IS IT A SEARCH COMMAND?
8619	036342	001007			BNE	1\$	;BRANCH IF NO
8620	036344	016246	000010		MOV	10(R2)-(SP)	;LOAD DESIRED TRACK & SECTOR
8621	036350	004037	041710		JSR	RO,WRT.RM	
8622	036354	000006			RMDA		
8623	036356	036736			CI7		
8624	036360	000403			BR	2\$	;GO LOAD CYLINDER
8625	036362	122703	000105		CMPB	#105,R3	;IS IT A SEEK COMMAND
8626	036366	001007			BNE	3\$	;BRANCH IF NO
8627	036370	016246	000012		MOV	12(R2)-(SP)	;LOAD DESIRED CYLINDER
8628	036374	004037	041710		JSR	RO,WRT.RM	
8629	036400	000034			RMDC		
8630	036402	036736			CI7		
8631	036404	000546			BR	CI6	
8632	036406	122703	000115		CMPB	#115,R3	;IS IT AN "OFFSET" COMMAND?
8633	036412	001013			BNE	4\$	;BR IF NO
8634	036414	004037	041534		JSR	RO,RO.PM	;MERGE THE OFFSET VALUE INTO RMOF
8635	036420	000032			RMOF		;BUT DON'T CHANGE THE UPPER

8636	036422	036736			CI7		
8637	036424	116216	000001		MOVB	1(R2), (SP)	; BYTE WHEN LOADING THE
8638	036430	004037	041710		JSR	RO, WRT.RM	; REGISTER (RMOF)
8639	036434	000032			RMOF		
8640	036436	036736			CI7		
8641	036440	000530			BR	CI6	; GO START THE COMMAND
8642	036442	122703	000107	4\$:	CMPB	#107, R3	; IS IT A "RECALIBRATE" COMMAND?
8643	036446	001525			BEQ	CI6	; BRANCH IF YES
8644	036450	122703	000117		CMPB	#117, R3	; IS IT A RETURN TO CENTER?
8645	036454	001522			BEQ	CI6	; BRANCH IF YES
8646	036456	122703	000103		CMPB	#103, R3	; IS IT AN "UNLOAD" COMMAND?
8647	036462	001016			BNE	5\$	; BRANCH IF NO
8648	036464	112761	000001	034462	MOVB	#1, DRVACT(R1)	; SET THE DRIVE ACTIVE INDICATOR
8649	036472	105061	034472		CLRB	DRVSTA(R1)	; PUT DRIVE STATUS TO OFFLINE
8650	036476	112761	000001	034540	MOVB	#1, ULDFLG(R1)	; SET "UNLOAD IN PROGRESS" FLAG
8651	036504	010346			MOV	R3, -(SP)	; START THE "UNLOAD" COMMAND
8652	036506	004037	041710		JSR	RO, WRT.RM	
8653	036512	000000			RMCS1		
8654	036514	036736			CI7		
8655	036516	000207			RTS	PC	; RETURN TO USER
8656	036520	122703	000143	5\$:	CMPB	#143, R3	; IS IT A "SET FORMAT" COMMAND?
8657	036524	001014			BNE	6\$	; BRANCH IF NO
8658	036526	004037	041534		JSR	RO, RD.RM	; READ THE OFFSET REGISTER
8659	036532	000032			RMOF		
8660	036534	036736			CI7		
8661	036536	116266	000001	000001	MOVB	1(R2), 1(SP)	; COMBINE "FMT22" "ECI" AND "HCI"
8662	036544	004037	041710		JSR	RO, WRT.RM	; LOAD "FMT22", "ECI", AND/OR "HCI".
8663	036550	000032			RMOF		
8664	036552	036736			CI7		
8665	036554	000436			BR	12\$	
8666	036556	122703	000141	6\$:	CMPB	#141, R3	; IS IT A "GET REGISTER" COMMAND?
8667	036562	001023			BNE	10\$	; BRANCH IF NO
8668	036564	016203	000006	7\$:	MOV	6(R2), R3	; POINTS TO 1ST ADDRESS OF WHERE
8669							; TO PUT THE REGISTER(S)
8670	036570	116237	000010	036606	MOVB	10(R2), 9\$	; INIT. THE INDEX FOR THE FIRST REG.
8671	036576	116205	000011		MOVB	11(R2), R5	; INDEX OF LAST REG. TO MOVE
8672	036602	004037	041534	8\$:	JSR	RO, RD.RM	; READ RH70/RM03 REGISTER
8673	036606	000000		9\$:	RMCS1		; INDEX OF REG. TO READ
8674	036610	036736			CI7		
8675	036612	012623			MOV	(SP)+, (R3)+	; GET THE CONTENTS OF RH70/RM03 REG.
8676	036614	023705	036606		CMP	9\$, R5	; LAST REG. BEEN READ?
8677	036620	001414			BEQ	12\$	; GET OUT IF YES
8678	036622	062737	000002	036606	ADD	#2, 9\$	; INCREASE THE INDEX BY 2
8679	036630	000764			BR	8\$	; LOOP--MORE TO READ
8680	036632	122703	000145	10\$:	CMPB	#145, R3	; IS IT A "SELECT DRIVE" COMMAND?
8681	036636	001405			BEQ	12\$	; BRANCH IF YES
8682	036640	010346		11\$:	MOV	R3, -(SP)	; LOAD THE COMMAND
8683	036642	004037	041710		JSR	RO, WRT.RM	
8684	036646	000000			RMCS1		
8685	036650	036736			CI7		
8686	036652	004737	042754	12\$:	JSR	PC, POPQUE	; REMOVE REQ. FROM QUEUE
8687	036656	052762	000200	000016	BIS	#8107, 16(R2)	; SET THE "DONE" BIT
8688	036664	005737	034560		TST	SAVEFG	; SAVE THE RH70/RM03 REGISTERS
8689	036670	100002			BPL	13\$	; BRANCH IF NO
8690	036672	004737	042076		JSR	PC, SVRH70	; YES--GO SAVE THE REGISTERS
8691	036676	000207		13\$:	RTS	PC	; RETURN TO USER

C2RM880 RM03 2 PERF EXER
C2RM88.P11 21-NOV-77 15:34

MACY11 30(1046) 22-NOV-77 10:06 PAGE 162
SINGLE/DUAL PORT RH70/RM03 DRIVER (REV 5.1)-24-AUG-77

SEQ 0161

8692	036700	006301			C15:	ASL	R1	
8693	036702	012761	060000	034564		MOV	#60000,TIMER(R1)	;SET A ONE SECOND TIMER
8694	036710	006201				ASR	R1	
8695	036712	112761	000001	034462		MOV8	#1,DRVACT(R1)	;SET THE DRIVE ACTIVE
8696	036720	000207				RTS	PC	;RETURN TO THE USER
8697	036722	010346			C16:	MOV	R3,-(SP)	;LOAD THE COMMAND
8698	036724	004037	041710			JSR	RO,WRT.RM	
8699	036730	000000				RMCS1		
8700	036732	036736				CI7		
8701	036734	000761				BR	C15	
8702	036736	032764	010000	000010	C17:	BIT	#BIT12,RMCS2(R4)	;DRIVE NON-EXISTENT ?
8703						BNE	C18	;BR IF YES
8704	036744	005702			1s:	TST	R2	;ANYTHING IN QUEUE ?
8705						BEQ	CI7B	;BR IF NOT
8706	036746	001001				BNE	2s	;BRANCH IF QUEUE IS THERE
8707	036750	000207				RTS	PC	;OTHERWISE EXIT
8708	036752	012762	104000	000016	2s:	MOV	#BIT15:BIT11,16(R2)	;SET "PARITY" ERROR INDICATOR
8709						JSR	PC,SVRH70	;GO SAVE THE RH70/RM03 REGISTERS
8710	036760	012746	000111		CI7B:	MOV	#111,-(SP)	;DO A "DRIVE CLEAR"
8711	036764	004037	041710			JSR	RO,WRT.RM	
8712	036770	000000				RMCS1		
8713	036772	037036				CI8		
8714	036774	004737	042636		2s:	JSR	PC,EMPTYQ	;EMPTY THE QUEUE
8715	037000	105061	034522			CLRB	DPRQS(R1)	;CLEAR THE PORT REQUEST FLAG
8716	037004	105061	034540			CLRB	ULDFLG(R1)	;CLEAR THE UNLOAD IN QUEUE FLAG
8717	037010	105061	034462			CLRB	DRVACT(R1)	;DRIVE IS IDLE
8718	037014	020237	034532			CMP	R2,TRNSWT	;IF THIS DRIVE HAD AN I/O REQUEST
8719						CMP	R1,DTUW	;IF THIS DRIVE HAD AN I/O REQUEST
8720	037020	001005				BNE	1s	;IN PROGRESS CLEAR ALL OF THE FLAGS
8721	037022	005037	034532			CLR	TRNSWT	
8722	037026	012737	177777	034604		MOV	#-1,DTUW	
8723	037034	000207			1s:	RTS	PC	
8724	037036	104412			CI8:	SAVREG		;SAVE R0 - R5
8725	037040	032764	010000	000010		BIT	#BIT12,RMCS2(R4)	;IS 'NED' SET ?
8726						BNE	1s	;BR IF YES
8727	037046	005001				CLR	R1	
8728	037050	005003				CLR	R3	
8729	037052	105761	034462		1s:	TSTB	DRVACT(R1)	;DRIVE ACTIVE?
8730						BEQ	5s	;BRANCH IF NO
8731	037056	001003				BNE	22s	;BRANCH IF IN ACTIVE
8732	037060	105761	034522			TSTB	DPRQS(R1)	;PORT REQUEST
8733	037064	001443				BEQ	5s	;BRANCH IF NOT
8734	037066	013702	034532		22s:	MOV	TRNSWT,R2	;GET THE "TRANSFER WAIT" QUEUE
8735	037072	020137	034604			CMP	R1,DTJW	;DID THIS DRIVE HAVE AN I/O IN PROGRESS?
8736	037076	001402				BEQ	2s	;BRANCH IF YES
8737	037100	004737	042732			JSR	PC,GETREQ	;GET THE DPB POINTER
8738	037104	005702			2s:	TST	R2	;QUEUE ENTRY FOR DRIVE ?
8739	037106	001413				BEQ	4s	;BR IF NOT
8740	037110	032764	010000	000010		BIT	#BIT12,RMCS2(R4)	; 'NED' SET ?
8741	037116	001404				BEQ	3s	;BR IF NOT
8742	037120	012762	100002	000016		MOV	#BIT15:BIT01,16(R2)	;SET 'DRIVE NON-EXISTENT' INDICATOR
8743	037126	000403				BR	4s	;CONTINUE
8744	037130	012762	102000	000016	3s:	MOV	#BIT15:BIT10,16(R2)	;SET "NON-CLEARABLE PARITY" ERROR INDICATOR
8745						JSR	PC,SVRH70	;SAVE RH70/RM03 REGISTERS
8746	037136	012763	177777	034564	4s:	MOV	#-1,TIMER(R3)	;STOP THE TIMER
8747	037144	105061	034462			CLRB	DRVACT(R1)	;SET "DRIVE ACTIVE" TO IDLE

```

8748 037150 105061 034522      CLRB   DPRQS(R1)      ; CLEAR PORT REQUEST FLAG
8749 037154 020137 034604      CMP    R1,DTUW      ; IS THIS DRIVE SETUP FOR A TRANSFER
8750 037160 001005              BNE     5$              ; BR IF NOT
8751 037162 012737 177777 034604  MOV    #-1,DTUW      ; RESET THE INDICATOR
8752 037170 005037 034532      CLR     TRANSWT      ; CLEAR THE TRANSFER QUEUE
8753 037174 105061 034540      CLRB   ULDFLG(R1)      ; CLEAR UNLOAD FLAG
8754 037200 032764 010000 000010 5$: BIT    #BIT12,RMCS2(R4) ; 'NED' SET ?
8755              ; BNE     6$              ; BR IF YES
8756 037206 005201              INC     R1              ; MOVE TO THE NEXT DRIVE
8757 037210 062703 000002      ADD     #2,R3
8758 037214 042701 177770      BIC     #1<7,R1
8759 037220 001314              BNE     1$              ; BRANCH IF MORE DRIVES
8760 037222 012737 177777 034604  MOV    #-1,DTUW      ; NO DATA TRANSFERS UNDERWAY
8761 037230 005037 034532      CLR     TRANSWT      ; CLEAR THE 'TRANSFER WAIT' QUEUE
8762 037234 004737 042560      JSR     PC,CLRQUE      ; CLEAR ALL OF THE REQUEST QUEUES
8763 037240 012764 000040 000010  MOV    #BIT05,RMCS2(R4) ; DO A MASSBUS INIT.
8764 037246 000406              BR      7$              ; CONTINUE
8765 037250 004737 042636      JSR     PC,EMPTYQ      ; CLEAR THE DRIVE'S QUEUE
8766 037254 105061 034472      CLRB   DRVSTA(R1)      ; SET DRIVE TO OFFLINE
8767 037260 105061 034502      CLRB   DRVTP(R1)      ; CLEAR THE DRIVE TYPE INDICATOR
8768 037264 004737 042214      JSR     PC,SET.IE      ; SET "IE" WITHOUT "TRE"
8769 037270 104413              RESREG
8770 037272 000207              RTS     PC              ; RESTORE R0 - R5
8771              ; RETURN
8772              ; LOOK AHEAD ROUTINE
8773              ; CALL
8774              ;
8775              MOV    #DRVNUM,R1      ; DRIVE NUMBER
8776              MOV    #DPB,R2         ; POINT TO DPB
8777              JSR     R0,LA          ; GO CHECK THE WINDOW
8778              RETURN1              ; ERROR RETURN
8779              RETURN2              ; START A SEARCH
8780              RETURN3              ; START A DATA TRANSFER
8781              ;
8782 037274 013704 034620      LA:     MOV    RMADR,R4      ; GET RMCS1'S ADDRESS
8783 037300 010164 000010      MOV    R1,RMCS2(R4)      ; SELECT DRIVE
8784 037304 004037 041534      JSR     R0,RD.RM          ; READ DRIVE STATUS
8785 037310 000012              RMDS     4$              ;
8786 037312 037442              4$              ; ERROR RETURN ADDRESS
8787 037314 042716 157577      BIC     #*C020200,(SP) ; ON CYLINDER ?
8788 037320 022726 000200      CMP     #200,(SP)+      ; PIP=0,DRY=1?
8789 037324 001044              BNE     3$              ; NO
8790 037326 105261 034550      INCB    LACNT(R1)      ; INCREMENT THE LOOK AHEAD COUNT
8791 037332 126137 034550 034626  CMPB   LACNT(R1),MXLACT ; EXCEED MAX?
8792 037340 003033              BGT     2$              ; BRANCH IF YES
8793 037342 116203 000010      MOVB   10(R2),R3      ; GET DESIRED SECTOR ADDRESS AND
8794 037346 000303              SWAB   R3              ; MULT. BY 64--ALIGN WITH
8795 037350 006203              ASR     R3              ; LOOK AHEAD REGISTER
8796 037352 006203              ASR     R3              ;
8797 037354 012737 000340 177776  MOV    #340,2#PS      ; PRIORITY LEVEL "7"
8798 037362 004037 041534      JSR     R0,RD.RM          ; READ LOOK AHEAD REGISTER
8799 037366 000020              RMLA   4$              ;
8800 037370 037442              4$              ;
8801 037372 021664 000020      CMP     (SP),RMLA(R4) ; CORRECT LA NUMBER ?
8802 037376 001402              BEQ     7$              ; YES
8803 037400 005726              *ST     (SP)+          ; NO,CLEAR STACK

```

H13

C2RM880 RM03/2 PERF EXER
C2RM88.P11 21-NOV-77 15:34

MACY11 30(1046) 22-NOV-77 10:06 PAGE 164
SINGLE DUAL PORT RH70/RM03 DRIVER (REV 5.1)-24-AUG-77

SEQ 0163

```

8804 037402 000415
8805 037404 162603
8806 037406 002002
8807 037410 062703 004000
8808 037414 023703 034630
8809 037420 002406
8810 037422 023703 034632
8811 037426 002003
8812 037430 105061 034550
8813 037434 005720
8814 037436 005720
8815 037440 000402
8816 037442 004737 036736
8817 037446 000200
8818
8819
8820
8821 037450 112737 000001 034536
8822 037456 104412
8823 037460 013704 034620
8824 037464 013701 034604
8825 037470 002403
8826 037472 004737 037514
8827 037476 000402
8828 037500 004737 037770
8829 037504 104413
8830 037506 105037 034536
8831 037512 000002
8832
8833
8834
8835 037514 105061 034462
8836 037520 012737 177777 034604
8837 037526 006301
8838 037530 012761 177777 034564
8839 037536 006201
8840 037540 013702 034532
8841 037544 005037 034532
8842 037550 052762 000200 000016
8843 037556 010164 000010
8844 037562 004037 041534
8845 037566 000000
8846 037570 036736
8847 037572 006126
8848 037574 100421
8849 037576 005737 034560
8850 037602 100002
8851 037604 004737 042076
8852 037610 004737 037670
8853 037614 004737 042732
8854 037620 005702
8855 037622 001403
8856 037624 004737 035656
8857 037630 000457
8858
8859 037632 012714 000113

7$: BR 3$
SUB (SP)+,R3 ;CALCULATE THE DELTA
BGE 1$
ADD #32,*64,R3 ;MAKE THE DELTA POSITIVE
1$: CMP MXDLTA,R3 ;CHECK THE DELTA TO SEE
BLT 3$ ;IF IT IS WITHIN THE
CMP MNDLTA,R3 ;WINDOW---IF YES, ZERO
BGE 3$ ;THE LOOK AHEAD COUNT
2$: CLRB LACNT(R1) ;AND TAKE THE I/O EXIT
TST (R0)+
3$: TST (R0)+ ;ADJUST THE RETURN ADDRESS
BR 5$ ;EXIT
4$: JSR PC,C17 ;PROCESS THE ERROR
5$: RTS R0 ;RETURN

; INTERRUPT SERVICE ROUTINE
ISR: MOVB #1,ACTDRV ;SET "ACTIVE DRIVER" FLAG
SAVREG ;SAVE R0 - R5
MOV RMADR,R4 ;ADDRESS OF RHSCS1
MOV DTUW,R1 ;GET "DATA TRANSFER UNDERWAY" INDICATOR
BLT 1$ ;BRANCH IF NO DATA TRANSFER UNDERWAY
JSR PC,TD ;CALL TRANSFER DONE
BR 2$ ;EXIT
1$: JSR PC,SC ;CALL SPECIAL CONDITIONS
2$: RESREG ;RESTORE R0 - R5
CLRB ACTDRV ;CLEAR "ACTIVE DRIVER" FLAG
RTI ;RETURN

; TRANSFER DONE ROUTINE
TD: CLRB DRVACT(R1) ;SET DRIVE ACTIVE INDICATOR TO IDLE
MOV #1,DTUW ;NO DATA TRANSFERS UNDERWAY
ASL R1
MOV #1,TIMER(R1) ;CANCEL TIMEOUT
ASR R1
MOV TRANSW,R2 ;GET "DPB" ADDRESS FROM THE
CLR TRANSW ;TRANSFER WAIT QUEUE--CLEAR QUEUE
BIS #BIT07,16(R2) ;SET DONE
MOV R1,RMCS2(R4) ;SELECT THE DRIVE
JSR R0,RD.RM ;TRANSFER ERROR(TRE=1)?
RMCS1
C17
ROL (SP)+
3$: BR IF YES
TST SAVEFG ;SAVE THE RH70/RM03 REGISTERS?
BPL 1$ ;BRANCH IF NO
JSR PC,SVRH70 ;YES--SAVE THE REGISTERS
1$: JSR PC,WC,HK ;SEE IF WRITE CHECK TO BE PUT IN QUEUE
JSR PC,GETREG ;GET DPB POINTER
TST R2 ;ENTRY FOR DRIVE ?
BEQ 2$ ;BR IF NOT
JSR PC,OPT ;CALL OPTIMIZER
BR SC ;CHECK OTHER DRIVES
2$: ;THE RELEASE DRIVE COMMAND IS FORCED TO ENTER FOR DUAL PORT OPERATION
MOV #113,(R4) ;RELEASE THE DRIVE

```



```

8860 037636 000454          BR      SC      ;CHECK FOR OTHER DRIVES
8861 037640 052762 100100 000016 3$: BIS      #BIT15:BIT06.16(R2) ;SET DATA ERROR FLAG
8862 037646 004737 042636      JSR      PC,EMPTYQ ;EMPTY THE "DRIVE'S WAIT" QUEUE
8863 037652 004737 042076      JSR      PC,SVRH70 ;SAVE THE RH70/RM03 REGISTERS
8864 037656 012714 040111      MOV      #40111,(R4) ;ISSUE A "DRIVE CLEAR"
8865 037662 012714 000113      MOV      #113,(R4) ;ISSUE A RELEASE TO THE DRIVE
8866 037666 000440          BR      SC      ;CHECK FOR OTHER DRIVES
8867
8868
8869 037670 122762 000002 000024 WC.HK: CMPB      #2,$CODE(R2) ;LAST OPERATION WRITE DATA ?
8870 037676 001404          BEQ      1$ ;BR IF IT WAS
8871 037700 122762 000003 000024      CMPB      #3,$CODE(R2) ;LAST OPERATION WRITE HEADER & DATA ?
8872 037706 001027          BNE      2$ ;BR IF NOT
8873 037710 004037 042656      JSR      R0,DRVQUE ;PUT THE OPERATION IN THE QUEUE
8874 037714 000424          BR      2$ ;QUEUE IS FULL
8875 037716 005062 000016      CLR      16(R2) ;CLEAR 'DONE' BIT IN DPB
8876 037722 116262 000234 000027      MOVB      $RMCS1(R2),$PREV0(R2) ;SAVE WRITE OPERATION CODE
8877 037730 016262 000012 000034      MOV      $CYL(R2),$PREVA+2(R2) ;SAVE CYLINDER
8878 037736 016262 000010 000032      MOV      $SEC(R2),$PREVA(R2) ;SAVE SECTOR AND TRACK ADDRESSES
8879 037744 142762 000002 000024      BICB      #2,$CODE(R2) ;CHANGE WRITE TO CHECK
8880 037752 142762 000020 000002      BICB      #20,$COMND(R2) ;CHANGE DRIVER CODE TO WRITE CHECK
8881 037760 152762 000010 000002      BISB      #10,$COMND(R2) ;FINISH CHANGING CODE TO WRITE CHECK
8882 037766 000207          2$: RTS      PC ;EXIT
8883
8884 ;SPECIAL CONDITION ROUTINE
8885
8886 037770 116403 000016          SC:      MOVB      RMA5(R4),R3 ;READ "RMA5"
8887 037774 001012          BNE      2$ ;BRANCH IF ANY 'ATA' BITS SET
8888 037776 004037 041534          JSR      R0,RD.RM ;READ CONTROL AND STATUS REGISTER
8889 040002 000000          ;
8890          ;
8891 040004 040020          ;
8892 040006 106126          ;
8893 040010 100403          ;
8894 040012 104001          ;
8895 040014 004737 042214          JSR      PC,SET.IE ;EXIT IF FAIL TO READ
8896 040020 000207          1$: RTS      PC ;IS "IE"=1?
8897 040022 005046          2$: CLR      -(SP) ;YES, NO DRIVES TO CHECK
8898 040024 110316          MOVB      R3,(SP) ;REPORT AN ILLEGAL INTERRUPT
8899 040026 012703 000001          MOV      #1,R3 ;SET INTERRUPT ENABLE
8900 040032 005001          CLR      R1 ;RETURN
8901 040034 030316          SC3: BIT      R3,(SP) ;PROCESS ALL DRIVES THAT HAVE
8902 040036 001005          BNE      SC5 ;AN "ATA"=1
8903 040040 005201          SC4: INC      R1 ;ATA=1?
8904 040042 106303          ASLB      R3 ;YES--BRANCH
8905 040044 001373          BNE      SC3 ;MOVE TO THE NEXT DRIVE
8906 040046 005726          TST      (SP)+ ;BRANCH IF MORE TO CHECK?
8907 040050 000207          RTS      PC ;CLEAN OFF THE STACK
8908 040052 105761 034512          SC5: TSTB      DPINT(R1) ;RETURN TO USER
8909 040056 001402          BEQ      1$ ;INITIALIZING THE DRIVE ?
8910 040060 000137 040764          JMP      SC13 ;BR IF NOT
8911 040064 105761 034522          1$: TSTB      DPRQS(R1) ;PROCESS THE DRIVE
8912 040070 001402          BEQ      2$ ;PORT REQUEST OUTSTANDING ?
8913 040072 000137 040764          JMP      SC13 ;BR IF NOT
8914 040076 105761 034472          2$: TSTB      DRVSTA(R1) ;START THE OUTSTANDING COMMAND
8915 040102 003025          BGT      5$ ;CHECK THE DRIVE STATUS
          ;BRANCH IF ONLINE

```

8916	040104	105761	034540		TSTB	ULDFLG(R1)	; UNLOAD IN PROGRESS?
8917	040110	003422			BLE	5\$	; BRANCH IF NOT
8918	040112	004737	042732		JSR	PC,GETREQ	; GET DPB POINTER
8919	040116	004737	042076		JSR	PC,SVRH70	; SAVE THE RH70/RM03 REGISTERS
8920	040122	004737	040722		JSR	PC,SC12	; SAVE RMD5, RMER1, RMER2, AND RMMR2
8921							; ALSO DO A DRIVE INIT (DRVINT)
8922	040126	105761	034472		TSTB	DRVSTA(R1)	; DID DRIVE COME ONLINE?
8923	040132	003416			BLE	6\$	; NO---BRANCH
8924	040134	032737	040000	034452	BIT	#BIT14,RMERRS	; WAS THERE AN ERROR?
8925	040142	001002			BNE	3\$	; BR IF ERROR
8926	040144	000137	040612		JMP	SC11	; NO ERROR
8927	040150	013705	034454	3\$:	MOV	RMERRS+2,R5	; YES -- PICKUP RMER1 AND
8928	040154	000500			BR	SC6A	; GO PROCESS THE ERROR
8929	040156	105761	034462	5\$:	TSTB	DRVACT(R1)	; DRIVE ACTIVE WITH COMMAND OR ERROR RECOVERY ?
8930	040162	001027			BNE	SC6	; BR IF EITHER
8931	040164	004737	040722		JSR	PC,SC12	; SAVE RMD5, RMER1, RMER2, AND RMMR2
8932							; ALSO DO A DRVINT
8933	040170	105761	034512	6\$:	TSTB	DPINT(R1)	; TRYING TO INIT THE DRIVE ?
8934	040174	001321			BNE	SC4	; BR IF YES, CHECK ON MORE DRIVES
8935	040176	105761	034472		TSTB	DRVSTA(R1)	; CHECK ON DRIVE'S STATUS
8936	040202	100412			BMI	7\$	; BR IF UNSAFE
8937	040204	032737	020000	034456	BIT	#BIT13,RMERRS+4	; ADDRESS PLUG CHANGED ?
8938	040212	001011			BNE	8\$	; BR IF YES
8939				:	MOV	#113,-(SP)	; RELEASE COMMAND
8940	040214	012746	000111		MOV	#111,-(SP)	; DRIVE CLEAR
8941	040220	004037	041710		JSR	RD,WAT.RM	; WRITE THE COMMAND INTO RMCS1
8942	040224	000000			RMCS1		; REGISTER INDEX
8943	040226	040562			SC8		; PARITY EXIT ADDRESS
8944	040230	011605		7\$:	MOV	(SP),R5	; PICKUP (RMAS) BEFORE THE ERROR CALL
8945	040232	104002			ERROR	2	; REPORT THE UNEXPECTED ATTENTION
8946	040234	000701			BR	SC4	; GO CHECK FOR MORE ATA'S
8947	040236			8\$:			
8948	040236	104005			ERROR	5	; REPORT THE ADDRESS PLUG CHANGE
8949	040240	000677			BR	SC4	; CHECK FOR MORE DRIVES
8950	040242	006301		SC6:	ASL	R1	; SETUP TO ADDRESS WORDS
8951	040244	012761	177777	034564	MOV	#-1,TIMER(R1)	; STOP THE TIMER
8952	040252	006201			ASR	R1	; RESTORE THE DRIVE ADDRESS
8953	040254	004737	042732		JSR	PC,GETREQ	; GET THE DPB POINTER FROM THE QUEUE
8954	040260	010164	000010		MOV	R1,RMCS2(R4)	; SELECT DRIVE
8955	040264	000137	040612		JMP	SC11	; PROCESS THE SEARCH
8956	040270	004037	041534		JSR	RD,RD.RM	; READ THE RM03'S STATUS REG.
8957	040274	000012			RMD5		
8958	040276	040562			SC8		
8959	040300	011605			MOV	(SP),R5	; AND PUT IT IN R5
8960	040302	006126			ROL	(SP)+	; WAS THERE AN ERROR?
8961	040304	100407			BMI	1\$	; BR IF ERROR
8962	040306	105761	034462		TSTB	DRVACT(R1)	; CHECK DRIVE'S STATE
8963	040312	003137			BGT	SC11	; BR IF DRIVE ACTIVE WITH ORDER
8964	040314	052762	100210	000016	BIS	#BIT15:BIT07:BIT03,16(R2)	; INFORM USER OF ERROR RECOVER COMPLETION
8965	040322	000470			BR	SC7	
8966	040324	004037	041534	1\$:	JSR	RD,RD.RM	; READ ERROR REGISTER #1
8967	040330	000014			RMER1		
8968	040332	040562			SC8		
8969	040334	012605			MOV	(SP)+,R5	; AND SAVE IT IN R5
8970	040336	004737	042076		JSR	PC,SVRH70	; SAVE RH70/RM03 REGISTERS
8971	040342	012746	000111		MOV	#111,-(SP)	; ISSUE A DRIVE CLEAR

8972	040346	004037	041710		JSR	RO,WRT.RM	
8973	040352	000000			RMCS1		
8974	040354	040562			SCB		
8975	040356	006105		SC6A:	ROL	R5	; WAS "UNSAFE" CONDITION =1?
8976	040360	100406			BMI	1\$	; BRANCH IF YES
8977	040362	005702			TST	R2	; ANYTHING IN QUEUE ?
8978	040364	001447			BEQ	SC7	; BR IF NOT
8979	040366	052762	100240	000016	BIS	#BIT15!BIT07!BIT05,16(R2)	; INFORM USER OF ERROR
8980	040374	000443			BR	SC7	
8981	040376	004037	041534	1\$:	JSR	RO,RD.RM	; READ DRIVE STATUS REG. #1
8982	040402	000012			RMDS		
8983	040404	040562			SCB		
8984	040406	011605			MOV	(SP),R5	; SAVE RMDS IN R5
8985	040410	006126			RPL	(SP)+	; "ERR"=1?
8986	040412	100011			BPL	2\$	; BR IF NO--UNSAFE CLEARED
8987	040414	112761	177777	034472	MOVB	#-1,DRVSTA(R1)	; DRIVE IS UNSAFE
8988	040422	004737	042076		JSR	PC,SVRH70	; SAVE RH70/RM03 REGISTERS
8989	040426	052762	110000	000016	BIS	#BIT15!BIT12,16(R2)	; INFORM USER OF UNSAFE ERROR
8990	040434	000423			BR	SC7	
8991	040436	032705	010000	2\$:	BIT	#BIT12,R5	; "MOL" = 1 ?
8992	040442	001015			BNE	3\$	; BR IF YES
8993	040444	112761	177777	034462	MOVB	#-1,DRVACT(R1)	; ACTIVE ERROR RECOVER
8994	040452	112761	000001	034472	MOVB	#1,DRVSTA(R1)	; ONLINE
8995	040460	006301			ASL	R1	
8996	040462	012761	072460	034564	MOV	#30000.,TIMER(R1)	; START 30 SECOND TIMER
8997	040470	006201			ASR	R1	
8998	040472	000137	040040		JMP	SC4	
8999	040476	052762	100220	000016	BIS	#BIT15!BIT07!BIT04,16(R2)	; INFORM USER OF ERROR
9000	040504	105061	034462	SC7:	CLRB	DRVACT(R1)	; DRIVE IS IDLE
9001					JSR	PC,EMPTYQ	; DUMP THE QUEUE
9002	040510	004737	042754		JSR	PC,POPQUE	; REMOVE THE QUEUE
9003	040514	105761	034540		TSTB	ULDFLG(R1)	; UNLOAD IN PROGRESS OR QUEUE?
9004	040520	003002			BGT	1\$	; BR IF NOT
9005	040522	105061	034540		CLRB	ULDFLG(R1)	; CLEAR UNLOAD FLAG
9006	040526	116164	034606	000016	1\$:	MOVB	ATABIT(R1),RMAS(R4)
9007	040534	105761	034472		TSTB	DRVSTA(R1)	; IS THE DRIVE UNSAFE ?
9008	040540	100406			BMI	2\$	; BR IF IT IS
9009					MOV	#113,-(SP)	; RELEASE COMMAND
9010	040542	012746	000111		MOV	#111,-(SP)	; DRIVE CLEAR COMMAND
9011	040546	004037	041710		JSR	RO,WRT.RM	; WRITE THE COMMAND INTO RPCS1
9012	040552	000000			RMCS1		; REGISTER INDEX
9013	040554	040562			SCB		; PARITY EXIT ADDRESS
9014	040556	000137	040040	2\$:	JMP	SC4	; CHECK FOR MORE DRIVES
9015	040562	105761	034462	SCB:	TSTB	DRVACT(R1)	; IS DRIVE IDLE?
9016	040566	001405			BEQ	1\$	; YES--BRANCH
9017	040570	004737	042732		JSR	PC,GETREQ	; GET DPB PCINTER
9018	040574	004737	036736		JSR	PC,C17	; PROCESS THE PARITY ERROR
9019	040600	000402			BR	2\$	; CONTINUE
9020	040602			1\$:			
9021					JSR	PC,C17	; PROCESS THE PARITY ERROR
9022	040602	004737	036760		JSR	PC,C17B	; PROCESS THE UNCORRECTABLE PARITY ERROR
9023	040606	000137	040040	2\$:	JMP	SC4	; CHECK MORE DRIVES
9024	040612	105761	034540	SC11:	TSTB	ULDFLG(R1)	; "UNLOAD IN PROGRESS"?
9025	040616	003402			BLE	1\$	; BRANCH IF NO
9026	040620	105061	034540		CLRB	ULDFLG(R1)	; CLEAR UNLOAD FLAG
9027	040624	105061	034462	1\$:	CLRB	DRVACT(R1)	; SET DRIVE IDLE

```

9028 040630 136137 034606 034534 BITB ATABIT(R1),SRCHWT ;DOING A SEARCH OPERATION FOR
9029 ;AN I/O COMMAND?
9030 040636 001012 BNE 2$ ;BRANCH IF YES
9031 040640 004737 042754 JSR PC,POPQUE ;REMOVE REQUEST FROM QUEUE
9032 040644 052762 000200 000016 BIS #BIT07,16(R2) ;SET "DONE" BIT
9033 040652 005737 034560 TST SAVEFG ;SAVE THE REGISTERS?
9034 040656 100002 BPL 2$ ;BRANCH IF NO
9035 040660 004737 042076 JSR PC,SVRH70 ;YES--SAVE ALL OF THE RH70/RM03 REG'S
9036 040664 116164 034606 000016 2$: MOVB ATABIT(R1),RMAS(R4) ;CLEAR ATTENTION BIT
9037 040672 146137 034606 034534 BICB ATABIT(R1),SRCHWT ;CLEAR IMPLIED SEEK SET
9038 040700 006301 ASL R1 ;WORD INDEX
9039 040702 012761 177777 034564 MOV #-1,TIMER(R1) ;STOP CLOCK
9040 040710 006201 ASR R1 ;RESTORE R1
9041 040712 004737 035656 JSR PC,OPT ;START A REQUEST
9042 040716 000137 040040 JMP SC4 ;CHECK FOR MORE DRIVES
9043 040722 010164 000010 SC12: MOV R1,RMCS2(R4) ;SELECT DRIVE
9044 040726 016437 000012 034452 MOV RMDS(R4),RMERRS ;SAVE THE FOUR REGISTERS THAT
9045 040734 016437 000014 034454 MOV RMER1(R4),RMERRS+2 ;WILL TELL US SOMETHING
9046 040742 016437 000042 034456 MOV RMER2(R4),RMERRS+4
9047 040750 016437 000040 034460 MOV RMMR2(R4),RMERRS+6
9048 ; JSR RO,DRVINT ;INIT. THE STATE OF THE DRIVE
9049 ; BR 1$ ;TAKE ERROR EXIT
9050 040756 000207 RTS PC ;RETURN
9051 040760 005726 1$: TST (SP)+ ;POP PC OFF OF THE STACK
9052 040762 000677 BR SC8 ;PROCESS THE PARITY ERROR
9053 040764 006301 SC13: ASL R1 ;SETUP TO ADDRESS WORDS
9054 040766 012761 177777 034564 MOV #-1,TIMER(R1) ;STOP THE TIMER
9055 040774 006201 ASR R1
9056 040776 010164 000010 MOV R1,RMCS2(R4) ;SELECT THE DRIVE
9057 041002 116164 034606 000016 MOVB ATABIT(R1),RMAS(R4) ;CLEAR THE ATTENTION BIT
9058 041010 105761 034512 1$: TSTB DPINT(R1) ;INITIALIZING THE DRIVE ?
9059 041014 001424 BEQ 2$ ;BR IF NOT
9060 041016 105061 034512 CLRB DPINT(R1) ;CLEAR THE INIT INDICATOR
9061 041022 004037 035050 JSR RO,DRVINT ;GO INIT THE DRIVE
9062 041026 000240 NOP ;DUMMY PARITY ERROR RETURN
9063 041030 105761 034472 TSTB DRVSTA(R1) ;DRIVE ONLINE ?
9064 041034 003014 BGT 2$ ;BR IF YES -- START ORDER
9065 041036 005702 TST R2 ;QUEUE ENTRY FOR THE DRIVE
9066 041040 001426 BEQ 3$ ;BR IF NOT
9067 041042 004737 042732 JSR PC,GETREQ ;GET OPB ADDRESS
9068 041046 052762 140000 000016 BIS #BIT15:BIT14,16(R2) ;INFORM USER THAT DRIVE OFFLINE
9069 041054 004737 042076 JSR PC,SVRH70 ;SAVE THE REGISTERS
9070 ; JSR PC,EMPTYQ ;EMPTY THE REQUEST QUEUE
9071 041060 004737 042754 JSR PC,POPQUE ;REMOVE THE QUEUE
9072 041064 000414 BR 3$
9073 041066 032764 004000 000000 2$: BIT #BIT11,RMCS1(R4) ;DVA SET ?
9074 041074 001006 BNE 4$ ;SET THEN CALL OPT
9075 041076 006301 ASL R1
9076 041100 012761 060000 034564 MOV #60000,TIMER(R1)
9077 041106 006201 ASR R1
9078 041110 000402 BR 3$
9079 041112 004737 035656 4$: JSR PC,OPT ;START THE PENDING REQUEST
9080 041116 000137 040040 3$: JMP SC4 ;PROCESS OTHER DRIVES
9081 ;
9082 ;/RM03 TIMER ROUTINE
9083 ;CALL

```

M13

CZRM880 RM03 2 PERF EXER
CZRM88.P11 21-NOV-77 15:34

MACY11 30(1046) 22-NOV-77 10:06 PAGE 169
SINGLE/DUAL PORT RH70/RM03 DRIVER (REV 5.1)-24-AUG-77

SEG 0168

```

9084      ;      MOV      #TIME-(SP)      ;ELAPSED TIME IN MILLISECONDS ON THE STACK
9085      ;      JSR      PC,RPTMR        ;CALL RM03 TIME ROUTINE
9086
9087      041122 005737 034536      RPTMR: TST      ACTDRV      ;CHECK "ACTDRV & ACTSTR"
9088      041126 001027              BNE      4$          ;IF NON ZERO EXIT
9089      041130 112737 000001 034537      MOV      #1,ACTSTR    ;SET "ACTSTR"
9090      041136 104412              SAVREG      ;SAVE R0 - R5
9091      041140 005001              CLR      R1          ;START WITH DRIVE 0
9092      041142 005003              CLR      R3
9093      041144 005763 034564      1$:  TST      TIMER(R3)      ;IS THE TIMER RUNNING?
9094      041150 002406              BLT      2$          ;BRANCH IF NO
9095      041152 166663 000002 034564      SUB      2(SP),TIMER(R3) ;COUNT THE INTERVAL
9096      041160 003002              BGT      2$          ;BR IF NO SOFTWARE TIMEOUT
9097      041162 004737 041212      JSR      PC,ST0        ;CALL SOFTWARE TIMEOUT ROUTINE
9098      041166 005201      2$:  INC      R1          ;MOVE TO NEXT DRIVE
9099      041170 005723              TST      (R3)+
9100      041172 022701 000010      CMP      #8,R1          ;OUT OF DRIVES?
9101      041176 003362              BGT      1$          ;BRANCH IF NO
9102      041200 104413      3$:  RESREG      ;RESTORE R0 - R5
9103      041202 105037 034537      CLRB      ACTSTR        ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
9104      041206 012616      4$:  MOV      (SP)+,(SP)      ;ADJUST THE STACK
9105      041210 000207      RTS      PC          ;RETURN
9106
9107      ;SOFTWARE TIMEOUT ROUTINE
9108
9109      ;NOTE: THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
9110      ;OR GREATER
9111
9112      ;CALL:
9113      ;      ST0      MOV      #DRVNUM,R1      ;DRIVE NUMBER
9114      ;      JSR      PC,ST0        ;CALL
9115      ;
9116
9117      041212 010146      ST0:  MOV      R1,-(SP)      ;SAVE R1
9118      041214 010246      MOV      R2,-(SP)      ;SAVE R2
9119      041216 010346      MOV      R3,-(SP)      ;SAVE R3
9120      041220 010446      MOV      R4,-(SP)      ;SAVE R4
9121      041222 013704 034620      MOV      RMADR,R4      ;GET ADDRESS OF "RMCS1"
9122      041226 010164 000010      MOV      R1,RMCS2(R4)    ;SELECT THE DRIVE
9123      041232 004037 041534      JSR      R0,RD.RM      ;READ "DRIVE STATUS REG"
9124      041236 000012      RMD5
9125      ;      ST05
9126      041240 041522      ;      ST09
9127      041242 105726      TST      (SP)+          ;IS "DRY"=1?
9128      041244 100436      BMI      ST02          ;BR IF YES
9129      041246 105761 034512      ST01: TSTB      DPINT(R1)      ;TRYING TO INITIALIZE THE DRIVE ^
9130      041252 001033      BNE      ST02          ;BR IF YES
9131      041254 105761 034522      TSTB      DPRQS(R1)      ;OUTSTANDING PORT REQUEST FOR THE DRIVE ^
9132      041260 001030      BNE      ST02          ;BR IF YES
9133      041262 013702 034532      MOV      TRANSW,R2      ;PICKUP TRANSFER WAIT QUEUE
9134      041266 020137 034604      CMP      R1,DTUW      ;TRANSFER UNDERWAY ON THIS DRIVE?
9135      041272 001404      BEQ      1$          ;BRANCH IF YES
9136      041274 000137 041522      JMP      ST09          ;IF NOT DON'T BOTHER DRIVES
9137      041300 004737 042732      JSR      PC,GETREQ      ;GET DPB ADDRESS
9138      041304 052762 101000      1$:  BIS      #BIT15:BIT09,16(R2) ;SET THE ERROR FLAGS
9139      041312 004737 042076      JSR      PC,SVRH70      ;SAVE RH70/RM03 REGISTERS

```

```

9140      ;      MOV      #BIT05,RMCS2(R4) ; "INIT" THE MASS BUS
9141      041316 105061 034462      ;      CLRB      DRVACT(R1) ; DRIVE IS IDLE
9142      041322 105061 034540      ;      CLRB      ULDFLG(R1) ; CLEAR THE UNLOAD FLAG
9143      041326 005037 034532      ;      CLR       TRANSW  ; CLEAR DPB ADDRESS
9144      041332 012737 177777 034604 ;      MOV      #-1,DTUM  ; CLEAR THE TRANSFER DRIVE #
9145      041340 000470      ;      BR       ST09      ; DON'T BOTHER OTHER DRIVES
9146      041342 116405 000016      ;      MOVB     RMAS(R4),R5 ; READ ATTENTION REG
9147      041346 136105 034606      ;      BITB     ATABIT(R1),R5 ; IS ATTENTION FOR THIS DRIVE UP ?
9148      041352 001007      ;      BNE      ST03      ; YES--BRANCH
9149      041354 105761 034512      ;      TSTB     DPINT(R1) ; TRYING TO INITIALIZE THE DRIVE ?
9150      041360 001021      ;      BNE      ST06      ; BR IF YES - DRIVE NOT ONLINE
9151      041362 105761 034522      ;      TSTB     DPRQS(R1) ; OUTSTANDING PORT REQUEST FOR THE DRIVE ?
9152      041366 001035      ;      BNE      ST07      ; BR IF YES - NO RESPONSE TO REQUEST
9153      041370 000454      ;      BR       ST09      ; OTHER WISE EXIT
9154      041372 105761 034512      ;      TSTB     DPINT(R1) ; INITIALIZING THE DRIVE ?
9155      041376 001003      ;      BNE      IS      ; BR IF INIT PENDING
9156      041400 105761 034522      ;      TSTB     DPRQS(R1) ; PORT REQUEST PENDING ?
9157      041404 001446      ;      BEQ      ST09      ; BR IF NOT
9158      041406 012763 177777 034564 ;      MOV      #-1,TIMER(R3) ; STOP THE TIMER
9159      041414 000442      ;      BR       ST09      ; EXIT
9160      041416 004737 037036      ;      JSR      PC,CIB    ; GO HANDLE THE PARITY ERROR
9161      041422 000437      ;      BR       ST09      ;
9162      041424 105061 034512      ;      CLRB     DPINT(R1) ; CLEAR THE INITIALIZE INDICATOR
9163      041430 105061 034472      ;      CLRB     DRVSTA(R1) ; SET UNIT OFFLINE
9164      041434 012763 177777 034564 ;      MOV      #-1,TIMER(R3) ; STOP THE TIMER
9165      041442 004737 042732      ;      JSR      PC,GETREQ ; GET THE DPB ADDRESS
9166      041446 005702      ;      TST      R2        ; REQUEST IN QUEUE ?
9167      041450 001424      ;      BEQ      ST09      ; BR IF NOT
9168      041452 052762 140000 000016 ;      BIS      #BIT15!BIT14,16(R2) ; INFORM THE USER DRIVE NOT AVAILABLE
9169      041460 000414      ;      BR       ST08      ; FINISH
9170      041462 012763 177777 034564 ;      MOV      #-1,TIMER(R3) ; STOP THE TIMER
9171      041470 105061 034522      ;      CLRB     DPRQS(R1) ; CLEAR PORT REQUEST INDICATOR
9172      041474 004737 042732      ;      JSR      PC,GETREQ ; GET DPB ADDRESS
9173      041500 005702      ;      TST      R2        ; QUEUE ENTRY FOR DRIVE ?
9174      041502 001407      ;      BEQ      ST09      ; BR IF NONE
9175      041504 012762 100004 000016 ;      MOV      #BIT15!BIT2,16(R2) ; INFORM USER OF PORT REQUEST ERROR
9176      041512 004737 042636      ;      JSR      PC,EMPTYQ ; CLEAR THE QUEUE FOR THE DRIVE
9177      041516 004737 042076      ;      JSR      PC,SVRH70 ; SAVE THE REGISTERS
9178      041522 012604      ;      MOV      (SP)+,R4 ; RESTORE R4
9179      041524 012603      ;      MOV      (SP)+,R3 ; RESTORE R3
9180      041526 012602      ;      MOV      (SP)+,R2 ; RESTORE R2
9181      041530 012601      ;      MOV      (SP)+,R1 ; RESTORE R1
9182      041532 000207      ;      RTS      PC      ; RETURN
9183
9184      ; ROUTINE TO READ A RM70/RM03 REGISTER
9185
9186      ; CALL
9187      ;      JSR      RD,RD.RM ; GO READ A REGISTER
9188      ;      INDEX   ERRADR ; REG. INDEX FROM BASE
9189      ;      ;      ; ERROR ADDRESS--PROCESS ERROR STARTING
9190      ;      ;      ; AT THIS ADDRESS
9191      ;      ;      ; CONTENTS OF REG. IS ON THE STACK
9192
9193      041534 013737 034616 041676 ; RD.RM: MOV      MCPMX,RD.RM2 ; MAX. RETRYS ALLOWED
9194      041542 011646      ;      MOV      (SP)-,(SP) ; SAVE RD FOR RETURN
9195      041544 013737 034620 041560 ;      MOV      RMADR,RD.ADR ; FORM THE DESIRED ADDRESS

```

Address	Hex	Dec	Label	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op41
---------	-----	-----	-------	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	------

Address	Offset	Hex	Label	Instruction	Comment
9252	041762	112637	041777	MOV B	(SP)+, WRT.WD+1
9253	041766	063737	034620	ADD	RMADR, WRT.RD
9254	041774	012737		WRT.R1: MOV	(PC)+, 2(PC)+
9255	041776	000000		WRT.WD: .WORD	0
9256	042000	000000		WRT.RD: .WORD	0
9257	042002	013746	034620	MOV	RMADR, -(SP)
9258	042006	062716	000010	ADD	#RMC52, (SP)
9259	042012	032736	010000	BIT	#BIT12, 2(SP)+
9260	042016	001023		BNE	WRT.R3
9261	042020	004037	041534	JSR	RD, RD.RM
9262	042024	000014		RMR1	
9263	042026	042066		WRT.R3	
9264	042030	032726	000010	BIT	#BIT03, (SP)+
9265	042034	001416		BEQ	WRT.R4
9266	042036	016037	177776	MOV	-2(RD), 1\$
9267	042044	004037	041534	JSR	RD, RD.RM
9268	042050	000000		1\$: .WORD	0
9269	042052	042066		WRT.R3	
9270	042054	104004		ERROR	4
9271	042056	005726		TST	(SP)+
9272	042060	005327		DEC	(PC)+
9273	042062	000003		WRT.R2: .WORD	3
9274	042064	002343		BGE	WRT.R1
9275	042066	011000		WRT.R3: MOV	(RD), RD
9276	042070	000401		BR	WRT.R5
9277	042072	005720		WRT.R4: TST	(RD)+
9278	042074	000200		WRT.R5: RTS	RD
9279					
9280					: ROUTINE TO SAVE THE RH70/RP04/5/RM03 REGISTERS AS PER DPB+14
9281					
9282					: CALL
9283					
9284				MOV	#DPBNUM, R2
9285				JSR	PC, SVRH70
9286	042076	104412		SVRH70: SAVREG	
9287	042100	005702		TST	R2
9288	042102	001442		BEQ	6\$
9289	042104	013704	034620	MOV	RMADR, R4
9290	042110	111264	000010	MOVB	(R2), RMCS2(R4)
9291	042114	016203	000014	MOV	14(R2), R3
9292	042120	001433		BEQ	6\$
9293	042122	005037	042156	CLR	3\$
9294	042126	023727	042156	CMP	3\$, #RMOB
9295	042134	001006		BNE	2\$
9296	042136	032764	000200	BIT	#BIT07, RMCS2(R4)
9297	042144	001002		BNE	2\$
9298	042146	005023		CLR	(R3)+
9299	042150	000405		BR	4\$
9300	042152	004037	041534	JSR	RD, RD.RM
9301	042156	000000		3\$: .WORD	0
9302	042160	042204		5\$	
9303	042162	012623		MOV	(SP)+, (R3)+
9304	042164	023727	042156	CMP	3\$, #RMEC2
9305	042172	001406		BEQ	6\$
9306	042174	062737	000002	ADD	#2, 3\$
9307	042202	000751		BR	1\$

CZRM880 RM03/2 PERF EXER
CZRM88.P11 21-NOV-77 15:34

MACY11 30(1046) 22-NOV-77 10:06 PAGE 173
SINGLE/DUAL PORT RH70/RM03 DRIVER (REV 5.1)-24-AUG-77

SEQ 0172

9308 042204 004737 036736
9309 042210 104413
9310 042212 000207
9311
9312
9313
9314
9315
9316
9317
9318 042214 010446
9319 042216 013704 034620
9320 042222 010164 000010
9321 042226 011446
9322 042230 052716 040000
9323 042234 000316
9324 042236 112714 000100
9325 042242 032764 010000 000010
9326 042250 001002
9327 042252 005726
9328 042254 000402
9329 042256 112664 000001
9330 042262 012604
9331 042264 000207
9332
9333
9334 042266 000
9335 042267 000
9336 042270 000
9337 042271 000
9338 042272 000
9339 042273 000
9340 042274 000
9341 042275 000
9342
9343
9344
9345 042276 042360
9346 042300 042400
9347 042302 042420
9348 042304 042440
9349 042306 042460
9350 042310 042500
9351 042312 042520
9352 042314 042540
9353
9354
9355
9356 042316 042360
9357 042320 042400
9358 042322 042420
9359 042324 042440
9360 042326 042460
9361 042330 042500
9362 042332 042520
9363 042334 042540

```

5$: JSR PC,C17 ;PROCESS THE UNCORRECTABLE PARITY ERROR
6$: RESREG ;RESTORE R0 - R5
RTS PC ;RETURN

;ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A "TRE"
;CALL
;   MOV #DRVNUM,R1 ;DRIVE NUMBER TO R1
;   JSR PC,SET.IE ;SET "IE"
;   RETURN

SET.IE: MOV R4, -(SP) ;SAVE R4
        MOV RMADR,R4 ;PICKUP ADDRESS OF RMCS1
        MOV R1,RMCS2(R4) ;SELECT DRIVE
        MOV (R4),-(SP) ;READ RMCS1
        BIS #BIT14,(SP) ;SET THE "TRE" BIT OF THE WORD READ
        SWAB (SP) ;ADJUST FOR DAT0
        MOVB #BIT06,(R4) ;SET "IE"
        BIT #BIT12,RMCS2(R4) ;IS "NED"=1?
        BNE 1$ ;YES--CLEAR "TRE"
        TST (SP)+ ;CLEAN OFF THE STACK
        BR 2$

1$: MOVB (SP)+,1(R4) ;CLEAR "TRE"
2$: MOV (SP)+,R4 ;RESTORE R4
RTS PC ;RETURN TO CALLER

;QUEUE COUNT
QCNT: .BYTE 0 ;DRIVE 0
      .BYTE 0 ;DRIVE 1
      .BYTE 0 ;DRIVE 2
      .BYTE 0 ;DRIVE 3
      .BYTE 0 ;DRIVE 4
      .BYTE 0 ;DRIVE 5
      .BYTE 0 ;DRIVE 6
      .BYTE 0 ;DRIVE 7

;QUEUE INPUT POINTERS
QINPT: .WORD QDRV0 ;DRIVE 0
       .WORD QDRV1 ;DRIVE 1
       .WORD QDRV2 ;DRIVE 2
       .WORD QDRV3 ;DRIVE 3
       .WORD QDRV4 ;DRIVE 4
       .WORD QDRV5 ;DRIVE 5
       .WORD QDRV6 ;DRIVE 6
       .WORD QDRV7 ;DRIVE 7

;QUEUE OUTPUT POINTERS
QOUTPT: .WORD QDRV0 ;DRIVE 0
        .WORD QDRV1 ;DRIVE 1
        .WORD QDRV2 ;DRIVE 2
        .WORD QDRV3 ;DRIVE 3
        .WORD QDRV4 ;DRIVE 4
        .WORD QDRV5 ;DRIVE 5
        .WORD QDRV6 ;DRIVE 6
        .WORD QDRV7 ;DRIVE 7

```

```

9364
9365 042336 042360 QSTART: .WORD QDRV0 ;DRIVE 0 START ADDRESS
9366 042340 042400 QSTOP: .WORD QDRV1 ;DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
9367 042342 042420 .WORD QDRV2 ;STOP DRIVE 1--START DRIVE 2
9368 042344 042440 .WORD QDRV3 ;STOP DRIVE 2--START DRIVE 3
9369 042346 042460 .WORD QDRV4 ;STOP DRIVE 3--START DRIVE 4
9370 042350 042500 .WORD QDRV5 ;STOP DRIVE 4--START DRIVE 5
9371 042352 042520 .WORD QDRV6 ;STOP DRIVE 5--START DRIVE 6
9372 042354 042540 .WORD QDRV7 ;STOP DRIVE 6--START DRIVE 7
9373 042356 042560 .WORD QTERM ;STOP DRIVE 7
9374
9375 ;DRIVE REQUEST QUEUES
9376
9377 042360 000010 QDRV0: .BLKW 10
9378 042400 000010 QDRV1: .BLKW 10
9379 042420 000010 QDRV2: .BLKW 10
9380 042440 000010 QDRV3: .BLKW 10
9381 042460 000010 QDRV4: .BLKW 10
9382 042500 000010 QDRV5: .BLKW 10
9383 042520 000010 QDRV6: .BLKW 10
9384 042540 000010 QDRV7: .BLKW 10
9385 042560 QTERM=.
9386
9387 ;ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
9388
9389 ;CALL
9390 JSR PC,CLRQUE
9391
9392 042560 104412 CLRQUE: SAVREG ;SAVE R0 - R5
9393 042562 012702 042266 MOV #QCNT,R2 ;ZERO THE QUEUE COUNTS
9394 042566 005022 CLR (R2)+ ;DRIVES 0 & 1
9395 042570 005022 CLR (R2)+ ;DRIVES 2 & 3
9396 042572 005022 CLR (R2)+ ;DRIVES 4 & 5
9397 042574 005022 CLR (R2)+ ;DRIVES 6 & 7
9398 042576 012703 000010 MOV #8,R3 ;MOVE THE STARTING
9399 042602 012701 042336 MOV #QSTART,R1 ;ADDRESS OF THE QUEUE INTO
9400 042606 012122 1$: MOV (R1)+,(R2)+ ;THE QUEUE INPUT POINTER
9401 042610 005303 DEC R3
9402 042612 001375 BNE 1$
9403 042614 012703 000010 MOV #8,R3 ;MOVE THE STARTING ADDRESS
9404 042620 012701 042336 MOV #QSTART,R1 ;OF THE QUEUE INTO THE
9405 042624 012122 2$: MOV (R1)+,(R2)+ ;QUEUE OUTPUT POINTER
9406 042626 005303 DEC R3
9407 042630 001375 BNE 2$
9408 042632 104413 RESREG ;RESTORE R0 - R5
9409 042634 000207 RTS PC
9410
9411 ;EMPTY THE QUEUE SPECIFIED BY R1
9412
9413 ;CALL
9414 MOV DRVNUM,R1 ;DRIVE NUMBER TO R1
9415 JSR PC,EMPTYQ
9416
9417 042636 105061 042266 EMPTYQ: CLRB QCNT(R1) ;CLEAR NUMBER OF ITEMS IN QUEUE
9418 042642 006301 R1
9419 042644 016161 042276 042316 MOV QINPT(R1),QOUTPT(R1) ;SET OUTPUT QUEUE POINTER=INPUT POINTER

```



```

9420 042652 006201      ASR      R1
9421 042654 000207      RTS      PC
9422
9423      ;ROUTINE TO PUT A REQUEST IN QUEUE
9424      :CALL
9425      :      MOV      #DRVNUM,R1      ;DRIVE NUMBER
9426      :      MOV      #DPB,R2      ;ADDRESS OF PARAMETER BLOCK
9427      :      JSR      RO,DRVQUE      ;GO PUT REQUEST IN QUEUE
9428      :      RETURN1      ;RETURN HERE IF QUEUE IS FULL
9429      :      RETURN2      ;RETURN HERE IF REQUEST IS IN QUEUE
9430
9431
9432 042656 122761 000010 042266 DRVQUE: CMPB      #10,QCNT(R1)      ;IS QUEUE FULL?
9433 042664 001421      BEQ      2$      ;BR IF YES-TAKE RETURN1
9434 042666 105261 042266      INCB      QCNT(R1)      ;INCREMENT QUEUE COUNT
9435 042672 006301      ASL      R1
9436 042674 010271 042276      MOV      R2,QOINPT(R1)      ;PUT THIS REQUEST IN QUEUE
9437 042700 062761 000002 042276      ADD      #2,QINPT(R1)      ;UPDATE THE QUEUE POINTER
9438 042706 026161 042276 042340      CMP      QINPT(R1),QSTOP(R1)      ;TIME TO RESET THE POINTER
9439 042714 001003      BNE      1$      ;BRANCH IF NO
9440 042716 016161 042336 042276      MOV      QSTART(R1),QINPT(R1)      ;YES--RESET POINTER
9441 042724 006201      1$:      ASR      R1
9442 042726 005720      TST      (R0)+      ;TAKE RETURN 2
9443 042730 000200      2$:      RTS      RO      ;RETURN TO USER
9444
9445      ;ROUTINE TO GET THE "DPB" ADDRESS OF NEXT REQUEST IN QUEUE
9446      :CALL
9447      :      MOV      #DRVNUM,R1      ;DRIVE NUMBER TO R1
9448      :      JSR      PC,GETREQ      ;GO GET THE REQUEST
9449      :      RETURN      ;R2="DPB" ADDRESS OF THE REQUEST
9450      :      ;R2=0 IF NO REQUEST IN QUEUE
9451
9452
9453 042732 005002      GETREQ: CLR      R2
9454 042734 105761 042266      TSTB      QCNT(R1)      ;IS THERE ANY REQUEST IN QUEUE?
9455 042740 001404      BEQ      2$      ;NO---BRANCH
9456 042742 006301      1$:      ASL      R1
9457 042744 017102 042316      MOV      QOOUTPT(R1),R2      ;PICKUP "DPB" POINTER FOR THIS DRIVE
9458 042750 006201      ASR      R1
9459 042752 000207      RTS      PC      ;RETURN TO JSR
9460
9461      ;ROUTINE TO "POP" THE REQUEST FROM QUEUE
9462      :CALL
9463      :      MOV      #DRVNUM,R1      ;DRIVE NUMBER TO R1
9464      :      JSR      PC,POPQUE      ;CALL TO REMOVE REQUEST
9465      :      RETURN      ;R2=ADDRESS OF DPB REMOVED
9466
9467
9468 042754 105361 042266      POPQUE: DECB      QCNT(R1)      ;DECREMENT QUEUE COUNT
9469 042760 006301      ASL      R1
9470 042762 017102 042316      MOV      QOOUTPT(R1),R2      ;GET THE "DPB" POINTER
9471 042766 005071 042316      CLR      QOOUTPT(R1)      ;REMOVE DPB ADDRESS FROM THE QUEUE
9472 042772 062761 000002 042316      ADD      #2,QOOUTPT(R1)      ;UPDATE THE QUEUE POINTER
9473 043000 026161 042316 042340      CMP      QOOUTPT(R1),QSTOP(R1)      ;TIME TO RESET THE POINTER?
9474 043006 001003      BNE      1$      ;NO--BRANCH TO EXIT
9475 043010 016161 042336 042316      MOV      QSTART(R1),QOOUTPT(R1)      ;YES--RESET THE POINTER

```

```

9476 043016 006201      1S:   ASR   R1
9477 043020 000207      RTS   PC                ;RETURN TO USER
9478
9479
9480
9481
9482
9483
9484
9485
9486
9487
9488
9489
9490
9491
9492
9493
9494
9495
9496
9497
9498
9499
9500
9501
9502
9503
9504
9505
9506
9507
9508
9509
9510
9511
9512
9513
9514
9515
9516
9517
9518
9519
9520
9521
9522
9523
9524
9525
9526
9527
9528
9529
9530
9531

```

```

;*****
.SBTL  DATA, CONTROL, & STATUS BLOCKS
;*****

;BLOCK LOCATION EQUATE STATEMENTS
SFMT   =      1      ;FMT HCI ECI OR OFFSET CODE
$COMND =      $FMT+1 ;OPERATION CODE
$PSEL  =      $FMT+2 ;PORT SELECT & BITS A16, A17
$WRDM  =      $FMT+3 ;WORD COUNT (2'S COMP)
$BUF   =      $FMT+5 ;BUFFER ADDR OR REGISTER TABLE POINTER
$SEC   =      $FMT+7 ;SECTOR ADDRESS OR 1ST REG ADDR
$TRK   =      $FMT+10;TRACK ADDRESS OF LAST REG ADDR
$CYL   =      $FMT+11;CYLINDER ADDR
$REG   =      $FMT+13;REGISTER STORAGE (IF ERROR)
$STATUS=      $FMT+15;STATUS WORD (SET BY DRIVER)

;DRIVE'S HISTORY AND CURRENT INDICATOR STORAGE EQUATES
$WRDL  =      $FMT+17;WORD COUNT (NOT 2'S COMP)
$SSEC  =      $WRDL+2;SECTOR SIZE FOR CURRENT OPERATION
$CODE  =      $WRDL+4;PRESENT COMMAND SELECTION CODE
$PACK  =      $WRDL+6;WRITE DATA PACK INDICATOR
$PREVO =      $WRDL+7;PREVIOUS COMMAND SELECTION CODE
$PATT  =      $WRDL+10;PATTERN CODE
$PREVA =      $WRDL+12;PREVIOUS ADDRESS - TRACK, SECTOR, CYLINDER
$OPERC =      $WRDL+16;OPERATION COUNT
$POSIT =      $WRDL+22;SEEK COUNT
$TRANS =      $WRDL+26;TOTAL BITS XFERED COUNT (R & W)
$READ  =      $WRDL+32;TOTAL BITS READ COUNT
$TOTAL =      $WRDL+36;TOTAL ERRORS (ALL TYPES) COUNT
$SOFT  =      $WRDL+40;'SOFT' ERROR COUNT
$HARD  =      $WRDL+42;'HARD' ERROR COUNT
$SKI   =      $WRDL+44;'SKI' OR 'OCYL' ERROR COUNT
$MISPO =      $WRDL+46;PROG DETECTED MISPOSITIONING ERROR S COUNT
$PASSC =      $WRDL+50;PASS COUNTER
$FAIR  =      $WRDL+52;OPERATION QUEUE 'FAIRNESS' COUNT

;INDEX EQUATES TO THE NEXT OPERATION PARAMETERS
$NCODE =      $WRDL+54;NEXT OPERATION CODE
$NPATC =      $NCODE+1;NEXT PATTERN
$NSEC  =      $NCODE+2;NEXT SECTOR
$NTRK  =      $NCODE+3;NEXT TRACK
$NCYL  =      $NCODE+4;NEXT CYLINDER
$NWRDL =      $NCODE+6;NEXT BUFFER SIZE
$NEXT  =      $NCODE+10;PARAMETER SELECTION INDICATOR

;INDEX EQUATES FOR MAXIMUM MINIMUM ADDRESSES

```

```

9532      000106      MAXCYL =      $NCODE+12      ;MAXIMUM CYLINDER ADDRESS
9533      000110      MINCYL =      MAXCYL+2      ;MINIMUM CYLINDER ADDRESS
9534      000112      MAXTRK =      MAXCYL+4      ;MAXIMUM TRACK ADDRESS
9535      000114      MINTRK =      MAXCYL+6      ;MINIMUM TRACK ADDRESS
9536      000116      MAXSEC =      MAXCYL+10     ;MAXIMUM SECTOR ADDRESS
9537      000120      MINSEC =      MAXCYL+12     ;MINIMUM SECTOR ADDRESS
9538      000122      $FIRST =      MAXCYL+14     ;FIRST OPERATION INDICATOR
9539
9540      ;BAD SECTOR/TRACK ADDRESS STORAGE AREA INDEX EQUATE
9541
9542      000124      $BDSEC =      MAXCYL+16      ;BAD SECTOR STORAGE TABLE
9543
9544      ;DRIVE ID AREA INDEX EQUATE
9545
9546      000224      $DRVID =      $BDSEC+100      ;DRIVE ID
9547
9548      ;RH11/RM03 REGISTER EQUATES
9549
9550      000234      $RMCS1 =      $DRVID+10      ;RM03 REGISTER STORAGE
9551      000236      $RMWC =      $RMCS1+2
9552      000240      $RMBA =      $RMCS1+4
9553      000242      $RMDA =      $RMCS1+6
9554      000244      $RMCS2 =      $RMCS1+10
9555      000246      $RMDS =      $RMCS1+12
9556      000250      $RMER1 =      $RMCS1+14
9557      000252      $RMAS =      $RMCS1+16
9558      000254      $RMLA =      $RMCS1+20
9559      000256      $RMD8 =      $RMCS1+22
9560      000260      $RMMR1 =      $RMCS1+24
9561      000262      $RMDT =      $RMCS1+26
9562      000264      $RMSN =      $RMCS1+30
9563      000266      $RMOF =      $RMCS1+32
9564      000270      $RMDC =      $RMCS1+34
9565      000272      $RMHR =      $RMCS1+36
9566      000274      $RMMR2 =      $RMCS1+40
9567      000276      $RMER2 =      $RMCS1+42
9568      000300      $RMEC1 =      $RMCS1+44
9569      000302      $RMEC2 =      $RMCS1+46
9570
9571
9572      ;BLOCK FOR DRIVE 0
9573
9574      043022      000      000      DRIVED: .BYTE 0,0      ;DRIVE NUMBER
9575      043024      000005      .BLKW 5
9576      043036      043256      .WORD .+$RMCS1-$REG
9577      043040      000266      .BLKB $RMEC2-$REG
9578
9579
9580      ;BLOCK FOR DRIVE 1
9581
9582      043326      001      000      DRIVE1: .BYTE 1,0      ;DRIVE NUMBER
9583      043330      000005      .BLKW 5
9584      043342      043562      .WORD .+$RMCS1-$REG
9585      043344      000266      .BLKB $RMEC2-$REG
9586
9587

```

```

9588      ;BLOCK FOR DRIVE 2
9589
9590      043632      002      000      DRIVE2: .BYTE      2,0      ;DRIVE NUMBER
9591      043634      000005      .BLKW      5
9592      043646      044066      .WORD      .+SRMCS1-$REG
9593      043650      000266      .BLKB      $RMEC2-$REG
9594
9595
9596      ;BLOCK FOR DRIVE 3
9597
9598      044136      003      000      DRIVE3: .BYTE      3,0      ;DRIVE NUMBER
9599      044140      000005      .BLKW      5
9600      044152      044372      .WORD      .+SRMCS1-$REG
9601      044154      000266      .BLKB      $RMEC2-$REG
9602
9603
9604      ;BLOCK FOR DRIVE 4
9605
9606      044442      004      000      DRIVE4: .BYTE      4,0      ;DRIVE NUMBER
9607      044444      000005      .BLKW      5
9608      044456      044676      .WORD      .+SRMCS1-$REG
9609      044460      000266      .BLKB      $RMEC2-$REG
9610
9611
9612      ;BLOCK FOR DRIVE 5
9613
9614      044746      005      000      DRIVE5: .BYTE      5,0      ;DRIVE NUMBER
9615      044750      000005      .BLKW      5
9616      044762      045202      .WORD      .+SRMCS1-$REG
9617      044764      000266      .BLKB      $RMEC2-$REG
9618
9619
9620      ;BLOCK FOR DRIVE 6
9621
9622      045252      006      000      DRIVE6: .BYTE      6,0      ;DRIVE NUMBER
9623      045254      000005      .BLKW      5
9624      045266      045506      .WORD      .+SRMCS1-$REG
9625      045270      000266      .BLKB      $RMEC2-$REG
9626
9627
9628      ;BLOCK FOR DRIVE 7
9629
9630      045556      007      000      DRIVE7: .BYTE      7,0      ;DRIVE NUMBER
9631      045560      000005      .BLKW      5
9632      045572      046012      .WORD      .+SRMCS1-$REG
9633      045574      000266      .BLKB      $RMEC2-$REG
9634
9635
9636      ;GENERAL PURPOSE DPB - USED BY 'READHD', 'RECALT', 'OFFSET', & 'RTNCTR'
9637
9638      046062      000000      000000      177776      GENDPB: .WORD      0,0,-2,CYLDER
9639      046070      056512
9640      046072      000000      000000      046102      .WORD      0,0,GENREG,0
9641      046100      000000
9642      046102      000024      GENREG: .BLKW      24      ;REGISTER STORAGE IF ERROR
9643

```

9644
9645
9646
9647
9648
9649
9650

;*****

.SBTTL ERROR MESSAGES

;*****

.NLIST BEX

046152	044122	030067	030450	EM1:	.ASCIZ	/RH70(11) INTERRUPT OCCURRED (RMAS = 0)/
046221	125	042516	050130	EM2:	.ASCIZ	/UNEXPECTED ATTENTION OCCURRED/
046257	115	051501	041123	EM3:	.ASCIZ	/MASSBUS PARITY ERROR (MCPE=1)/
046315	115	051501	041123	EM4:	.ASCIZ	/MASSBUS PARITY ERROR (PAR=1)/
046352	042101	051104	051505	EM5:	.ASCIZ	/ADDRESS PLUG CHANGE BIT SET/
046406	044122	030067	030450	EM6:	.ASCIZ	/RH70(11) DIDN'T RESPOND TO ADDRESSING/
046454	047125	047503	051122	EM10:	.ASCIZ	/UNCORRECTABLE MASSBUS PARITY ERROR/
046517	106	052101	046101	EM11:	.ASCIZ	/FATAL MASSBUS PARITY ERROR/
046552	042520	051522	051511	EM12:	.ASCIZ	/PERSISTENT DEVICE UNSAFE/
046603	117	042520	040522	EM13:	.ASCIZ	/OPERATION NOT COMPLETED WITHIN TIME LIMIT/
046655	104	044522	042526	EM14:	.ASCIZ	/DRIVE WENT OFFLINE/
046700	047516	051040	051505	EM15:	.ASCIZ	/NO RESPONSE TO PORT REQUEST/
046734	042510	042101	051105	EM20:	.ASCIZ	/HEADER CRC ERROR/
046755	104	052101	020101	EM21:	.ASCIZ	/DATA CHECK ('DCK') ERROR/
047006	051127	052111	020105	EM22:	.ASCIZ	/WRITE CHECK ERROR - DATA CHECK ('DCK') SET/
047061	127	044522	042524	EM23:	.ASCIZ	/WRITE CHECK ERROR - DATA CHECK ('DCK') NOT SET/
047140	042510	042101	051105	EM24:	.ASCIZ	/HEADER READ ERROR - 'FMT' BIT DROPPED/
047206	042510	042101	051105	EM25:	.ASCIZ	/HEADER READ ERROR - HEADER COMPARE ('HCE') ERROR/
047267	106	051117	040515	EM26:	.ASCIZ	/FORMAT ERROR ('FER')/
047314	042510	042101	051105	EM27:	.ASCIZ	/HEADER COMPARE ('HCE') ERROR/
047351	115	051511	042503	EM30:	.ASCIZ	/MISCELLANEOUS DRIVE ERROR/
047403	117	042520	040522	EM31:	.ASCIZ	/OPERATION INCOMPLETE ('OPI') ERROR/
047446	051104	053111	020105	EM32:	.ASCIZ	/DRIVE TIMING ('DTE') ERROR/
047501	120	051101	052111	EM33:	.ASCIZ	/PARITY ('PAR') ERROR AFTER OPERATION STARTED.

K14

CZRM880 RM03/2 PERF EXER
CZRM88.P11 21-NOV-77 15:34MACY11 30(1046) 22-NOV-77 10:06 PAGE 180
ERROR MESSAGES

SEQ 0179

```

047556 051127 052111 020105 EM34: .ASCIZ /WRITE CLOCK FAILURE ('WCF') ERROR/
047620 047111 040526 044514 EM35: .ASCIZ /INVALID ADDRESS ('IAE') ERROR/
047656 051127 052111 020105 EM36: .ASCIZ /WRITE LOCK ('WLE') ERROR/
047707 104 052101 020101 EM37: .ASCIZ /DATA CHECK ('DCK') SET DURING WRITE CHECK COMMAND/
047771 122 030510 020061 EM40: .ASCIZ /RH11 OR UNIBUS TRANSFER ERROR/
050027 102 051525 040440 EM41: .ASCIZ /BUS ADDRESS OR WORD COUNT INCORRECT/
050073 104 052101 020101 EM42: .ASCIZ /DATA COMPARE ERRORS - NO OTHER ERROR(S) DETECTED/
050154 040503 023516 020124 EM43: .ASCIZ /CAN'T MATCH DATA READ WITH A PATTERN/
050221 105 051122 051117 EM44: .ASCIZ /ERROR BIT(S) SET, BUT NO ERROR SIGNALLED BY THE RH11/
050305 105 041503 046040 EM45: .ASCIZ /ECC LOGIC FAILURE - POSITION REGISTER VALUE NOT VALID/
050373 102 051525 040440 EM46: .ASCIZ /BUS ADDRESS AND WORD COUNT NOT CONSISTENT/
050445 123 042505 020113 EM50: .ASCIZ /SEEK INCOMPLETE ('SKI') ERROR/
050503 120 047522 051107 EM51: .ASCIZ /PROGRAM DETECTED POSITIONING ERROR/
050546 051104 053111 020105 EM60: .ASCIZ /DRIVE UNSAFE ERROR/
050571 122 040515 000123 DH1: .ASCIZ /RMAS/
050576 051104 053111 020105 DH2: .ASCIZ /DRIVE RMD5 RMER1 RMER2 RMMR2 RMAS/
050652 051104 053111 020105 DH3: .ASCIZ /DRIVE REG ADR DATA/
050700 051104 053111 020105 DH4: .ASCIZ /DRIVE REG ADR GOOD BAD/
050737 044 046522 042101 DH6: .ASCIZ /$RMADR/
050746 051104 020126 046522 DH14: .ASCII /DRV RMCS1 RMCS2 RMD5 RMER1 /
051012 046522 051115 020062 .ASCIZ /RMMR2 RMER2 RMEC1 RMEC2/<CR><LF>
051051 122 053515 020103 DH15: .ASCII /RMWC RMBA RMDA RMAS RMLA /
051121 122 042115 020102 .ASCIZ /RMOB RMMR1 RMDT/<CR><LF>
051150 046522 047123 020040 DH16: .ASCIZ /RMSN RMOF RMDC RMHR STATUS/<CR><LF>
051222 .EVEN

```

.LIST BEX

```

9651
9652 051222 001342 000000 DT1: .WORD ATTN,0
9653
9654 051226 001216 034452 034454 DT2: .WORD DRIVE,RMERRS,RMERRS+2,RMERRS+4,RMERRS+6,ATTN,0
9655 051234 034456 034460 001342
9656 051242 000000
9657

```


CZRM880 RM03.2 PERF EXER
CZRM88.P11 21-NOV-77 15:34

MACY11 30(1046) 22-NOV-77 10:06 PAGE 181
ERROR MESSAGES

SEQ 0180

9658	051244	001216	041560	041562	DT3:	.WORD	DRIVE, RD. ADR, RD. WRD, 0
9659	051252	000000					
9660							
9661	051254	001216	042000	041776	DT4:	.WORD	DRIVE, WRT. AD, WRT. WD, RD. WRD, 0
9662	051262	041562	000000				
9663							
9664	051266	001270	000000		DT6:	.WORD	\$RMADR, 0
9665							
9666	051272	000234	000244	000246	DT14:	.WORD	\$RMCS1, \$RMCS2, \$RMDS, \$RMER1, \$RMMR2, \$RMER2, \$RMEC1, \$RMEC2, 0
9667	051300	000250	000274	000276			
9668	051306	000300	000302	000000			
9669							
9670	051314	000236	000240	000242	DT15:	.WORD	\$RMWC, \$RMBA, \$RMDA, \$RMAS, \$RMLA, \$RMDB, \$RMMR1, \$RMDT, 0
9671	051322	000252	000254	000256			
9672	051330	000260	000262	000000			
9673							
9674	051336	000264	000266	000270	DT16:	.WORD	\$RMSN, \$RMOF, \$RMDC, \$RMHR, \$STATUS, 0
9675	051344	000272	000016	000000			
9676							
9677							
9678							

.NLIST BEX

051352	051120	051505	047105	LIN2C:	.ASCIZ	/PRESENT ORDER = /
051373	040	050040	042522	LIN2P:	.ASCIZ	/ PREVIOUS ORDER = /
051417	052	042440	051122	LIN2S:	.ASCIZ	2* ERROR AT BAD TRACK/SECTOR2
051453	105	051122	051117	LINM3:	.ASCIZ	/ERROR AT C/
051466	052040	000		T:	.ASCIZ	/ T/
051471	120	042522	042523	LINN3:	.ASCIZ	/PRESENT ADDR = C/
051512	051440	000		S:	.ASCIZ	/ S/
051515	040	020040	051120	LIMP3:	.ASCIZ	/ PREV ADDR = C/
051536	052123	051101	020124	LINS3:	.ASCIZ	/START CYL = /
051553	040	042440	042116	LINEN3:	.ASCIZ	/ END CYL = /
051570	020040	041501	052524	LINA3:	.ASCIZ	/ ACTUAL CYL =
051610	020040	051124	020113	LINT3:	.ASCIZ	/ TRK = /
051621	040	046522	041504	LINCA3:	.ASCIZ	/ RMDC = /
051632	046522	040504	036440	LINDA3:	.ASCIZ	/RMDA = /
051642	046522	040502	036440	LINB3:	.ASCIZ	/RMBA = /
051652	020040	046522	041527	LINW3:	.ASCIZ	/ RMWC = /
051664	052123	051101	020124	LINST3:	.ASCIZ	/START TRK = /
051701	123	040524	052122	LINSS3:	.ASCIZ	/START SEC = /
051716	052502	043106	051105	LINM4:	.ASCIZ	/BUFFER ADDR = /
051735	040	051440	055111	LINS4:	.ASCIZ	/ SIZE = /
051747	040	040440	052103	LINX4:	.ASCIZ	/ ACTUAL NMBA WRDS XFRO =
052002	047507	042117	042040	LIND5:	.ASCIZ	/GOOD DATA = /
052017	040	041040	042101	LINB5:	.ASCIZ	/ BAD DATA = /
052035	040	053440	051117	LIMP5:	.ASCIZ	/ WORD POS = /
052053	110	040505	042504	LINS5:	.ASCIZ	/HEADER FROM ERROR SECTOR = /
052107	122	042515	030503	LINEP5:	.ASCIZ	/RMEC1 = /
052120	051040	042515	031103	LINEO5:	.ASCIZ	/ RMEC2 = /
052132	042523	052103	051117	LINB6:	.ASCIZ	/SECTOR IS ECC CORRECTABLE
052165	123	041505	047524	LINC6:	.ASCIZ	/SECTOR READ CORRECTLY
052214	047503	051122	041505	LING6:	.ASCIZ	/CORRECTED ON /
052232	051040	052105	044522	LINR6:	.ASCIZ	/ RETRIES/
052243	125	041516	051117	LINU06:	.ASCIZ	/UNCORRECTABLE AFTER /
052270	020040	047524	040524	LIN7M:	.ASCIZ	/ TOTAL MISPOS ERR =
052316	051117	042504	051522	LIN7O:	.ASCIZ	/ORDERS:/

M14

C2RM880 RM03/2 PERF EXER
C2RM88.P11 21-NOV-77 15:34MACY11 30(1046) 22-NOV-77 10:06 PAGE 182
ERROR MESSAGES

SEQ 0181

052326	052040	052117	046101	LIN7P:	.ASCIZ	/ TOTAL SEEKS = /
052346	052040	052117	046101	LIN7S:	.ASCIZ	/ TOTAL SKI ERR = /
052370	020040	051105	047522	LIN7T:	.ASCIZ	/ ERRORS: /
052402	020040	051127	051504	LIN7X:	.ASCIZ	/ WRDS XFR: /
052416	020040	051127	051504	LIN7R:	.ASCIZ	/ WRDS READ: /
052433	105	051122	051117	LIN8M:	.ASCIZ	/ERROR DURING RETRY/
052456	040504	040524	041440	LIN9B:	.ASCIZ	/DATA COMPARISON ERRORS/
052505	040	020040	020040	LIN9H:	.ASCIZ	/
052533	114	041517	020040	LIN9I:	.ASCIZ	/LOC GOOD BAD/(CR)<LF>
052563	114	041517	020040	LIN9E:	.ASCIZ	/LOC DATA DATA/(CR)<LF>
052603	124	052117	046101	LIN9C:	.ASCIZ	/LOC DATA/(CR)<LF>
052633	124	042510	042040	LIN10A:	.ASCIZ	/TOTAL COMPARE ERRORS = /
052662	051105	047522	020122	LIN10B:	.ASCIZ	/THE DATA COMPARED OK/(CR)<LF>
052716	044440	020116	040504	LIN10C:	.ASCIZ	/ERROR BURST BEGINS AT WORD /
052757	105	051122	051117	LIN10H:	.ASCIZ	/ IN DATA FIELD OF ERROR SECTOR/(CR)<LF>
053022	041505	020103	047503	LIN10I:	.ASCIZ	/ERROR WAS NOT IN THE DATA READ - /(CR)<LF>
053064	041505	020103	047503	LIN10J:	.ASCIZ	/ECC CORRECTION CAN'T BE PERFORMED/
053114	042101	051104	020040	LIN10K:	.ASCIZ	/ECC CORRECTION RESULTS/(CR)<LF>
053151	103	047117	042524	LIN11H:	.ASCIZ	/ADDR BAD CORRECTED /(CR)<LF>
053225	101	042104	020122	LIN11I:	.ASCIZ	/CONTENTS OF ERROR SECTOR (REPORTED ABOVE)/(CR)<LF>
053244	020040			LIN11J:	.ASCIZ	/ADDR DATA/(CR)<LF>
053246	040			LIN4SP:	.ASCIZ	/
053247	040	000		LINSP:	.ASCIZ	/
053251	122	052105	044522	LINSP0:	.ASCIZ	/
				LINX5:	.ASCIZ	/RETRIEVED BY A RDHD COMMAND/

.SBTTL TELETYPE MESSAGES

053305	104	044522	042526	UNMSG:	.ASCIZ	/DRIVE/
053313	040	043117	046106	UNOFF:	.ASCIZ	/ OFFLINE/
053324	047440	046116	047111	UNON:	.ASCIZ	/ ONLINE/
053334	047040	052117	041040	UNNOT:	.ASCIZ	/ NOT BEING TESTED/
053356	040440	051114	040505	UNASN:	.ASCIZ	/ ALREADY BEING TESTED/
053404	047040	052117	040440	NOTRM:	.ASCIZ	/ NOT AN RM03/RM02/
053426	047040	052117	050040	NOTPRS:	.ASCIZ	/ NOT PRESENT/
053443	040	047516	020124	NOTAVL:	.ASCIZ	/ NOT AVAILABLE/
053462	052440	051516	043101	NOTSAF:	.ASCIZ	/ UNSAFE/
053472	047125	052111	051440	SYSTAT:	.ASCIZ	/UNIT STATUS:/(CR)<LF><LF>
053512	046522	031460	000	RM03A:	.ASCIZ	/RM03/
053517	122	030115	000062	RM03B:	.ASCIZ	/RM02/
053524	051104	053111	020105	STAT:	.ASCIZ	/DRIVE PERFORMANCE SUMMARY/(CR)<LF>
053557	104	053122	050040	STAT:	.ASCIZ	/DRV PASS ORDERS SEEKS WRDS XFER WRDS READ /
053637	123	043117	020124	STAT:	.ASCIZ	/SOFT HARD SKI MISP OTHER/(CR)<LF>
053673	104	047117	006505	POONE:	.ASCIZ	/DONE/(CR)<LF><LF>
053703	007	043077	052101	DROPNG:	.ASCIZ	/07/? FATAL OR EXCESSIVE ERRORS/
053737	105	042116	047440	ENDPAS:	.ASCIZ	/END OF PASS/
053753	015	042412	042116	ENDTST:	.ASCIZ	/END OF TEST/
053771	015	042012	044522	DEASSG:	.ASCIZ	/DRIVE DEASSIGNED/
054014	005015	025052	025052	DRNUM:	.ASCIZ	/***** DRIVE #/
054041	040	052123	051101	ASCND:	.ASCIZ	/ STARTED/(CR)<LF>
054054	005015	020077	046047	NEDCLK:	.ASCIZ	/CR/<LF>/? 'L' OR 'P' CLOCK REQUIRED ON SYSTEM/(CR)<LF>
054126	000056			PERIOD:	.ASCIZ	/
054130	000077			QUES:	.ASCIZ	/
054132	047111	040526	044514	INVLD:	.ASCIZ	/INVALID COMMAND/(CR)<LF>
054154	005015	047105	042524	ENTCOM:	.ASCIZ	/CR/<LF>/ENTER COMMANDS: /(CR)<LF>
054201	105	052116	051105	ENTDRV:	.ASCIZ	/ENTER I.D. FOR DRV #/
054226	005015	047105	042524	ENTLMT:	.ASCIZ	/CR/<LF>/ENTER ADDRESS LIMITS FOR DRV #/

N14

CZRM880 RM03/2 PERF EXER
CZRM88.P11 21-NOV-77 15:34

MACY11 30(1046) 22-NOV-77 10:06 PAGE 183
TELETYPE MESSAGES

SEQ 0182

054267	105	052116	051105	ENTADR: .ASCIZ	ENTER BAD TRK/SEC ADRS FOR DRV #2
054330	000072			COLON: .ASCIZ	:/
054332	005015	040504	042524	DATEIS: .ASCIZ	<CR><LF>/DATE: /
054343	015	047412	042520	IDIS: .ASCIZ	<CR><LF>/OPERATOR I.D.: /
054365	015	005012	051104	MEDLIN: .ASCIZ	<CR><LF><LF>/DRV DRV I.D./<CR><LF>
054410	047516	042516	005015	NONE: .ASCIZ	/NONE/<CR><LF>
054417	077	044440	053116	BADENT: .ASCIZ	/? INVALID ENTRY/<CR><LF>
054441	123	051531	042524	BUSY: .ASCIZ	/SYSTEM BUSY.../<CR><LF>
054462	005015	051120	043517	INTDOM: .ASCII	<CR><LF>/PROGRAM INITIALIZATION COMPLETE/
054523	015	052012	050131		<CR><LF>/TYPE A 'CONTROL C' TO ENTER COMMANDS/<CR><LF><LF>
054575	015	043012	044501	MERR1: .ASCIZ	<CR><LF>/FAIL TO RETRIEVE BAD SPOT FILE ON /
054643	015	047412	042526	MERR2: .ASCIZ	<CR><LF>/OVER 16 BAD SECTORS ARE DETECTED ON /

054714

.EVEN

.LIST BEX

;PARAMETER ENTRY TABLE

9679					
9680					
9681					
9682					
9683	054714	055042	020000	001462	PAR1,8192,MAXDL
9684	054722	055052	177777	001466	PAR2,-1,INTRVL
9685	054730	055222	177777	001460	PAR3,-1,PASCNT
9686	054736	055062	000017	001510	PAR4,15,PATTEN
9687	054744	055162	000001	001476	PAR11,1,WCSEL
9688	054752	055172	000007	001500	PAR14,7,RATIO
9689	054760	055212	000001	001506	PAR16,1,ENDET
9690	054766	055152	000001	001474	PAR10,1,FORMAT
9691					PAR15,1,AUTOCK
9692	054774	055232	000001	001504	PAR20,1,NOTPRT
9693	055002	055242	000001	001512	PAR21,1,HEADER
9694	055010	000000			0
9695					;RANDOM ADDRESS FLAG
9696					;TABLE TERMINATOR
9697					

.NLIST BEX

055012	047105	042524	020122	ASKPAR: .ASCIZ	/ENTER PARAMETERS:
055036	027440	000040		SLASH: .ASCIZ	/ /
055042	044523	042532	020040	PAR1: .ASCIZ	/SIZE /
055052	044515	052516	042524	PAR2: .ASCIZ	/MINUTE /
055062	040520	052124	051105	PAR3: .ASCIZ	/PATTERN/
055072	040515	041530	046131	PAR4: .ASCIZ	/MAXCYL /
055102	044515	041516	046131	PAR5: .ASCIZ	/MINCYL /
055112	040515	052130	045522	PAR6: .ASCIZ	/MAXTRK /
055122	044515	052116	045522	PAR7: .ASCIZ	/MINTRK /
055132	040515	051530	041505	PAR8: .ASCIZ	/MAXSEC /
055142	044515	051516	041505	PAR9: .ASCIZ	/MINSEC /
055152	047506	046522	052101	PAR10: .ASCIZ	/FORMAT /
055162	040522	042116	046517	PAR11: .ASCIZ	/RANDOM /
055172	040522	044524	020117	PAR14: .ASCIZ	/RATIO /
055202	044103	041505	020113	PAR15: .ASCIZ	/CHECK /
055212	047105	044504	043516	PAR16: .ASCIZ	/ENDING /
055222	040520	051523	051505	PAR19: .ASCIZ	/PASSES /
055232	042515	051523	043501	PAR20: .ASCIZ	/MESSAGE /
055242	042510	042101	051105	PAR21: .ASCIZ	/HEADER /

.EVEN

.LIST BEX

;PARAMETER TABLE POINTERS FOR ADDRESS LIMITS

TABLE:	.WORD	TABLE0	;PARAMETER TABLE FOR DRIVE 0
	.WORD	TABLE1	;PARAMETER TABLE FOR DRIVE 1
	.WORD	TABLE2	;PARAMETER TABLE FOR DRIVE 2
	.WORD	TABLE3	;PARAMETER TABLE FOR DRIVE 3
	.WORD	TABLE4	;PARAMETER TABLE FOR DRIVE 4
	.WORD	TABLE5	;PARAMETER TABLE FOR DRIVE 5
	.WORD	TABLE6	;PARAMETER TABLE FOR DRIVE 6
	.WORD	TABLE7	;PARAMETER TABLE FOR DRIVE 7

;PARAMETER TABLE FOR ADDRESS LIMITS

TABLE0:	.WORD	PAR5,0,MINCYL+DRIVE0
	.WORD	PAR4,0,MAXCYL+DRIVE0
	.WORD	PAR7,4.,MINTRK+DRIVE0
	.WORD	PAR6,4.,MAXTRK+DRIVE0
	.WORD	PAR9,31.,MINSEC+DRIVE0
	.WORD	PAR8,31.,MAXSEC+DRIVE0,0

TABLE1:	.WORD	PAR5,0,MINCYL+DRIVE1
	.WORD	PAR4,0,MAXCYL+DRIVE1
	.WORD	PAR7,4.,MINTRK+DRIVE1
	.WORD	PAR6,4.,MAXTRK+DRIVE1
	.WORD	PAR9,31.,MINSEC+DRIVE1
	.WORD	PAR8,31.,MAXSEC+DRIVE1,0

TABLE2:	.WORD	PAR5,0,MINCYL+DRIVE2
	.WORD	PAR4,0,MAXCYL+DRIVE2
	.WORD	PAR7,4.,MINTRK+DRIVE2
	.WORD	PAR6,4.,MAXTRK+DRIVE2
	.WORD	PAR9,31.,MINSEC+DRIVE2
	.WORD	PAR8,31.,MAXSEC+DRIVE2,0

TABLE3:	.WORD	PAR5,0,MINCYL+DRIVE3
	.WORD	PAR4,0,MAXCYL+DRIVE3
	.WORD	PAR7,4.,MINTRK+DRIVE3
	.WORD	PAR6,4.,MAXTRK+DRIVE3
	.WORD	PAR9,31.,MINSEC+DRIVE3
	.WORD	PAR8,31.,MAXSEC+DRIVE3,0

TABLE4:	.WORD	PAR5,0,MINCYL+DRIVE4
	.WORD	PAR4,0,MAXCYL+DRIVE4
	.WORD	PAR7,4.,MINTRK+DRIVE4
	.WORD	PAR6,4.,MAXTRK+DRIVE4

9638				
9639				
9700				
9701				
9702	055252	055272		
9703	055254	055340		
9704	055256	055406		
9705	055260	055454		
9706	055262	055522		
9707	055264	055570		
9708	055266	055636		
9709	055270	055704		
9710				
9711				
9712				
9713	055272	055102	000000	043132
9714	055300	055072	000000	043130
9715	055306	055122	000004	043136
9716	055314	055112	000004	043134
9717	055322	055142	000037	043142
9718	055330	055132	000037	043140
9719	055336	000000		
9720				
9721	055340	055102	000000	043436
9722	055346	055072	000000	043434
9723	055354	055122	000004	043442
9724	055362	055112	000004	043440
9725	055370	055142	000037	043446
9726	055376	055132	000037	043444
9727	055404	000000		
9728				
9729	055406	055102	000000	043742
9730	055414	055072	000000	043740
9731	055422	055122	000004	043746
9732	055430	055112	000004	043744
9733	055436	055142	000037	043752
9734	055444	055132	000037	043750
9735	055452	000000		
9736				
9737	055454	055102	000000	044246
9738	055462	055072	000000	044244
9739	055470	055122	000004	044252
9740	055476	055112	000004	044250
9741	055504	055142	000037	044256
9742	055512	055132	000037	044254
9743	055520	000000		
9744				
9745	055522	055102	000000	044552
9746	055530	055072	000000	044550
9747	055536	055122	000004	044556
9748	055544	055112	000004	044554

CZRM880 RM03/2 PERF EXER
CZRM88.P11 21-NOV-77 15:34

MACY11 30(1046) 22-NOV-77 10:06 PAGE 185
TELETYPE MESSAGES

SEQ 0184

9749	055552	055142	000037	044562	.WORD	PAR9,31.,MINSEC+DRIVE4
9750	055560	055132	000037	044560	.WORD	PAR8,31.,MAXSEC+DRIVE4,0
9751	055566	000000				
9752						
9753	055570	055102	000000	045056	TABLE5: .WORD	PAR5,0,MINCYL+DRIVE5
9754	055576	055072	000000	045054	.WORD	PAR4,0,MAXCYL+DRIVE5
9755	055604	055122	000004	045062	.WORD	PAR7,4.,MINTRK+DRIVE5
9756	055612	055112	000004	045060	.WORD	PAR6,4.,MAXTRK+DRIVE5
9757	055620	055142	000037	045066	.WORD	PAR9,31.,MINSEC+DRIVE5
9758	055626	055132	000037	045064	.WORD	PAR8,31.,MAXSEC+DRIVE5,0
9759	055634	000000				
9760						
9761	055636	055102	000000	045362	TABLE6: .WORD	PAR5,0,MINCYL+DRIVE6
9762	055644	055072	000000	045360	.WORD	PAR4,0,MAXCYL+DRIVE6
9763	055652	055122	000004	045366	.WORD	PAR7,4.,MINTRK+DRIVE6
9764	055660	055112	000004	045364	.WORD	PAR6,4.,MAXTRK+DRIVE6
9765	055666	055142	000037	045372	.WORD	PAR9,31.,MINSEC+DRIVE6
9766	055674	055132	000037	045370	.WORD	PAR8,31.,MAXSEC+DRIVE6,0
9767	055702	000000				
9768						
9769	055704	055102	000000	045666	TABLE7: .WORD	PAR5,0,MINCYL+DRIVE7
9770	055712	055072	000000	045664	.WORD	PAR4,0,MAXCYL+DRIVE7
9771	055720	055122	000004	045672	.WORD	PAR7,4.,MINTRK+DRIVE7
9772	055726	055112	000004	045670	.WORD	PAR6,4.,MAXTRK+DRIVE7
9773	055734	055142	000037	045676	.WORD	PAR9,31.,MINSEC+DRIVE7
9774	055742	055132	000037	045674	.WORD	PAR8,31.,MAXSEC+DRIVE7,0
9775	055750	000000				
9776						
9777						
9778						
9779						
9780						
9781						
9782						
9783						
9784						
9785						
9786						
9787						
9788	055752	104401	056074			
9789	055756	104411				
9790	055760	012605				
9791	055762	112537	001320			
9792	055766	112537	001321			
9793	055772	112537	001322			
9794	055776	112537	001323			
9795	056002	112537	001324			
9796	056006	112537	001325			
9797	056012	112537	001326			
9798	056016	112537	001327			
9799	056022	005037	001330			
9800	056026	104401	056113			
9801	056032	104411				
9802	056034	012605				
9803	056036	112537	001332			
9804	056042	112537	001333			

;*****

;ROUTINE TO GET THE DATE AND THE OPERATOR FROM THE OPERATOR

;CALL:

JSR PC,OPRDAT

RETURN

;NOTE: THIS ROUTINE IS ENTERED ONLY AT INITIAL START

OPRDAT:	TYPE	,ENTDAT	;'ENTER DATE'
	RDLIN		;'READ THE ENTRY
	MOV	(SP)+,R5	;'PUT THE ENTRY ADDRESS INTO R5
	MOVB	(R5)+,DATE	;'STORE THE DATE
	MOVB	(R5)+,DATE+1	;'STORE THE DATE
	MOVB	(R5)+,DATE+2	;'STORE THE DATE
	MOVB	(R5)+,DATE+3	;'STORE THE DATE
	MOVB	(R5)+,DATE+4	;'STORE THE DATE
	MOVB	(R5)+,DATE+5	;'STORE THE DATE
	MOVB	(R5)+,DATE+6	;'STORE THE DATE
	MOVB	(R5)+,DATE+7	;'STORE THE DATE
	CLR	DATE+8.	;'SET TERMINATOR
	TYPE	,ENTID	;'ENTER OPERATOR I.D.'
	RDLIN		;'READ THE ENTRY
	MOV	(SP)+,R5	;'ENTRY ADDRESS
	MOVB	(R5)+,OPERID	;'STORE THE I.D.
	MOVB	(R5)+,OPERID+1	;'STORE THE I.D.

CZRM380 RM03/2 PERF EXER
CZRM88.P11 21-NOV-77 15:34

MACY11 30(1046) 22-NOV-77 10:06 PAGE 186
TELETYPE MESSAGES

SEQ 0185

```

9805 056046 112537 001334      MOV8      (R5)+,OPERID+2  ;STORE THE I.D.
9806 056052 112537 001335      MOV8      (R5)+,OPERID+3  ;STORE THE I.D.
9807 056056 112537 001336      MOV8      (R5)+,OPERID+4  ;STORE THE I.D.
9808 056062 112537 001337      MOV8      (R5)+,OPERID+5  ;STORE THE I.D.
9809 056066 005037 001340      CLR        OPERID+6      ;SET TERMINATOR
9810 056072 000207              RTS          PC          ;RETURN
9811
9812 056074 005015 047105 042524  ENTDAT: .ASCIZ  <CR><LF>/ENTER DATE: /
9813 056102 020122 040504 042524
9814 056110 020072 000
9815 056113 105 052116 051105  ENTID: .ASCIZ  /ENTER OPERATOR I.D.: /
9816 056120 047440 042520 040522
9817 056126 047524 020122 027111
9818 056134 027104 020072 000
9819 056142
9820
9821 .EVEN
9822 .SBTTL  ROUTINE TO SIZE MEMORY
9823
9824 ;*****
9825 ;*CALL:
9826 ;*      JSR      PC,$SIZE
9827 ;*      RETURN
9828 ;*$LSTAD WILL CONTAIN THE LAST AVAILABLE MEMORY LOCATION
9829 $SIZE:  MOV      R0,-(SP)      ;;SAVE R0 ON THE STACK
9830        MOV      R1,-(SP)      ;;SAVE R1 ON THE STACK
9831        MOV      2*ERRVEC,-(SP) ;;SAVE PRESENT ERROR VECTOR PS & PC
9832        MOV      2*ERRVEC+2,-(SP)
9833        MOV      SP,R0        ;;SAVE THE STACK POINTER
9834        ;;SET THE ERRVEC PS TO THE PRESENT PS
9835        TRAP
9836        MOV      (SP)+,2*ERRVEC+2 ;;PUSH OLD PSW AND PC ON STACK
9837        MOV      2*ERRVEC+2,2*ERRVEC ;;SAVE THE PSW IN 2*ERRVEC+2
9838        MOV      2*ERRVEC,2*ERRVEC+2 ;;SET FOR TIMEOUT
9839        MOV      20000,R1      ;;FIRST ADDRESS
9840        1$:      TST      (R1)      ;;TEST THIS ADDRESS
9841        TST      (R1)+          ;;STEP TO NEXT ADDRESS
9842        BR       1$            ;;TRY ANOTHER
9843        2$:      SUB      2,R1      ;;DROP BACK
9844        MOV      R0,SP          ;;RESTORE THE STACK
9845        MOV      (SP)+,2*ERRVEC+2 ;;RESTORE ERROR VECTOR
9846        MOV      (SP)+,2*ERRVEC
9847        MOV      R1,$LSTAD      ;;LAST ADDRESS
9848        MOV      (SP)+,R1      ;;RESTORE R1
9849        MOV      (SP)+,R0      ;;RESTORE R0
9850        RTS      PC
9851 $LSTAD: .WORD      0          ;;CONTAINS THE LAST ADDRESS
9852
9853 .SBTTL  BUSADR - GET BUS ADDRESS AND VECTOR ADDRESS FOR RH11
9854 ;THIS ROUTINE IS USED TO INSURE THE BUS ADDRESS
9855 ;OF THE RH11 IS SETUP FOR THE PROPER ADDRESS.
9856 ;IT WILL ALSO READ THE ADDRESS FROM THE TTY IF
9857 ;REQUIRED.
9858 ;NOTE: THIS ROUTINE DESTROYS R0-R4
9859 ;CALL
9860 ;
9861 ;      JSR      PC,BUSADR

```

```

9861 ; RETURN
9862
9863 056240 005737 001356 BUSADR: TST CHGADR ; INPUT FROM TTY REQUESTED?
9864 056244 002036 BGE 3$ ; NO--BRANCH
9865 056246 005037 001356 CLR CHGADR ; YES--CLEAR THE REQUEST FLAG
9866 056252 104401 001201 TYPE $CRLF ; TYPE A CR-LF
9867 056256 012700 001270 1$: MOV $RMADR, R0 ; FIRST ADDRESS
9868 056262 104401 056360 TYPE MRMCS1 ; "RMCS1="
9869 056266 011046 MOV (R0), -(SP) ; PRESENT RMCS1 ADDRESS
9870 056270 104402 TPOC ; TYPE IT
9871 056272 104401 053246 TYPE ,LINSF ; 2 SPACES
9872 056276 104411 RDLIN ; GET THE ENTRY
9873 056300 012601 MOV (SP)+, R1 ; ADDRESS OF ASCII TEXT
9874 056302 004537 056402 JSR R5, CK.NUM ; ENTER AND STORE THE NEW ADDRESS
9875 056306 000763 BR 1$ ; ERROR EXIT
9876 056310 012700 001272 2$: MOV $RMVEC, R0 ; VECTOR ADDRESS
9877 056314 104401 056371 TYPE MRHVEC ; "RHVEC="
9878 056320 011046 MOV (R0), -(SP) ; PRESENT RH11 VECTOR ADDRESS ON THE STACK
9879 056322 104402 TPOC ; TYPE IT
9880 056324 104401 053246 TYPE ,LINSF ; 2 SPACES
9881 056330 104411 RDLIN ; READ THE ENTRY
9882 056332 012601 MOV (SP)+, R1 ; ASCII TEXT ADDRESS
9883 056334 004537 056402 JSR R5, CK.NUM ; ENTER AND STORE NEW ADDRESS
9884 056340 000763 BR 2$ ; ERROR EXIT
9885 056342 012700 001270 3$: MOV $RMADR, R0 ; FIRST ADDRESS OF NEW PARAMETERS
9886 056346 012701 034620 MOV $RMADR, R1 ; FIRST ADDRESS OF WHERE TO PUT THEM
9887 056352 012021 MOV (R0)+, (R1)+ ; BUS ADDRESS
9888 056354 012021 MOV (R0)+, (R1)+ ; VECTOR ADDRESS
9889 056356 000207 RTS PC ; RETURN
9890
9891 056360 046522 051503 020061 MRMCS1: .ASCIZ @RMCS1 = @
9892 056366 020075 000
9893 056371 122 053110 041505 MRHVEC: .ASCIZ @RHVEC = @
9894 056376 036440 000040
9895
9896 ;SMTTL CK.NUM - CHECK NUMBER (OCTAL)
9897 ;THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL CHARACTERS
9898 ;AND FORMS AN OCTAL NUMBER IN R2
9899 ;CALL:
9900 ; MOV $ADR, R1 ; ADDRESS OF ASCII STRING
9901 ; JSR R5, CK.NUM ; R5 CHANGED
9902 ; RET ; ERROR EXIT
9903 ; RET ; NORMAL EXIT
9904 056402 010246 CK.NUM: MOV R2, -(SP) ; SAVE R2
9905 056404 010346 MOV R3, -(SP) ; SAVE R3
9906 056406 010446 MOV R4, -(SP) ; SAVE R4
9907 056410 012703 000006 MOV $6, R3 ; MAX OCTAL DIGITS IN THE NUMBER
9908 056414 005002 CLR R2 ; FINAL OCTAL VALUE
9909 056416 112104 1$: MOV B (R1)+, R4 ; GET CURRENT POINTED BYTE
9910 056420 001424 BEQ 3$ ; BRANCH IF TERMINATOR DETECTED
9911 056422 120427 000060 CMPB R4, #'0 ; SMALLER THAN ASCII-0 ?
9912 056426 103425 BLO 5$ ; YES, ERROR EXIT
9913 056430 120427 000067 CMPB R4, #'7 ; LARGER THAN ASCII-7 ?
9914 056434 101022 BHI 5$ ; YES, ERROR EXIT
9915 056436 006302 ASL R2 ; SHIFT LEFT
9916 056440 103420 BCS 5$

```

CZRM880 RM03/2 PERF EXER
CZRM88.P11 21-NOV-77 15:34

MACY11 30(1046) 22-NOV-77 10:06 PAGE 188
CK.NUM - CHECK NUMBER (OCTAL)

SEQ 0187

```

9917 056442 006302      ASL      R2      ; ONE
9918 056444 103416      BCS      S$      ;
9919 056446 006302      ASL      R2      ; OCTAL DIGIT
9920 056450 103414      BCS      S$      ; ERROR IF CARRY BIT SET
9921 056452 042704 177770 BIC      #177770,R4 ; CHOP OFF HIGHER BITS
9922 056456 060402      ADD      R4,R2    ; APPENDING CURRENT DIGIT TO NUMBER
9923 056460 005303      DEC      R3      ; DECREMENT BYTE COUNT
9924 056462 001401      BEQ      2$      ; BRANCH IF LAST BYTE
9925 056464 000754      BR       1$      ; LOOPING BACK
9926 056466 112104      2$: MOVB    (R1)+,R4 ; CHECK TERMINATOR
9927 056470 001004      BNE      S$      ; ERROR EXIT
9928 056472 005702      3$: TST     R2      ; FINAL VALUE = 0
9929 056474 001401      BEQ      4$      ; YES, THEN NOT REPLACE THE ORIGINAL VALUE
9930 056476 010210      MOV      R2,(R0)  ; REPLACE THE ORIGINAL VALUE
9931 056500 005725      4$: TST     (R5)+  ; ADJUST FOR NORMAL RETURN
9932 056502 012604      5$: MOV      (SP)+,R4 ; RESTORE R4
9933 056504 012603      MOV      (SP)+,R3 ; RESTORE R3
9934 056506 012602      MOV      (SP)+,R2 ; RESTORE R2
9935 056510 000205      RTS      R5      ; EXIT
9936
9937 056512 000400      CYLDER: .BLKW 256.    ; ONE SECTOR WORD CTR MAX SIZE
9938 057512 057512      ENDPGM  =
9939
9940
9941
9942

```

.NLIST BEX

```

057512 005015 055103 046522 TITLE: .ASCII <CR><LF>/CZRM880/<CR><LF>
057525 122 030115 027463 .ASCII2 0RM03/RM02 PERFORMANCE EXERCISER 2<CR><LF><LF>
057571 015 052012 020117 LOADRV: .ASCII <CR><LF>/TO TEST DRIVE 0, REPLACE THE 'XXDP' PACK ON DRIVE 0/<CR><LF>
057660 044527 044124 040440 .ASCII /WITH ANOTHER PACK, CLEAR MEMORY LOCATION 40,/<CR><LF>
057736 047101 020104 042522 .ASCII /AND RESTART THE PROGRAM/<CR><LF>
057770 005015 054523 052123 NOLOAD: .ASCII2 <CR><LF>/SYSTEM HAS 16K MEMORY, 'XXDP' LOADER WILL BE OVERWRITTEN/<CR><L

```

.LIST BEX

9943
9944

000001

.END

CZRM880 RM03 2 PERF EXER
CZRM88.P11 21-NOV-77 15:34MACY11 30(1046) 22-NOV-77 10:06 PAGE 190
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0188

ABASE = 176700	2271#	2561	2602			
ABNRML 027064	3702	6735#				
ACDW1 = 000000	2561	2604				
ACDW2 = 000000	2561	2605				
ACK = 000123	2456#					
ACPUOP= 000000	2561	2576				
ACTDRV 034536	8171#	8444*	8495*	8821*	9830*	9087
ACTSTR 034537	8177#	9089*	9103*			
ADDW0 = 000000	2561					
ADDW1 = 000000	2561					
ADDW10= 000000	2561					
ADDW11= 000000	2561					
ADDW12= 000000	2561					
ADDW13= 000000	2561					
ADDW14= 000000	2561					
ADDW15= 000000	2561					
ADDW2 = 000000	2561					
ADDW3 = 000000	2561					
ADDW4 = 000000	2561					
ADDW5 = 000000	2561					
ADDW6 = 000000	2561					
ADDW7 = 000000	2561					
ADDW8 = 000000	2561					
ADDW9 = 000000	2561					
ADEVCT= 000000	2561	2567				
ADEVN = 000000	2561	2603				
AENV = 000000	2561	2572				
AENVN = 000000	2561	2573				
AFATAL= 000000	2561	2564				
AMAOR1= 000000	2561	2589				
AMAOR2= 000000	2561	2593				
AMAOR3= 000000	2561	2596				
AMAOR4= 000000	2561	2599				
AMAMS1= 000000	2561	2583				
AMAMS2= 000000	2561	2591				
AMAMS3= 000000	2561	2594				
AMAMS4= 000000	2561	2597				
AMSGAD= 000000	2561	2569				
AMSGLG= 000000	2561	2570				
AMSGTY= 000000	2561	2563				
AMTYP1= 000000	2561	2584				
AMTYP2= 000000	2561	2592				
AMTYP3= 000000	2561	2595				
AMTYP4= 000000	2561	2598				
AGE = 001000	2361#					
APASS = 000000	2561	2566				
APRIOR= 000000	2561					
APTCSU= 000040	7394	7500#				
APTENV= 000001	7302	7387	7456	7498#		
APTSIZ= 000200	3312	7497#				
APTSP0= 000100	7389	7458	7499#			
ASGND 054041	3598	9678#				
ASGN1 024470	6264#					
ASGN2 024544	3480	6263	6276#			
ASGN3 024626	6273	6284	6291#			
ASGN4 024740	6292	6314#				

H15

CZRM880 RM03/2 PERF EXER
CZRM88.P11 21-NOV-77 15:34

MACY11 30(1046) 22-NOV-77 10:06 PAGE 191
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0189

ASGN6	024742	6299	6315#												
ASGN7	024754	6298	6317#												
ASKPAR	055012	3410	9697#												
ASNERR	026734	6275	6289	6316	6326	6352	6372	6692#							
ASNLST	001544	2751#	3554	3605*	5956	5963	6266	6282	6291	6341	6343*	6365	6374	6388	
		6711*	6725	6807*	6810										
ASNMSG	026754	6265*	6270*	6281*	6315*	6320*	6322*	6324*	6351*	6371*	6700#				
ASSIGN	024462	6262#	6331	6414	6419	6423									
ASWREG=	000000	2561	2574												
ATA	= 100000	2348#													
ATABIT	034606	3605	5963	6266	6282	6291	6341	6343	6365	6388	6711	6807	8243#	8411	
		8507	8613	9006	9028	9036	9037	9057	9147						
ATESTN=	000000	2561	2565												
ATY	001342	2627#	7282*	9652	9654										
ATY0	= 000001	2374#													
ATY1	= 000002	2375#													
ATY2	= 000004	2376#													
ATY3	= 000010	2377#													
ATY4	= 000020	2378#													
ATY5	= 000040	2379#													
ATYE	= 000100	2380#													
ATYE	= 000200	2381#													
UNIT	= 000000	2561	2568												
USWR	= 000000	2561	2575												
BUCK	001502	2704#													
AVAIL	001612	2761#	3557	3599	3603*	3645	3648	3681	3705						
AVECT1=	000254	2272#	2561	2600											
AVECT2=	000000	2561	2601												
A16	= 000400	2284#													
A17	= 001000	2285#													
BADENT	054417	6675	9678#												
BADSEC	001362	2636#	3701*	5429*	5503	6084	6096	6108	6120	6132					
BAT	= 000010	2302#													
BEGCOO	001516	2732#	6452	6453											
BEGPAT	001514	2731#	6455												
BEGSIZ	001520	2739#	6457	6458											
B170	= 000001	2256#	6588	6617											
B1700	= 000001	2246#	2256	4403	5041	5162	6339								
B1701	= 000002	2245#	2255	3778	4908	8487	8742								
B1702	= 000004	2244#	2254	4452	5037	5190	5197	6494							
B1703	= 000010	2243#	2253	3871	4238	8964	9264								
B1704	= 000020	2242#	2252	3871	3883	8999									
B1705	= 000040	2241#	2251	4339	8333	8763	8979								
B1706	= 000100	2240#	2250	3952	3953	4043	4058	4900	5674	6256	7161	8420	8478	8552	
		8861	9324												
B1707	= 000200	2239#	2249	27E4	3886	4900	8420	8687	8842	8964	8979	8999	9032	9296	
B1708	= 000400	2238#	2248	3860	8420										
B1709	= 001000	2237#	2247	3776	9138										
B171	= 000002	2255#	6613	6627											
B1710	= 002000	2236#	3774	3898	3906	7290	8744								
B1711	= 004000	2235#	3772	3901	8380	8514	8708	9073							
B1712	= 010000	2234#	3770	3909	3952	3953	3972	4464	5044	5662	8375	8402	9420	9489	
		8522	8702	8725	8740	8754	8989	8991	9203	9259	9325				
		8889	8915	4210	4306	6987	7297	8472	8937	9206					
		8932#	8951	3753	3778	3874	3877	3915	4033	4084	4128	4168	4274	4299	
		4481	4551	6484	8519	8924	9068	9168	9214	9218	9322				

CRMB80 RM03 2 PERF EXER
CRMB80.P11 21-NOV-77 15:34

MACY11 30(1046) 22-NOV-77 10:06 PAGE 192
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0190

BIT15 = 100000	2231#	3749	3917	3952	4012	4056	4306	4481	4551	8472	8484	8487	8489
	8519	8522	8708	8742	8744	8861	8964	8979	9989	8999	9068	9138	9168
	9175												
BIT2 = 000004	2254#	3780	9175										
BIT3 = 000010	2253#	3892											
BIT4 = 000020	2252#	4134											
BIT5 = 000040	2251#	3895											
BIT6 = 000100	2250#	3972	5523										
BIT7 = 000200	2249#	3455	4174	4874									
BIT8 = 000400	2248#	4	4113	4118	4148	4153	4306						
BIT9 = 001000	2247#	3904	4306										
BLKADR 002102	2807#	3361	3508	5961	6302	6308	6345	6368	7566				
BPTVEC= 000014	2263#												
BUFTBL 001700	2773#	3380*	3381*	3382*	3385*	3386*	3388*	3389*	3392*	3396	3398	3399	4935
	4937	4951*	4977	4978	4990*	4995*	5000	5001	5014*				
BUSADR 056240	3352	9863#											
BUSY 054441	6199	9678#											
CFLAG 001360	2635#	3371*	3540	6202*	6208	6525*	6535	6652*	6664	6680*	7174*	7231*	
CHGADR 001356	2634#	3272*	3274*	3478	3497	7563*	9863	9865*					
CHKADR 017172	3736	5167	5213#										
CI1 036114	8542	8570#											
CI3 036222	8544	8593#											
CI4 036322	8533	8615#											
CI5 036700	8592	8614	8692#	8701									
CI6 036722	8631	8641	8643	8645	8697#								
CI7 036736	8474	8551	8582	8586	8590	8598	8608	8612	8623	8630	8636	8640	8654
	8660	8664	8674	8685	8700	8702#	8816	8846	9018	9308			
CI7B 036760	8710#	9022											
CI8 037036	8713	8724#	9160										
CKBUS 012732	3756	4434#											
CKCLK 022632	3424	5917#	7557										
CKCLK1 022726	5919	5930#											
CKCLK2 022774	5931	5939#											
CKCLK3 023024	5929	5938	5946#										
CKEPR 012640	3755	4415#											
CKFMT 010712	3885	4113#											
CKHCE 011106	3888	4148#											
CKSWR = 104407	7281	7312	8052#										
CK.CHR 030226	7043#	7080	7094										
CK.DEC 030200	7021#	7049											
CK.DIG 030300	6666	7074#											
CK.NUM 056402	987	9883	9904#										
CK.OCT 030152	7003#	7051											
CLKFLG 001310	2618#	5917*	5922*	5933*	6142								
CLOCK 023766	5925	5935	6163#										
CLR = 000040	2304#												
CLRDPB 025432	3509	6303	6431#										
CLRQUE 042560	8315	8762	9392#										
CMCNT 001410	2651#	4402*	4406*	4461*	4462*	4472	4474	4475*	4478*	4529	4532	4535*	4559*
CMCYL 001412	2652#	4463*	4464*	4465*	4482	4547*	4548	4552					
CMDAT 013270	4480	4490	4494#	4536	4538								
CMHED 013216	4481#												
CMPAR 013016	3759	4452#											
CMPARD 013034	4001	4456#											
CMPLMT 001472	2688#	4410	4467										
CMPPES 017666	3576	3634	3683	3709	5324#	5327							

SEQ 0191

DRIVE3	044136	2810	9598#	9737	9738	9739	9740	9741	9742						
DRIVE4	044442	2811	9606#	9745	9746	9747	9748	9749	9750						
DRIVE5	044746	2812	9614#	9753	9754	9755	9756	9757	9758						
DRIVE6	045252	2813	9622#	9761	9762	9763	9764	9765	9766						
DRIVE7	045556	2814	9630#	9769	9770	9771	9772	9773	9774						
DRNUM	054014	3788	3814	3842	6717	6775	6783	9678#							
DROP	026764	3802	3856	6709#	6739										
DROPNG	053703	6716	9678#												
DRQ	= 004000	2397#													
DRVACT	034462	8089#	8468	8648*	8695*	8717*	8729	8747*	8835*	8929	8962	9993*	9000*	9015	
		9027*	9141*												
DRVCLR=	000111	2451#													
DRVER	010660	3919	4103#												
DRVINT	035050	8335	8369#	8456	9061										
DRVPRM	025632	6304	6473#												
DRVQUE	042656	8463	8476	8873	9432#										
DRVSTA	034472	3441	3506	6297	8103#	8325*	8326*	8327*	8328*	8338*	8370*	8379*	8417*	8422*	
		8450	8458	8482	8516	8520*	8649*	8766*	8914	8922*	8935	8987*	8994*	9007	
		9063	9163*												
DRVTyp	034502	3444	3458	3461	6317	6494	6497	8118#	8371*	8386*	8391*	8396*	8485	8767*	
DRY	= 000200	2340#													
DSWR	= 177570	2177#	2543	3299											
DTE	= 010000	2364#													
DTEER	011516	3911	4224#												
DTUW	034604	8237#	8321	8535	8591*	8722*	8735	8749	8751*	8760*	8824	8836*	9134	9144*	
		9212													
DT00	= 000001	2388#													
DT01	= 000002	2389#													
DT02	= 000004	2390#													
DT03	= 000010	2391#													
DT04	= 000020	2392#													
DT05	= 000040	2393#													
DT06	= 000100	2394#													
DT07	= 000200	2395#													
DT08	= 000400	2396#													
DT1	051222	3225	9652#												
DT14	051272	5513	9666#												
DT15	051314	5518	9670#												
DT16	051336	5521	9674#												
DT2	051226	3232	3253	9654#											
DT3	051244	3239	9658#												
DT4	051254	3246	9661#												

[illegible]

SEQ 0194

[illegible]

SEQ 0195

[illegible]

CZRM860 RM03 2 PERF EXER
CZRM88.P11 21-NOV-77 15:34

MACY11 30(1046) 22-NOV-77 10:06 PAGE 198
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0196

LINE28	020650	5504	5507#												
LINE3	021070	3798	3832	3852	3863	3949	4018	4040	4081	4106	4126	4166	4207	4272	
		4287	4303	4353	4426	5554#									
LINE3A	021076	4334	4371	4593	5560#										
LINE3B	021104	4323	5566#												
LINE3C	021116	4192	5573#												
LINE3D	021126	4444	5533	5579#											
LINE3E	021174	4235	5593#												
LINE3F	021262	4253	5612#												
LINE4	021540	3799	3833	3853	3864	3950	4019	4041	4082	4127	4167	4208	4236	4273	
		4288	4304	4335	4372	4427	4445	4594	5534	5674#					
LINE5	021630	4020	4042	4086	4130	4170	4276	4291	5693#						
LINE5A	021730	4083	4131	4171	4194	4277	4292	5714#							
LINE5B	021776	3992	5729#												
LINE6	022040	3955	5742#												
LINE6A	022052	3998	5749#												
LINE6B	022060	3991	5755#												
LINE6C	022066	3939	4025	4051	4060	4093	4138	4178	4214	4242	4312	4342	4358	5761#	
LINE6D	022074	3941	3996	4027	4096	4141	4181	4217	4245	4314	4345	4360	5767#		
LINE7	022150	3801	3835	3855	3866	3942	3988	4006	4065	4068	4094	4097	4108	4139	
		4142	4179	4182	4215	4218	4243	4255	4264	4279	4294	4315	4343	4361	
		4429	4447	4631	4888	5791#									
LINE7A	022276	4197	4326	5819#											
LINE8	022416	3985	4062	4887	5845#										
LINE6	052214	5761	9678#												
LINXOV	027527	4666	4758	5704	6846	6856#									
LINX3	05145	5554	9678#												
LINX4	051716	5676	9678#												
LINX3	051471	5560	9678#												
LINOC*	022430	4378	4381	4384	4387	4390	4393	4396	4399	4610	4613	4616	4724	4727	
		4730	4736	4739	4742	4772	4777	5543	5583	5586	5616	5620	5678	5696	
		5699	5717	5720	5731	5735	5855#								
LINP3	051515	5634	9678#												
LINP5	052035	5706	9678#												
LINP6	052232	5784	9678#												
LINSP	053246	3456	4379	4385	4391	4397	4611	4614	4725	4728	4731	4737	4740	4743	
		4775	5512	5545	5596	5601	5617	5718	5721	5732	6016	6021	6026	6032	
		6038	9678#	9871	9880										
LINSP0	053247	5455	5509	6044	6051	6056	6061	6066	9678#						
LINSS3	051701	5602	9678#												
LINST3	051664	5597	9678#												
LINS3	051536	5593	5650	9678#											
LINS4	051735	5679	9678#	9678#											
LINS5	052053	5715	9678#												
LINT3	051610	5664	9678#												
LINU06	052243	5767	9678#												
LINW3	051652	5584	9678#												
LINX4	051747	5682	9678#												
LINX5	053251	5722	9678#												
LIN10A	052662	4685	9678#												
LIN10B	052716	4690	9678#												
LIN10C	052757	4745	9678#												
LIN10H	053064	4722	9678#												
LIN11H	053151	4769	9678#												
LIN2C	051352	5475	9678#												
LIN2P	051373	5481	9678#												

[illegible]

[illegible]

[illegible]

SEQ 0200

[illegible]

[illegible]

SEQ 0202

[illegible]

[illegible]

[illegible]

\$CMTAG	001114	2523#	3280	3281	3295									
\$CM3	= 000000	2553#												
\$CNTLC	031170	7173	7232	7257#										
\$CNTLG	033623	7182	7739	7821#										
\$CNTLU	033616	7227	7756	7820#										
\$CODE	= 000024	4403	4452	4479	4530	4671	4765	4908	4910	5037	5041	5245	5288#	5296#
		5311	5362#	5382#	5386#	6452#	6461	8869	8871	8879#	9503#			
\$COMND	= 000002	4792#	4812#	4828#	4847#	5290#	5291	5295#	5298	5383#	5387#	6454#	6567#	8880#
		8881#	9489#											
\$CPUOP	001232	2576#												
\$CRLF	001201	2555#	3409	3432	3466	3470	3786	3794	3806	3812	3820	3840	3848	4375
		4382	4388	4394	4400	4597	4617	4629	4734	4746	4780	4782	5450	5451
		5452	5472	5505	5507	5532	5548	5567	5587	5606	5621	5645	5668	5687
		5708	5723	5736	5743	5776	5785	5809	5810	5835	5836	5966	5997	6144
		6355	6402	6408	6492	6532	6625	6635	6701	6714	6715	6724	6765	6781
		6789	6804	7172	7181	7237	7300	7315	7324	7343	7348	7352	7408	7443
		7767	9866											
\$CYL	= 000012	4463	4845#	4916	5043	5078	5278	5306#	5353#	5354	5361#	5594	6515#	6568#
		8877	9495#											
\$DBLK	033204	7674	7708	7716#										
\$DB20	034050	5794	5802	5807	5822	5827	6023	6029	6035	6041	7131	7923#		
\$DB20	034244	7148	7984#											
\$DECVL	034230	7925	7971#											
\$DEVCT	001214	2567#	6812#											
\$DEVM	001262	2603#												
\$DOAGN	027460	6820	6825	6831#										
\$DRVID	= 000224	6387	6539#	6540#	6541#	6542#	6543#	6544#	9546#	9550				
\$OSPLY	030124	6987#	8057											
\$DTBL	033174	7677	7712#											
\$ENDAO	027450	2488	6827#											
\$ENDCT	027434	6822#												
\$ENV	001224	2572#	3331	3339	3407	7302	7387	7456	7480					
\$ENVH	001225	2573#	3312	7389	7394	7458								
\$EOP	027112	6751#												
\$EOPCT	027426	6819#	6823											
\$ERFLG	001117	2526#	7287#	7315										
\$ERMAX	001131	2532#												
\$ERROR	031176	3287	7280#											
\$ERRPC	001132	2533#	7294#	7295#	7296	7315	7330							
\$ERRTB	003542	3218#	7338											
\$ERRTY	031362	7299	7323#											

MACY11 30(1046) 22-NOV-77 10:06 PAGE 208
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0206

[illegible]

CZRM880 RM03/2 PERF EXER MACY11 30(1046) 22-NOV-77 10:06 PAGE 209
CZRM88.P11 21-NOV-77 15:34 CROSS REFERENCE TABLE -- USER SYMBOLS

CZRM880 RM03/2 PERF EXER
CZRM88.P11 21-NOV-77 15:34

MACY11 30(1046) 22-NOV-77 10:06 PAGE 209
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0207

\$PASSC=	000070	6017	6309*	6767	6772	6802*	9517#										
\$PASTM	001106	2512#															
\$PATTC=	000030	5050	5304*	5389*	6455*	6456*	9506#										
\$POSIT=	000042	4918*	4919*	5080*	5081*	5825	6760	6763	9509#								
\$POWER	032420	7537	7544#														
\$PREVA=	000032	5078	5278*	5279*	5282*	5283*	5284*	5349*	5350*	5351*	5635	5639	5643	5651			
		8877*	8878*	9507#													
\$PREVO=	000027	5274*	5275*	5293	5300	5347*	5482	8876*	9505#								
\$PSEL =	000003	9490#															
\$PWRAD	032406	7539#															
\$PWRDN	032246	3291	7506#	7534													
\$PWRMG	032402	7537#															
\$PWRUP	032320	7516	7522#														
\$QUES	001200	2554#	7240	7315	7443	7786											
\$RAND	033652	5090	5181	5261	5339	7838#											
\$RDCHR	033476	7799#	8053														
\$RDEEC=	*****	8055															
\$RDLIN	030700	7200#	8054														
\$RDOCT=	*****	8055															
\$RDSZ =	000001	7820#															
\$READ =	000052	4914*	4915*	5805	6754	6757	9511#										
\$REG =	000014	6443	9496#	9576	9577	9584	9585	9592	9593	9600	9601	9608	9609	9616			
		9617	9624	9625	9632	9633											
\$RESRE	034012	7897#	8056														
\$RETRY	015344	3937	4023	4091	4136	4176	4212	4240	4310	4340	4356	4867#	4885				
\$RMADR	001270	2610#	3346*	3347	3353	9664	9867	9885									
\$RMAS =	000252	9557#	9670														
\$RMBA =	000240	4439	4661	4692	4713	4753	4773	4902	5529	5582	5683	5694*	5695	9552#			
		9670															
\$RMCS1=	000234	3749	3751	4415	5274	5347	5476	8876	9550#	9551	9552	9553	9554	9555			
		9556	9557	9558	9559	9560	9561	9562	9563	9564	9565	9566	9567	9568			
		9569	9576	9584	9592	9600	9608	9616	9624	9632	9666						
\$RMCS2=	000244	3874	4033	4084	4128	4168	4274	4289	4306	4417	9554#	9666					
\$RMDA =	000242	5105	5352	5615	5666	5897	5898	9553#	9670								
\$RMD8 =	000256	5698	9559#	9670													
\$RMDC =	000270	3728	4156	4916	5106	5353	5619	5654	5899	9564#	9674						

CZRM880 RM03/2 PERF EXER
CZRM88.P11 21-NOV-77 15:34MACY11 30(1046) 22-NOV-77 10:06 PAGE 210
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0208

SSB20	030436	4688	5872	6018	6048	6053	6058	6063	6072	6146	6151	6156	7129#	
SSB20	030466	5856	7146#											
SSC =	000010	4466	4844#	5046	5226	5279	5305*	5352*	5360*	5604	6513*	6578*	6585*	6586
SSC =	000010	6615*	8878	9493#										
SSC =	000010	3263#	3286	3287	3289	3291	3293	3295	3328	6816	7281	7312	7314	7723
SSC =	000010	7826												
SSC =	000010	3378	9829#											
SSC =	000010	5833	6070	6110	6112*	9515#								
SSC =	000010	6068	6086	6088*	9513#									
SSC =	000010	4472	4477	4478	4665	4757	5310*	5313*	5381*	5385*	5703	6460*	6463*	9502#
SSC =	000010	3263#												
SSC =	000010	4689	5795	5803	5808	5823	5828	5873	6958#					
SSC =	000010	2486#	2491											
SSC =	000010	2128#	2139	2143	2144	2145	2146	2553	3277	3295	6748	6816	6826	6832
SSC =	000010	6834	7273	7274	7275	7276	7277	7290	7297	7309	7313	7315	7540	
SSC =	000010	2574#	3314											
SSC =	000010	3694	3747	3764	3770	3772	3774	3776	3778	3780	3871	3962	3965	3982
SSC =	000010	4047	4816	4852	4868	4874	4900	5523	5674	6582	9497#	9674		
SSC =	000010	8058#												
SSC =	000010	2565#	3277*	6834*										
SSC =	000010	5454	5995	6142#	6203	7286								
SSC =	000010	2546#	6204	6255	7160	7721	7732	7749	7803	7809				
SSC =	000010	3327	3489	7158#										
SSC =	000010	2545#	6256*	7161*	7175*	7185*	7187*	7721	7730	7746	7770*	7801	7807	
SSC =	000010	7158	7168#											
SSC =	000010	2128#	2139	3275	3277#									
SSC =	000010	7930	7931	7951#										
SSC =	000010	5797	6067	6134	6136*	6737	9512#							
SSC =	000010	2548#	7432*	7443										
SSC =	000010	2552#	7381	7443										
SSC =	000010	2547#	7430	7443										
SSC =	000010	4906#	4907*	4912*	4913*	5800	9510#							
SSC =	000010	3289	8021#											
SSC =	000010	8032#	8043											
SSC =	000010	4843*	5224	5357	5359*	5599	6514*	6569*	9494#					
SSC =	000010	8036#	8045#	8046#	8047#	8048#	8049#	8050	8051#	8052	8053#	8054#	8055#	8056#
SSC =	000010	8057#	8058#											
SSC =	000010	8026	8043#	8058										
SSC =	000010	2511#												
SSC =	000010	2525#	7239	7315										
SSC =	000010	7202	7203	7215	7238	7252	7256#							
SSC =	000010	8049												
SSC =	000010	7662#	8048											
SSC =	000010	6977	6990	7381#	7475	8036	8044							
SSC =	000010	7411	7418	7425	7430#	7431	7772							
SSC =	000010	7436	7438	7441#										
SSC =	000010	7602#	8045											
SSC =	000010	7601	7604#	8047										
SSC =	000010	7597#	8046											
SSC =	000010	2568#	2626											
SSC =	000010	2513#												
SSC =	000010	2575#												
SSC =	000010	2600#	3341	3343										
SSC =	000010	2601#												
SSC =	000010	4436	4461	4563	4754	4912	4938	4945	4947*	4948	4949*	4980	4989	4994
SSC =	000010	4998	5040	5307*	5370*	5371*	5377*	5525	5680	5701	6457*	9501#	9502	9503

```
SWRDM = 000004
SSGET4= 000000
$OFILL 032765
$4OCAT= ***** U
. = 060065
```

[illegible]

```
.SASTA= ***** U
.SX    = 001100
```

MACY11 30(1046) 22-NOV-77 10:06 PAGE 213
CROSS REFERENCE TABLE -- MACRO NAMES

A	6078#	6079	6091	6103	6115	6127										
CKCHR	2128#	7080	7094													
CKDIG	2128#	6666														
CKNUM	2128#															
COMMEN	2271#															
ENDCOM	2271#															
ERENTR	7267#	7282														
ERRCAL	8062#	8894	8945	8947	9210	9270										
ERROR	2165#	8894	8945	8948	9211	9270										
ESCAPE	2271#															
GETPRI	2271#	9835														
GETSWR	2128#	2271#	3328													
MORETA	2492#	2608														
MULT	2271#															
NEWTST	2271#	3275														
POP	2271#	4583	5234	5431	5535	6464	6639	7490	7491	7527	7528	7703	7857	7902		
PUSH	2271#	4574	5213	5405	5468	6431	6556	7451	7453	7474	7508	7514	7662	7838	7882	
REPORT	2271#															
RMO3.D	1#	8062														
SCOPE	2166#															
SETPRI	2271#															
SETTRA	8036#	8045	8046	8047	8048	8050	8052	8053	8054	8055	8056	8057				
SETUP	2271#	3279														
SKIP	2271#															
SLASH	2271#															
SPACE	2271#															
STARS	2271#	2274	2278	2320	2324	2442	2446	2484	2495	2497	2504	2518	2557	2560	2669	
	2673	2725	2729	2741	2745	2907	2911	3263	3270	3275	3534	3538	5436	5440	5877	
	5881	6746	7263	7269	7318	7366	7446	7504	7520	7574	7652	7720	7723	7791	7829	
	7866	7912	7975	8015	8060	8073	9480	9484	9644	9648	9779	9823				
SWRSU	2271#	3295#														
TRMTRP	8036#	8058														
TYPBIN	2271#															
TYPDEC	2271#															
TYPNAM	2271#															
TYPNUM	2271#															
TYPOCS	2271#	3434	3593	3789	3815	3843	6011	6390	6479	6694	6719	6776	6784			
TYPOCT	2271#	7330	7354	7741												
TYPTXT	2271#															
XXEP	6744#	6752														
\$\$CHRE	2516#															
\$\$CHTH	2516#															
\$\$ESCA	2271#		</													

CZRM880 RM03/2 PERF EXER
CZRM88.P11 21-NOV-77 15:34

MACY11 30(1046) 22-NOV-77 10:06 PAGE 214
CROSS REFERENCE TABLE --- MACRO NAMES

SEQ 0211

.\$CATC	2128#	2468
.\$CMTA	2128#	2516
.\$DB2D	2128#	7910
.\$DB2O	2128#	7973
.\$EOP	2128#	6744
.\$ERRO	2128#	7267
.\$ERRT	2128#	7316
.\$POWE	2128#	7502
.\$RAND	2128#	7827
.\$READ	2128#	7718
.\$SAVE	2128#	7864
.\$SIZE	2128#	9821
.\$TRAP	2128#	8013
.\$TYPD	2128#	7650
.\$TYPE	2128#	7364
.\$TYPO	2128#	7572

. ABS. 060065 000

ERRORS DETECTED: 0

RM03:CZRM88, RM03:CZRM88.SEQ/NL:MD:MC:CND:TOC/DOC/SOL/CRF=RM03:RMDRV5.P11, RM03:CZRM88.P11
RUN-TIME: 34 27 2 SECONDS
RUN-TIME RATIO: 2633/64=40.7
CORE USED: 42K (83 PAGES)

DOCUMENT PAGES: 211