

RM03,02

FUNCTIONAL TEST 3 CZRMEB0

AH-B004B-MC

JAN 1978

COPYRIGHT © 1977

digital

FICHE 1 OF 2

MADE IN USA

This microfiche card contains 144 frames of data, organized in a 12x12 grid. Each frame displays a small, high-contrast image of a document page. The pages appear to be technical or test-related documents, but the text within them is too small to be legible. The frames are evenly spaced and cover most of the card's surface.

RM03,02

FUNCTIONAL TEST 3
CZRMEB0

AH-B004B-MC
COPYRIGHT © 1977
FICHE 2 OF 2

JAN 1978
digital
MADE IN USA

The main table on the right side of the page is a large, multi-column data table. It contains approximately 10 columns of data, with the first few columns appearing to be headers or identifiers. The data is organized in a structured, tabular format, likely representing test results or system configurations. The text is too small to be legible, but the layout is consistent with a technical data table.

.REM \

IDENTIFICATION

PRODUCT CODE:	AC-8003B-MC
PRODUCT NAME:	CZRM0B0 RM03/RM02 FUNCTIONAL TEST, PART 3
DATE CREATED:	1 AUGUST 77
MAINTAINER:	DIAGNOSTIC GROUP
AUTHOR:	DOUG RIIKONEN

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

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

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.

COPYRIGHT (C) 1977, DIGITAL EQUIPMENT CORPORATION

Vertical text on the left margin, possibly a page number or document identifier.

CZRMBO PM03/2 FCTNL TST 3
CZRMBO.F11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 3

DO1

SEQ 0003

109

2. DUAL PORT CONFIGURATIONS

EO1

CZRMEBO RMO3/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 4

SEQ 0004

PAGE 2

110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125

3. MEMORY PARITY HARDWARE
 4. MEMORY MANAGEMENT HARDWARE
 5. ACT,APT COMPATIBILITY
 6. XXDP COMPATIBILITY
 7. OPERATING SYSTEM COMPATIBILITY
-
6. TEST DESCRIPTION

126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181

1.0 INTRODUCTION

1.1 ABSTRACT

THE RMO3 SUBSYSTEM FUNCTIONAL TEST IS A STAND ALONE PROGRAM WHICH USES FUNCTIONAL MEANS TO VERIFY THE OPERABILITY OF THE RMO3 DISK SUBSYSTEM. IN PARTICULAR, THE PROGRAM SERVES THE FOLLOWING PURPOSES:

TO EXPLICITLY ESTABLISH CONFIDENCE IN THE BASIC OPERATIONS OF THE DISK DRIVE, INCLUDING MECHANICAL POSITIONING AND DATA TRANSFER OPERATIONS;

TO IMPLICITLY ESTABLISH CONFIDENCE IN THE DRIVE/ADAPTER ELECTRICAL INTERFACE;

TO VERIFY THE FUNCTIONALITY OF THE RMO3 SUBSYSTEM, INCLUDING THE MASSBUS CONTROLLER, MASSBUS ADAPTER AND THE DISK DRIVE.

THE TEST IS COMPRISED OF 3 PARTS, WHICH WOULD NORMALLY BE RUN IN SEQUENCE, STARTING WITH PART 1. BRIEFLY, PART 1 TESTS HOUSEKEEPING AND MECHANICAL POSITIONING OPERATIONS; PART 2 TESTS WRITE, READ AND WRITE CHECK OPERATIONS USING HEADER AND DATA; PART 3 TESTS WRITE, READ AND WRITE CHECK OPERATIONS USING DATA.

1.2 UNIT UNDER TEST

THE UNIT UNDER TEST (UUT) IS THE RMO3 DISK SUBSYSTEM WHICH CONSISTS OF THE RHXX MASSBUS CONTROLLER, THE RMO3 MASSBUS ADAPTER AND THE STORAGE MODULE DISK DRIVE. NOTE THAT A DISK PACK IS REQUIRED FOR TESTING AND IS CONSIDERED AN INTEGRAL OF THE STORAGE MODULE DISK DRIVE.

2.0 OPERATING REQUIREMENTS

2.1 HARDWARE REQUIREMENTS

THE FOLLOWING MINIMUM HARDWARE CONFIGURATION, ASSUMED TO BE OPERATIONAL IS REQUIRED TO LOAD AND EXECUTE THE RMO3 SUBSYSTEM FUNCTIONAL TEST:

- PDP-11 PROCESSOR
- 16K MEMORY
- KW11-L OR KW11-P CLOCK
- PROGRAM LOADING DEVICE
- TERMINAL
- RH11 OR RH70 CONTROLLER
- UNIT UNDER TEST.

CZRMEBO RMO3/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 6

GO1

SEQ 0006

182
183

WHERE THE UNIT UNDER TEST CONSISTS OF ONE TO EIGHT RMO3 ADAPTERS, DISK
DRIVES AND DISK PACKS.

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
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239

2.2 MEDIA REQUIREMENTS

EACH UNIT BEING TESTED MUST BE LOADED WITH A SCRATCH DISK PACK BEFORE TESTING BEGINS ON THAT UNIT. THE PACK MAY BE FORMATTED OR UNFORMATTED, BUT MUST NOT CONTAIN NEEDED INFORMATION BECAUSE THE PACK WILL BE WRITTEN DURING THE TEST.

NOTE

WHEN THE PROGRAM IS LOADED VIA THE RMO3 DRIVE 0 IS NOT TESTED UNLESS THE OPERATOR SPECIFICALLY ENTERS THE DRIVE NUMBER DURING CONSOLE DIALOGUE.

2.3 PREREQUISITE DIAGNOSTIC PROGRAMS

RMO3 DISKLESS DIAGNOSTIC. CZRMJ-B

3.0 OPERATING PROCEDURE

3.1 LOADING

THE PROGRAM MAY BE LOADED BY EITHER OF THE FOLLOWING MEDIA:
. PAPER TAPE, USING THE STANDARD PAPER TAPE LOADING PROCEDURE;
.XXDP MEDIA, USING THE APPROPRIATE LOADING DEVICE.

NOTE

WHEN THE PROGRAM IS LOADED VIA THE RMO3 DRIVE 0 IS NOT TESTED UNLESS THE OPERATOR SPECIFICALLY ENTERS THE DRIVE NUMBER DURING CONSOLE DIALOGUE.

3.2 SWITCH OPTIONS

THE FOLLOWING SWITCH OPTIONS ARE PROVIDED TO ENHANCE THE UTILITY OF THE PROGRAM.

IO1

CZRMEBO RMO3/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 8

SEQ 0008

240
241

SW15 HALT ON ERROR
SW14 LOOP ON TEST (CURRENTLY BEING EXECUTED)

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
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297

SW13 INHIBIT ERROR TYPEOUTS
SW12 UNUSED
SW11 INHIBIT TEST ITERATIONS
SW10 BELL ON ERROR
SW09 LOOP ON ERROR
SW08 LOOP ON TEST IN SW07-00

THE LOW ORDER 8 SWITCHES (SW07-SW00) ARE USED IN CONJUNCTION WITH SW08 TO SPECIFY THE OCTAL NUMBER OF THE TEST WHICH THE PROGRAM WILL LOOP ON.

3.3 STARTING

THE PROGRAM STARTS AT LOCATION 200, WHICH PROVIDES WORST CASE TEST CONDITIONS IF RUNNING IN AN AUTOMATIC ENVIRONMENT. IF RUNNING IN A STAND-ALONE ENVIRONMENT, THE PROGRAM USES CONSOLE DIALOGUE TO ALLOW THE OPERATOR TO CONTROL TEST CONDITIONS.

3.4 HALTING

THE PROGRAM CAN BE HALTED BY TYPING CONTROL C FROM THE CONSOLE.

3.5 RESTARTING

THE PROGRAM CAN BE RESTARTED AT ADDRESS 200.

4.0 OPERATOR INTERFACE

4.1 PROGRAM ID

THE PROGRAM TYPES ITS TITLE AND MAINDEC NUMBER THE FIRST TIME IT IS STARTED AFTER BEING LOADED. PROGRAM IDENTIFICATION DOES NOT OCCUR IF THE PROGRAM IS RESTARTED.

4.2 CONSOLE DIALOGUE

WHEN THE PROGRAM IS RUNNING IN STAND ALONE MODE, IT ENTERS A CONSOLE DIALOGUE SEQUENCE AFTER TYPING THE PROGRAM I.D..

THE FIRST QUESTION TYPED OUT IS: "TYPE HELP TEXT (Y OR N)??". IF THE OPERATOR RESPONDS WITH A Y, THE PROGRAM WILL TYPE A BRIEF HELP MESSAGE WHICH WILL LIST SWITCH OPTIONS, ETC.

298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353

THE SECOND QUESTION TYPED OUT IS, "CHANGE RMO3 UNIBUS ADDRESS OR VECTOR ADDRESS (Y OR N)?" IF THE UNIBUS ADDRESS OF THE RMO3 IS NON STANDARD, THE OPERATOR SHOULD RESPOND Y, THEN ANSWER SUBSEQUENT QUESTIONS TO SPECIFY THE UNIBUS ADDRESS, VECTOR ADDRESS AND INTERRUPT PRIORITY. IF THE OPERATOR DOES NOT RESPOND WITH A Y, THE PROGRAM SKIPS TO THE THIRD QUESTION.

THE THIRD QUESTION TYPED OUT IS, "TYPE (A) TO TEST ALL DEVICES, OR TYPE DEVICE NUMBER(S). TERMINATE INPUT WITH CARRIAGE RETURN". IF THE OPERATOR TYPES A, ALL POSSIBLE DEVICES ARE TESTED. OTHERWISE, THE OEPRATOR CAN TYPE THE NUMBER(S) OF THE DEVICE(S) HE WANTS TESTED, AND TERMINATE HIS INPUT WITH A CARRIAGE RETURN.

IF THE PROGRAM IS RESTARTED, THE FIRST QUESTION TYPED IS, "USE SAME DEVICES (Y OR N)?" IF THE OPERATOR TYPES Y, THE TEST IS RESTARTED USING THE SAME DEVICES AS THE LAST TIME, OTHERWISE, THE TEST RESTARTS THE DIALOGUE AS IF THE PROGRAM WERE STARTED FOR THE FIRST TIME.

4.3 PROGRESS REPORTS

AN END OF PASS REPORT OCCURS EACH TIME THE PROGRAM IS EXECUTED FOR ALL DEVICES IN THE TEST QUE. THE END OF PASS REPORT INCLUDES A MESSAGE AND AN ERROR SUMMARY.

4.4 PERFORMANCE REPORT

NO PERFORMANCE REPORTS ARE GIVEN DURING THE EXECUTION OF THE PROGRAM.

4.5 PROGRAM HALTS

THERE ARE NO SCHEDULED HALTS DURING THE EXECUTION OF THE PROGRAM. PROCESSOR HALTS ARE DUE TO THE TRAP CATCHER.

4.6 ERROR REPORTS

THE FIRST LINE OF THE ERROR REPORT CONTAINS THE NUMBER OF THE UNIT BEING TESTED, THE TEST NUMBER, THE ERROR NUMBER AND THE VALUE OF THE PROGRAM COUNTER WHERE THE ERROR WAS CALLED. THIS LINE IS FOLLOWED BY THE ERROR MESSAGE: ONE OR MORE LINES OF TEXT WHICH GIVE A BRIEF, YET COMPREHENSIVE DESCRIPTION OF THE ERROR. THE ERROR MESSAGE IS NORMALLY FOLLOWED BY ONE OR MORE PAIRS OF LINES CONTAINING DATA HEADERS AND DATA PERTINENT TO THE ERROR, INCLUDING EXPECTED AND ACTUAL TEST RESULTS.

:

354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404

5.0 ENVIRONMENTAL SUPPORT

5.1 PROCESSOR COMPATIBILITY

THE RMO3 SUBSYSTEM FUNCTIONAL TEST IS EXECUTABLE ON ANY PDP-11 PROCESSOR, PROVIDING PREVIOUSLY MENTIONED HARDWARE REQUIREMENTS ARE MET, AND PROVIDING THAT DATA THROUGHPUT ON THE SYSTEM IS SUFFICIENT TO SUSTAIN DATA TRANSFER OPERATIONS.

5.2 DUAL PORT CONFIGURATIONS

THE RMO3 SUBSYSTEM FUNCTIONAL TEST DOES NOT SPECIFICALLY TEST DUAL PORT LOGIC IN THE RMO3 ADAPTER BUT IS EXECUTABLE ON RMO3 SUBSYSTEMS HAVING THE DUAL PORT OPTION PROVIDING THE DUAL PORT SWITCH IS SET TO THE APPROPRIATE PORT (A OR B).

5.3 MEMORY PARITY HARDWARE

MEMORY PARITY HARDWARE IS NOT USED DURING THE EXECUTION OF THE RMO3 SUBSYSTEM FUNCTIONAL TEST.

5.4 MEMORY MANAGEMENT HARDWARE

MEMORY MANAGEMENT HARDWARE IS NOT USED DURING THE RMO3 SUBSYSTEM FUNCTIONAL TEST. CAPABILITIES OF THE MASSBUS CONTROLLER.

5.5 ACT11, APT11 COMPATIBILITY

THE RMO3 SUBSYSTEM FUNCTIONAL TEST IS COMPATIBLE WITH ACT11 AND APT11 IN BOTH DUMP AND AUTOMATIC MODES. FURTHER, THE PROGRAM WILL EXECUTE A QUICK PASS DURING THE FIRST PASS IN SUPPORT OF QUICK VERIFY MODE.

5.6 XXDP COMPATIBILITY

THE RMO3 SUBSYSTEM FUNCTIONAL TEST IS COMPATIBLE WITH XXDP IN DUMP AND CHAIN MODES, AND PROVIDES MEDIA PROTECTION IN THE CASE WHERE THE RMO3 IS THE XXDP LOADING DEVICE.

MO1

CZRMEBO RMO3/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 12

SEQ 0012

PAGE 8

405
406
407
408
409
410

5.7 OPERATING SYSTEM COMPATIBILITY

THE PROGRAM IS NOT COMPATIBLE WITH ANY SOFTWARE OPERATING SYSTEM.

CZMEBO RMO3:2 FCTNL TST 3
CZMEB.P11 23-NOV-77 12:23

467

MACY11 30(1046) 23-NOV-77 12:49 PAGE 14

802

SEQ 0014

NONEXISTENT DEVICE;

468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517

.DEVICE AVAILABLE - THE DEVICE IS AVAILABLE FOR TESTING.

THE PROGRAM JUMPS TO THE SUBPASS HANDLER WHICH SELECTS THE NEXT DRIVE TO BE TESTED IF THE DEVICE IS NOT AVAILABLE.

DRIVE TYPE TEST

PURPOSE:

TO VERIFY THAT THE UNIT UNDER TEST IS AN RMO3 SINGLE PORT OR DUAL PORT SUBSYSTEM.

PROCEDURE:

THIS TEST READS THE DRIVE TYPE REGISTER, RMDT, OF THE SELECTED DEVICE AND VERIFIES THAT THE DEVICE IS A SINGLE PORT OR DUAL PORT RMO3 SUBSYSTEM. IF THE SELECTED DEVICE IS NOT AN RMO3, THE PROGRAM JUMPS TO THE SUBPASS HANDLER WHICH WILL SELECT THE NEXT DEVICE FOR TESTING.

WRITE/READ DATA TESTS

PURPOSE:

TO TEST WRITE DATA AND READ DATA FUNCTIONALITY OF THE RMO3 SUBSYSTEM USING A SET OF VARIABLES WHICH INCLUDE WORD COUNT, HEAD MOTION, HEAD SWITCHING AND ERROR CONDITIONS.

PROCEDURE:

ALTHOUGH EACH TEST EXERCISES A DIFFERENT VARIABLE, THE GENERAL PROCEDURE OF EACH TEST IS THE SAME. THE DRIVE IS INITIALIZED AND RECALIBRATED IF "PIP" OR "SKI" ARE ACTIVE SO THAT THERE ARE NO ERRORS WHEN A TEST BEGINS. THEN, THE TEST FORMATS THE SECTOR BEING USED, WHICH MAY VARY FROM THE PROGRAM LISTING, BECAUSE SECTORS ARE SUBSTITUTED DURING RUN TIME IF THE SELECTED SECTOR IS LISTED IN THE BAD BLOCK TABLE OF THE LAST TRACK. FOLLOWING THAT, THE TEST PERFORMS ANY EXPLICIT SEEKS REQUIRED FOR THE CONDITIONS OF THE TEST. REGISTERS ARE PRESET AND THE WRITE

518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571

DATA COMMAND IS EXECUTED. WHEN THE WRITE COMMAND IS COMPLETE, THE TEST STORES ALL SUBSYSTEM STATUS AND CHECKS FOR PRIMARY ERRORS WHICH PRECLUDE OTHER STATUS CHECKS. IF THERE ARE NO PRIMARY ERRORS, THE TEST VERIFIES THE RESULTS OF THE WRITE COMMAND AND THEN CHECKS FOR SECONDARY ERRORS. LOOP ADDRESSES ARE MODIFIED FOLLOWING THE SUCCESSFUL COMPLETION OF THE WRITE COMMAND IN ORDER TO SHORTEN EXECUTION TIMES AND ENHANCE SCOPING LOOPS, THEN THE PROGRAM EXECUTES THE READ DATA PORTION OF THE TEST, VERIFYING THE SAME TYPE OF ERRORS AS IN THE WRITE COMMAND.

WRITE, READ ZEROS

THE TEST WRITES AND READS AN ALL ZEROS DATA FIELD, CAUSING THE DRIVE TO USE NORMAL WRITE GATE THROUGHOUT THE WRITE PROCESS.

WRITE, WRITE CHECK ZEROS

THE TEST WRITES AND WRITE CHECKS AN ALL ZEROS DATA FIELD, VERIFYING THAT THERE ARE NO ERRORS.

WRITE, WRITE CHECK ZEROS W' WCE ERROR

THE TEST WRITES AN ALL ZEROS DATA FIELD, THEN COMPLEMENTS THE LAST WORD OF THE WRITE BUFFER. A WRITE CHECK COMMAND IS EXECUTED AND THE TEST VERIFIES THAT THE CORRECT WRITE CHECK ERROR IS DETECTED.

WRITE, READ ONES

THE TEST WRITES AND READS AN ALL ONES DATA FIELD, CAUSING THE DRIVE TO USE NORMAL WRITE GATE THROUGHOUT THE WRITE PROCESS.

WRITE, WRITE CHECK ONES

THE TEST WRITES AND WRITE CHECKS AN ALL ONES DATA FIELD.

WRITE, WRITE CHECK ONES W/ WCE ERROR

THE TEST WRITES AN ALL ONES DATA FIELD, THEN PERFORMS A WRITE CHECK DATA COMMAND AFTER COMPLEMENTING THE LAST WORD OF THE WRITE BUFFER. THE TEST VERIFIES THAT THE CORRECT WRITE CHECK ERROR IS DETECTED.

WRITE, WRITE CHECK MULTIPLE SECTORS

THE TEST SEEKS TO CYLINDER 0 TO INSURE THERE IS NO HEAD MOTION DURING DATA TRANSFER. THE WRITE DATA COMMAND FOLLOWS, WITH THE WORD COUNT EQUAL TO MULTIPLE SECTORS. THE WORD COUNT DURING THE READ DATA PORTION OF THE TEST IS SET FOR ALL OF THE FIRST SECTOR AND THE 1 WORD OF THE SECOND SECTOR.

WRITE, WRITE CHECK WITH HEAD SWITCHING

THE TEST SEEKS TO CYLINDER 0 TO INSURE THERE IS NO HEAD MOTION DURING DATA TRANSFER. THE WRITE DATA COMMAND STARTS WITH CYLINDER 0, TRACK 0, SECTOR 31. THE WORD COUNT IS EQUAL TO MULTIPLE SECTORS WHICH CAUSES THE SUBSYSTEM TO SWITCH FROM TRACK 0 TO TRACK 1 AFTER THE FIRST SECTOR IS WRITTEN. THE READ DATA COMMAND USES THE SAME WORD COUNT AND STARTING SECTOR.

WRITE, READ WITH IMPLIED SEEK

THIS TEST SEEKS TO THE LAST CYLINDER PRIOR TO WRITING DATA ON CYLINDER 0, TRACK 1, SECTOR 1. THE EXPLICIT SEEK INSURES THAT THERE WILL BE MAXIMUM HEAD MOTION DURING THE IMPLIED SEEK OF THE WRITE COMMAND. THE SAME OPERATION, INCLUDING THE EXPLICIT SEEK

572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627

CZRMEBO RMO3/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

F02

MACY11 30(1046) 23-NOV-77 12:49 PAGE 18

SEQ 0018

628

IS REPEATED FOR READ DATA.

629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682

WRITE, WRITE CHECK WITH MIDTRANSFER SEEK

THIS TEST WRITES MULTIPLE SECTORS STARTING WITH CYLINDER 0, TRACK 4, SECTOR 31, CAUSING A MIDTRANSFER SEEK AFTER THE FIRST SECTOR IS WRITTEN. THE SAME SECTORS ARE READ WITH READ DATA COMMAND.

WRITE, READ W/ HCE ERROR

A SECTOR IS FORMATTED WITH AN INCORRECT HEADER, THEN THE TEST WRITES AND READS DATA FROM THE SECTOR AND VERIFIES THAT THE CORRECT ERROR IS DETECTED. EACH BIT POSITION OF BOTH HEADER WORDS IS TESTED IN THIS MANNER.

WRITE, READ W/ HCI

A SECTOR IS FORMATTED WITH AN INCORRECT HEADER, THEN WRITTEN AND READ WITH HEADER COMPARE INHIBITED. THE TEST VERIFIES THAT NO ERROR IS DETECTED.

WRITE, READ W/ IVC ERROR

VOLUME VALID IS RESET BY SETTING AND RESETTING DIAGNOSTIC MODE. THE TEST THEN EXECUTES A WRITE DATA COMMAND AND VERIFIES THAT INVALID COMMAND STATUS SETS. THE TEST IS REPEATED FOR READ DATA COMMAND.

WRITE, READ W/ BAI

THE TEST WRITES A SINGLE SECTOR WITH "BUS ADDRESS INCREMENT INHIBIT" SET AND VERIFIES THAT THE BUS ADDRESS REGISTER DOES NOT INCREMENT. THE SAME SECTOR IS READ WITH "BAI" RESET AND THE TEST CHECKS TO SEE THAT ALL THE DATA IS THE SAME.

683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
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

WRITE, READ EACH CURRENT LEVEL

THE TEST WRITES AND READS ON EACH OF THE FOLLOWING CYLINDERS IN ORDER TO WRITE AT EACH POSSIBLE CURRENT THRESHOLD.

CYLINDER	0	110 MA
"	128	104 MA
"	256	97 MA
"	384	91 MA
"	512	84 MA
"	640	77 MA
"	768	70 MA

WRITE, READ W/CURRENT LEVEL SWITCHING

THE TEST WRITES 2 SECTORS STARTING WITH THE LAST SECTOR OF CYLINDER 383. THE FIRST SECTOR IS WRITTEN AT ONE CURRENT LEVEL AND THE SECOND SECTOR IS WRITTEN AT ANOTHER CURRENT LEVEL. BOTH SECTORS ARE READ AND THE TEST VERIFIES THERE ARE NO ERRORS.

WRITE, READ EARLY PEAK SHIFT

THIS TEST WRITES AND READS A SINGLE SECTOR USING A DATA PATTERN DESIGNED TO INTRODUCE EARLY PEAK SHIFT.

WRITE, READ MIXED PEAK SHIFT

THIS TEST WRITES AND READS A SINGLE SECTOR USING A DATA PATTERN DESIGNED TO INTRODUCE MIXED PEAK SHIFT.

736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755

WRITE, READ EACH TRACK

THE TEST WRITES AND READS WITH EACH HEAD USING A MIX OF EARLY, NORMAL AND LATE WRITE GATES.

WRITE, READ W/ ABORT

THE TEST SETS AN ERROR IN THE ERROR REGISTER AND EXECUTES A WRITE DATA COMMAND VERIFYING THAT "PIP" REMAINS INACTIVE. THE SAME PROCEDURE IS USED FOR READ DATA COMMAND.

```

756 ;PROGRAM REVISION #001
757
758 .TITLE CZRMEBO RM03/2 FCTNL TST 3
759 ;*COPYRIGHT (C) 1977
760 ;*DIGITAL EQUIPMENT CORP.
761 ;*MAYNARD, MASS. 01754
762 ;*
763 ;*PROGRAM BY DOUG RIIKONEN
764 ;*
765 ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
766 ;*PACKAGE (MAINDEC-11-DZ04C-C3), JAN 19, 1977.
767 ;*
768 000001 $TN=1
769 .SBTTL OPERATIONAL SWITCH SETTINGS
770 ;*
771 ;* SWITCH USE
772 ;* -----
773 ;* 15 HALT ON ERROR
774 ;* 14 LOOP ON TEST
775 ;* 13 INHIBIT ERROR TYPEOUTS
776 ;* 12 ENABLE EXTENDED STATUS
777 ;* 11 INHIBIT ITERATIONS
778 ;* 10 BELL ON ERROR
779 ;* 9 LOOP ON ERROR
780 ;* 8 LOOP ON TEST IN SWR<7:0>
781 ;* 7 TN128
782 ;* 6 TN64
783 ;* 5 TN32
784 ;* 4 TN16
785 ;* 3 TN8
786 ;* 2 TN4
787 ;* 1 TN2
788 ;* 0 TN1
789 .SBTTL BASIC DEFINITIONS
790 ;*
791 001100 ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
792 STACK= 1100
793 .EQUIV EMT,ERROR ;:BASIC DEFINITION OF ERROR CALL
794 .EQUIV IOT,SCOPE ;:BASIC DEFINITION OF SCOPE CALL
795 ;*
796 ;*MISCELLANEOUS DEFINITIONS
797 000011 AT= 11 ;:CODE FOR HORIZONTAL TAB
798 000012 LF= 12 ;:CODE FOR LINE FEED
799 000015 CR= 15 ;:CODE FOR CARRIAGE RETURN
800 000200 CRLF= 200 ;:CODE FOR CARRIAGE RETURN-LINE FEED
801 177776 PS= 177776 ;:PROCESSOR STATUS WORD
802 .EQUIV PS,PSW
803 177774 STKLMT= 177774 ;:STACK LIMIT REGISTER
804 177772 PIRQ= 177772 ;:PROGRAM INTERRUPT REQUEST REGISTER
805 177570 DSWR= 177570 ;:HARDWARE SWITCH REGISTER
806 177570 DDISP= 177570 ;:HARDWARE DISPLAY REGISTER
807 ;*
808 ;*GENERAL PURPOSE REGISTER DEFINITIONS
809 000000 R0= %0 ;:GENERAL REGISTER
810 000001 R1= %1 ;:GENERAL REGISTER
811 000002 R2= %2 ;:GENERAL REGISTER

```

812	000003	R3=	%3	::	GENERAL REGISTER
813	000004	R4=	%4	::	GENERAL REGISTER
814	000005	R5=	%5	::	GENERAL REGISTER
815	000006	R6=	%6	::	GENERAL REGISTER
816	000007	R7=	%7	::	GENERAL REGISTER
817	000006	SP=	%6	::	STACK POINTER
818	000007	PC=	%7	::	PROGRAM COUNTER
819					
820		;*PRIORITY LEVEL DEFINITIONS			
821	000000	PR0=	0	::	PRIORITY LEVEL 0
822	000040	PR1=	40	::	PRIORITY LEVEL 1
823	000100	PR2=	100	::	PRIORITY LEVEL 2
824	000140	PR3=	140	::	PRIORITY LEVEL 3
825	000200	PR4=	200	::	PRIORITY LEVEL 4
826	000240	PR5=	240	::	PRIORITY LEVEL 5
827	000300	PR6=	300	::	PRIORITY LEVEL 6
828	000340	PR7=	340	::	PRIORITY LEVEL 7
829					
830		;* "SWITCH REGISTER" SWITCH DEFINITIONS			
831	100000	SW15=	100000		
832	040000	SW14=	40000		
833	020000	SW13=	20000		
834	010000	SW12=	10000		
835	004000	SW11=	4000		
836	002000	SW10=	2000		
837	001000	SW09=	1000		
838	000400	SW08=	400		
839	000200	SW07=	200		
840	000100	SW06=	100		
841	000040	SW05=	40		
842	000020	SW04=	20		
843	000010	SW03=	10		
844	000004	SW02=	4		
845	000002	SW01=	2		
846	000001	SW00=	1		
847		.EQUIV	SW09,SW9		
848		.EQUIV	SW08,SW8		
849		.EQUIV	SW07,SW7		
850		.EQUIV	SW06,SW6		
851		.EQUIV	SW05,SW5		
852		.EQUIV	SW04,SW4		
853		.EQUIV	SW03,SW3		
854		.EQUIV	SW02,SW2		
855		.EQUIV	SW01,SW1		
856		.EQUIV	SW00,SW0		
857					
858		;*DATA BIT DEFINITIONS (BIT00 TO BIT15)			
859	100000	BIT15=	100000		
860	040000	BIT14=	40000		
861	020000	BIT13=	20000		
862	010000	BIT12=	10000		
863	004000	BIT11=	4000		
864	002000	BIT10=	2000		
865	001000	BIT09=	1000		
866	000400	BIT08=	400		
867	000200	BIT07=	200		


```

868      000100      BIT06= 100
869      000040      BIT05= 40
870      000020      BIT04= 20
871      000010      BIT03= 10
872      000004      BIT02= 4
873      000002      BIT01= 2
874      000001      BIT00= 1
875      .EQUIV      BIT09, BIT9
876      .EQUIV      BIT08, BIT8
877      .EQUIV      BIT07, BIT7
878      .EQUIV      BIT06, BIT6
879      .EQUIV      BIT05, BIT5
880      .EQUIV      BIT04, BIT4
881      .EQUIV      BIT03, BIT3
882      .EQUIV      BIT02, BIT2
883      .EQUIV      BIT01, BIT1
884      .EQUIV      BIT00, BIT0
885
886      .*BASIC "CPU" TRAP VECTOR ADDRESSES
887      000004      ERRVEC= 4          ;; TIME OUT AND OTHER ERRORS
888      000010      RESVEC= 10         ;; RESERVED AND ILLEGAL INSTRUCTIONS
889      000014      TBITVEC= 14        ;; "T" BIT
890      000014      TRIVEC= 14         ;; TRACE TRAP
891      000014      BPTVEC= 14         ;; BREAKPOINT TRAP (BPT)
892      000020      IOTVEC= 20         ;; INPUT/OUTPUT TRAP (IOT) **SCOPE**
893      000024      PWRVEC= 24         ;; POWER FAIL
894      000030      EMTVEC= 30         ;; EMULATOR TRAP (EMT) **ERROR**
895      000034      TRAPVEC= 34        ;; "TRAP" TRAP
896      000060      TKVEC= 60          ;; TTY KEYBOARD VECTOR
897      000064      TPVEC= 64         ;; TTY PRINTER VECTOR
898      000240      PIRQVEC= 240       ;; PROGRAM INTERRUPT REQUEST VECTOR
899
900      .SBTTL      RMO3 REGISTER BIT DEFINITIONS
901
902      ;RMCS1      CONTROL STATUS REGISTER
903
904      004000      DVA      =          BIT11      ;DEVICE AVAILABLE-READ ONLY
905      000040      F4      =          BIT05      ;FUNCTION CODE
906      000020      F3      =          BIT04      ;FUNCTION CODE
907      000010      F2      =          BIT03      ;FUNCTION CODE
908      000004      F1      =          BIT02      ;FUNCTION CODE
909      000002      F0      =          BIT01      ;FUNCTION CODE
910      000001      GO      =          BIT00      ;GO BIT
911      000077      FNCMSK  =          000077    ;FUNCTION CODE MASK
912
913      ;FUNCTION CODES (BITS 01-05 OF RMCS1)
914
915      000000      NOP      =          000000    ;NOP COMMAND
916      006302      ILFO2   =          000002    ;ILLEGAL COMMAND
917      000004      SEEK    =          000004    ;SEEK COMMAND
918      000006      RECAL   =          000006    ;RECALIBRATE COMMAND
919      000010      DRVCLR  =          000010    ;DRIVE CLEAR COMMAND
920      000012      RLEASE  =          000012    ;RELEASE COMMAND
921      000014      OFFSET  =          000014    ;OFFSET COMMAND
922      000016      RTC     =          000016    ;RETURN TO CENTERLINE COMMAND
923      000020      RIP     =          000020    ;READ IN PRESET COMMAND

```

924	000022	PAKACK =	000022	;PACK ACKNOWLEDGE COMMAND
925	000022	PACACK =	PAKACK	
926	000024	ILF24 ==	000024	;ILLEGAL COMMAND
927	000026	ILF26 ==	000026	;ILLEGAL COMMAND
928	000030	SEARCH ==	000030	;SEARCH COMMAND
929	000030	ILF30 ==	000030	;ILLEGAL COMMAND
930	000032	ILF32 ==	000032	;ILLEGAL COMMAND
931	000034	ILF34 ==	000034	;ILLEGAL COMMAND
932	000036	ILF36 ==	000036	;ILLEGAL COMMAND
933	000040	ILF40 ==	000040	;ILLEGAL COMMAND
934	000042	ILF42 ==	000042	;ILLEGAL COMMAND
935	000044	ILF44 ==	000044	;ILLEGAL COMMAND
936	000046	ILF46 ==	000046	;ILLEGAL COMMAND
937	000050	WCD ==	000050	;WRITE CHECK DATA COMMAND
938	000052	WCH ==	000052	;WRITE CHECK HEADER AND DATA
939	000054	ILF54 ==	000054	;ILLEGAL COMMAND
940	000056	ILF56 ==	000056	;ILLEGAL COMMAND
941	000060	WD ==	000060	;WRITE DATA COMMAND
942	000062	WH ==	000062	;WRITE HEADER AND DATA COMMAND
943	000064	ILF64 ==	000064	;ILLEGAL COMMAND
944	000066	ILF66 ==	000066	;ILLEGAL COMMAND
945	000070	RD ==	000070	;READ DATA COMMAND
946	000072	RH ==	000072	;READ HEADER AND DATA COMMAND
947	000074	ILF74 ==	000074	;ILLEGAL COMMAND
948	000076	ILF76 =	000076	;ILLEGAL COMMAND
949				
950		;RMDA	DISK ADDRESS REGISTER	
951				
952	002000	TA4 =	BIT10	;TRACK ADDRESS 4
953	001000	TA2 =	BIT09	;TRACK ADDRESS 2
954	000400	TA1 =	BIT08	;TRACK ADDRESS 1
955	000020	SA16 =	BIT04	;SECTOR ADDRESS 16
956	000010	SA8 =	BIT03	;SECTOR ADDRESS 8
957	000004	SA4 =	BIT02	;SECTOR ADDRESS 4
958	000002	SA2 =	BIT01	;SECTOR ADDRESS 2
959	000001	SA1 =	BIT00	;SECTOR ADDRESS 1
960				
961		;TRACK,SECTOR MASKS		
962				
963	003400	TADMSK =	003400	;TRACK ADDRESS MASK
964	000037	SADMSK =	000037	;SECTOR ADDRESS MASK
965				
966		;RMD5	DRIVE STATUS REGISTER	
967				
968	100000	ATA =	BIT15	;ATTENTION ACTIVE
969	040000	ERR =	BIT14	;COMPOSITE ERROR
970	020000	PIP =	BIT13	;POSITIONING IN PROGRESS
971	010000	MOL =	BIT12	;MEDIUM ON LINE
972	004000	WRL =	BIT11	;WRITE LOCK
973	002000	LBT =	BIT10	;LAST BLOCK TRANSFERRED
974	001000	PGM =	BIT09	;PROGRAMMABLE
975	000400	DPR =	BIT08	;DRIVE PRESENT
976	000200	DRY =	BIT07	;DRIVE READY
977	000100	VV =	BIT06	;VOLUME VALID
978	000001	OM =	BIT00	;OFFSET MODE ACTIVE
979				

```

980          ;RMER1  ERROR REGISTER #1
981
982          100000      DCK      =      BIT15      ; DATA CHECK ERROR
983          040000      UNS      =      BIT14      ; DRIVE UNSAFE
984          020000      OPI      =      BIT13      ; OPERATION INCOMPLETE
985          010000      DTE      =      BIT12      ; DRIVE TIMING ERROR
986          004000      WLE      =      BIT11      ; WRITE LOCK ERROR
987          002000      IAE      =      BIT10      ; INVALID ADDRESS ERROR
988          001000      AOE      =      BIT09      ; ADDRESS OVERFLOW ERROR
989          000400      HCRC     =      BIT08      ; HEADER CRC ERROR
990          000200      HCE      =      BIT07      ; HEADER COMPARE ERROR
991          000100      ECH      =      BIT06      ; ECC "HARD" ERROR
992          000040      WCF      =      BIT05      ; WRITE CLOCK FAILURE
993          000020      FER      =      BIT04      ; FORMAT ERROR
994          000010      PAR      =      BIT03      ; PARITY ERROR
995          000004      RMR      =      BIT02      ; REGISTER MODIFICATION REFUSED
996          000002      ILR      =      BIT01      ; ILLEGAL REGISTER
997          000001      ILF      =      BIT00      ; ILLEGAL FUNCTION
998
999          115760      NDTMSK   =      DCK!DTE!WLE!AOE!HCRC!HCE!ECH!WCF!FER
1000          ; "NDTMSK" IS USED TO MASK ERROR REGISTER 1 DURING NON - DATA
1001          ; COMMANDS, I.E., HOUSEKEEPING AND POSITIONING COMMANDS
1002
1003          ;RMAS  ATTENTION SUMMARY REGISTER
1004
1005          000377      ATNMSK   =      377          ; MASK FOR ATTENTION BITS
1006
1007          ;RMLA  LOOK AHEAD REGISTER
1008
1009          002000      SC4      =      BIT10      ; SECTOR COUNT = 16
1010          001000      SC3      =      BIT09      ; SECTOR COUNT = 8
1011          000400      SC2      =      BIT08      ; SECTOR COUNT = 4
1012          000200      SC1      =      BIT07      ; SECTOR COUNT = 2
1013          000100      SC0      =      BIT06      ; SECTOR COUNT = 1
1014
1015          003700      SCTMSK   =      003700      ; SECTOR COUNT MASK
1016
1017          ;RMMR  MAINTENANCE REGISTER
1018
1019          ;      WRITE ONLY BITS
1020
1021          100000      DBCK     =      BIT15      ; DEBUG CLOCK
1022          040000      DBEN     =      BIT14      ; DEBUG CLOCK ENABLE
1023          020000      DEBL     =      BIT13      ; DIAGNOSTIC END OF BLOCK
1024          010000      DTO      =      BIT12      ; DIAGNOSTIC TIMEOUT
1025          004000      MCLK     =      BIT11      ; MAINTENANCE CLOCK
1026          002000      MRD      =      BIT10      ; READ DATA
1027          001000      MUR      =      BIT09      ; UNIT READY
1028          000400      MOC      =      BIT08      ; ON CYLINDER
1029          000200      MSER     =      BIT07      ; SEEK ERROR
1030          000100      MDF      =      BIT06      ; DRIVE FAULT
1031          000040      MS       =      BIT05      ; SECTOR PULSE
1032          000010      MWP      =      BIT03      ; WRITE PROTECT
1033          000004      MI       =      BIT02      ; INDEX PULSE
1034          000002      MSC      =      BIT01      ; SECTOR COMPARE
1035          000001      OMD      =      BIT00      ; DIAGNOSTIC MODE

```

```

1036
1037 ; READ ONLY BITS
1038
1039 100000 OCC = BIT15 ;OCCUPIED
1040 040000 RG = BIT14 ;RUN AND GO
1041 020000 EBL = BIT13 ;END OF BLOCK
1042 010000 REX = BIT12 ;EXCEPTION
1043 004000 ESRC = BIT11 ;ENABLE SEARCH
1044 002000 PLFS = BIT10 ;LOOKING FOR SYNC
1045 001000 ECRC = BIT09 ;ENABLE CRC OUT
1046 000400 PDA = BIT08 ;DATA AREA
1047 000200 PHA = BIT07 ;HEADER AREA
1048 000100 CONT = BIT06 ;CONTINUE
1049 000040 WC = BIT05 ;WORD CLOCK
1050 000020 EECC = BIT04 ;ENABLE ECC OUT
1051 000010 MWD = BIT03 ;WRITE DATA BIT
1052 000004 LS = BIT02 ;LAST SECTOR
1053 000002 LST = BIT01 ;LAST SECTOR AND TRACK
1054 000001 DMD = BIT00 ;DIAGNOSTIC MODE
1055
1056 ;RMDT DRIVE TYPE REGISTER
1057
1058 100000 NSA = BIT15 ;NOT SECTOR ADDRESSED=0
1059 040000 TAP = BIT14 ;TAPE DRIVE = 0
1060 020000 MOH = BIT13 ;MOVING HEAD = 1
1061 004000 DRQ = BIT11 ;DRIVE REQUEST REQUIRED
1062
1063 020024 SNGPRT = 020024 ;SINGLE PORT DRIVE TYPE
1064 024024 DULPRT = 024024 ;DUAL PORT DRIVE TYPE
1065
1066 ;RMOF OFFSET REGISTER
1067
1068 010000 FMT16 = BIT12 ;16 BIT WORD FORMAT
1069 004000 ECI = BIT11 ;ECC INHIBIT
1070 002000 HCI = BIT10 ;HEADER COMPARE INHIBIT
1071 000200 OFD = BIT07 ;OFFSET FORWARD
1072
1073
1074 ;RMDC DESIRED CYLINDER ADDRESS REGISTER
1075 001777 CYLMSK = 1777 ;MASK FOR CYLINDER ADDRESS
1076
1077 ;RMMR2 MAINTENANCE REGISTER #2
1078
1079 ; READ ONLY BITS
1080 100000 RQA = BIT15 ;PORT A REQUEST
1081 040000 RQB = BIT14 ;PORT B REQUEST
1082 020000 TAG = BIT13 ;TAG CONTROL
1083 010000 TST = BIT12 ;COMMAND SEQUENCE TEST BIT
1084 004000 CC = BIT11 ;CONTROL OR CYLINDER TAG
1085 002000 CH = BIT10 ;CONTROL OR HEAD TAG
1086 001000 BB09 = BIT09 ;TAG BUS
1087 000400 BB08 = BIT08 ;TAG BUS
1088 000200 BB07 = BIT07 ;TAG BUS
1089 000100 BB06 = BIT06 ;TAG BUS
1090 000040 BB05 = BIT05 ;TAG BUS
1091 000020 BB04 = BIT04 ;TAG BUS

```

1092	000010	8803	=	BIT03	: TAG BUS
1093	000004	8802	=	BIT02	: TAG BUS
1094	000002	8801	=	BIT01	: TAG BUS
1095	000001	8800	=	BIT00	: TAG BUS
1096					
1097					
1098					
1099					
1100	100000	BSE	=	BIT15	: BAD SECTOR ERROR
1101	040000	SKI	=	BIT14	: SEEK INCOMPLETE
1102	020000	OPE	=	BIT13	: OPERATOR PLUG ERROR
1103	010000	IVC	=	BIT12	: INVALID COMMAND ERROR
1104	004000	LSC	=	BIT11	: LOSS OF SYSTEM CLOCK
1105	002000	LBC	=	BIT10	: LOSS OF BIT CLOCK
1106	000200	DVC	=	BIT07	: DEVICE CHECK
1107	000010	DPE	=	BIT03	: DATA PARITY ERROR
1108					
1109					
1110		.SBTTL		PROGRAM MNEMONICS	
1111	100000	MSE	=	BIT15	: MANUFACTURING DETECTED SECTOR ERROR
1112	040000	USE	=	BIT14	: USER DETECTED SECTOR ERROR
1113					
1114		.SBTTL		RM03 REGISTER INDEX VALUES	
1115					
1116	000000	RMCS1	=	00	: CONTROL STATUS REGISTER
1117	000006	RMDA	=	06	: DISK ADDRESS REGISTER
1118	000012	RMDS	=	12	: DRIVE STATUS REGISTER
1119	000014	RMER1	=	14	: ERROR REGISTER 1
1120	000016	RMAS	=	16	: ATTENTION SUMMARY REGISTER
1121	000020	RMLA	=	20	: LOOK AHEAD REGISTER
1122	000024	RMMR1	=	24	: MAINTENANCE REGISTER
1123	000026	RMOT	=	26	: DRIVE TYPE REGISTER
1124	000030	RMSN	=	30	: SERIAL NUMBER REGISTER
1125	000032	RMOF	=	32	: OFFSET REGISTER
1126	000034	RMOC	=	34	: DESIRED CYLINDER REGISTER
1127	000036	RMCC	=	36	: CURRENT CYLINDER REGISTER
1128	000040	RMMR2	=	40	: MAINTENANCE REGISTER 2
1129	000042	RMER2	=	42	: ERROR REGISTER 2
1130	000044	RMEC1	=	44	: ECC POSITION REGISTER
1131	000046	RMEC2	=	46	: ECC PATTERN REGISTER
1132	000050	ILRG50	=	50	: ILLEGAL REGISTER 50
1133	000052	ILRG52	=	52	: ILLEGAL REGISTER 52
1134	000054	ILRG54	=	54	: ILLEGAL REGISTER 54
1135	000056	ILRG56	=	56	: ILLEGAL REGISTER 56
1136	000060	ILRG60	=	60	: ILLEGAL REGISTER 60
1137	000062	ILRG62	=	62	: ILLEGAL REGISTER 62
1138	000064	ILRG64	=	64	: ILLEGAL REGISTER 64
1139	000066	ILRG66	=	66	: ILLEGAL REGISTER 66
1140	000070	ILRG70	=	70	: ILLEGAL REGISTER 70
1141	000072	ILRG72	=	72	: ILLEGAL REGISTER 72
1142	000074	ILRG74	=	74	: ILLEGAL REGISTER 74
1143	000076	ILRG76	=	76	: ILLEGAL REGISTER 76
1144					
1145					
1146	000077	IDXMSK	=	77	: MASK FOR REGISTER INDEX NUMBER
1147					

```

1148 .SBTTL RH CONTROLLER REGISTER BIT DEFINITIONS
1149
1150 ;RMCS1 CONTROL STATUS REGISTER #1
1151
1152 100000 SC = BIT15 ;SPECIAL CONDITION-READ ONLY
1153 040000 TRE = BIT14 ;TRANSFER ERROR
1154 020000 MCPE = BIT13 ;MASSBUS CONTROL BUS PARITY
1155 ;ERROR-READ ONLY
1156 002000 FSEL = BIT10 ;PORT B SELECT
1157 001000 A17 = BIT09 ;ADDRESS EXTENSION
1158 000400 A16 = BIT08 ;ADDRESS EXTENSION
1159 000200 RDY = BIT07 ;READY-READ ONLY
1160 000100 IE = BIT06 ;INTERRUPT ENABLE
1161
1162 ;RMCS2 RH CONTROL STATUS REGISTER #2
1163
1164 100000 DLT = BIT15 ;DATA LATE-READ ONLY
1165 040000 WCE = BIT14 ;WRITE CHECK ERROR-READ ONLY
1166 020000 UPE = BIT13 ;UNIBUS PARITY ERROR
1167 010000 NED = BIT12 ;NONEXISTANT DRIVE-READ ONLY
1168 004000 NEM = BIT11 ;NONEXISTANT MEMORY-READ ONLY
1169 002000 PGE = BIT10 ;PROGRAM ERROR-READ ONLY
1170 001000 MXF = BIT09 ;MISSED TRANSFER
1171 000400 MDPE = BIT08 ;MASSBUS DATA BUS PARITY
1172 ;ERROR-READ ONLY
1173 000200 OR = BIT07 ;OUTPUT READY-READ ONLY
1174 000100 IR = BIT06 ;INPUT READY-READ ONLY
1175 000040 CLR = BIT05 ;CONTROLLER CLEAR
1176 000020 PAT = BIT04 ;PARITY TEST
1177 000010 BRI = BIT03 ;UNIBUS ADDRESS INCREMENT
1178 ;INHIBIT
1179 000004 U2 = BIT02 ;UNIT SELECT
1180 000002 U1 = BIT01 ;UNIT SELECT
1181 000001 U0 = BIT00 ;UNIT SELECT
1182
1183 ;UNIT SELECT MASK
1184 000007 UNTMSK = 7 ;UNIT SELECT MASK
1185
1186 ;RMCS3 RH70 CONTROL STATUS REGISTER #3
1187 100000 APE = BIT15 ;ADDRESS PARITY ERROR
1188 040000 DPEHI = BIT14 ;DATA PARITY ERROR HIGH WORD
1189 020000 DPELO = BIT13 ;DATA PARITY ERROR LOW WORD
1190 010000 WC_HI = BIT12 ;WRITE CHECK ERROR HIGH WORD
1191 004000 WCELO = BIT11 ;WRITE CHECK ERROR LOW WORD
1192 002000 DBL = BIT10 ;DOUBLE WORD TRANSFER
1193 000100 IE = BIT06 ;INTERRUPT ENABLE
1194 000010 IPCK3 = BIT03 ;INVERT PARITY CHECK
1195 000004 IPCK2 = BIT02 ;INVERT PARITY CHECK
1196 000002 IPCK1 = BIT01 ;INVERT PARITY CHECK
1197 000001 IPCK0 = BIT00 ;INVERT PARITY CHECK
1198
1199 .SBTTL RH CONTROLLER REGISTER INDEX VALUES
1200 000000 RMCS1 = 00 ;CONTROL STATUS REGISTER
1201 000002 RMWC = 02 ;WORD COUNT REGISTER
1202 000004 RMBA = 04 ;BUS ADDRESS REGISTER
1203 000010 RMCS2 = 10 ;CONTROLLER STATUS REGISTER

```

E03

CZRMEBO RM03/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 30
RM CONTROLLER REGISTER INDEX VALUES

SEQ 0030

```

1204          000022          RMOB      =      22          ;DATA BUFFER
1205          000050          RMBAE     =      50          ;BUS ADDRESS EXTENSION
1206          000052          RMCS3      =      52          ;CONTROL STATUS REGISTER #3
1207
1208          176700          ABASE     =      176700         ;UNIBUS ADDRESS
1209          120254          AVECT1     =      120254         ;UNIBUS VECTOR ADDRESS AND PRIORITY
1210
1211
1212          .SBTTL  TRAP CATCHER
1213
1214          000000          .=0
1215          ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
1216          ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
1217          ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
1218          000174          000174          .=174
1219 000174 000000          DISPREG: .WORD 0          ;;SOFTWARE DISPLAY REGISTER
1220 000176 000000          SWREG:  .WORD 0          ;;SOFTWARE SWITCH REGISTER
1221
1222          .SBTTL  ACT11 HOOKS
1223
1224          ;*****
1225          ;HOOKS REQUIRED BY ACT11
1226          $SVPC=.          ;SAVE PC
1227          .=46
1228 000046 037176          $ENDAD          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
1229          .=52
1230 000052 000000          .WORD 0          ;;2)SET LOC.52 TO ZERO
1231          000200          .=$SVPC          ;; RESTORE PC
1232
1233          .SBTTL  STARTING ADDRESS
1234
1235          ;THE PROGRAM STARTS AT LOCATION 200
1236          =      200
1237 000200 000137 005324          JMP      START          ;JUMP TO START OF PROGRAM
1238
1239          .=1100
1240          .SBTTL  APT PARAMETER BLOCK
1241
1242          ;*****
1243          ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
1244          ;*****
1245          .SX=.          ;;SAVE CURRENT LOCATION
1246          .=24          ;;SET POWER FAIL TO POINT TO START OF PROGRAM
1247 000024 000200          200          ;;FOR APT START UP
1248          .=44          ;;POINT TO APT INDIRECT ADDRESS PNTR.
1249 000044 001100          $APTHDR      ;;POINT TO APT HEADER BLOCK
1250          .=$X          ;;RESET LOCATION COUNTER
1251          ;*****
1252          ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
1253          ;INTERFACE SPEC.
1254
1255          $APTHD:
1256 001100 000000          $HIBTS: .WORD 0          ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
1257 001102 001222          $MADR:  .WORD $MAIL      ;;ADDRESS OF APT MAILBOX (BITS 0-15)
1258 001104 000001          $STM:   .WORD 1          ;;RUN TIM OF LONGEST TEST
1259

```


F03

CZRMEBO RM03/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 31
APT PARAMETER BLOCK

SEG 0031

1260 001106 000002
1261 001110 000002
1262 001112 000042
1263 001114 001114

\$PASTM: .WORD 2 ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
\$UNITM: .WORD 2 ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
TAGADR = . \$ETEND-\$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)

1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
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

001114
001114
001116
001117
001120
001122
001124
001126
001130
001131
001132
001134
001136
001140
001142
001144
001146
001150
001151
001152
001154
001156
001160
001162
001164
001166
001170
001171
001172
001173
001174
001176
001200
001202
001204
001206
001210
001212
001216
001217
001220

001222
001222
001224

000377

.SBTTL COMMON TAGS

; THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
; USED IN THE PROGRAM.

 .=TAGADR
\$CMTAG: .WORD 0 ; START OF COMMON TAGS
 .WORD 0
\$TSTNM: .BYTE 0 ; CONTAINS THE TEST NUMBER
\$ERFLG: .BYTE 0 ; CONTAINS ERROR FLAG
\$ICNT: .WORD 0 ; CONTAINS SUBTEST ITERATION COUNT
\$LPADR: .WORD 0 ; CONTAINS SCOPE LOOP ADDRESS
\$LPERR: .WORD 0 ; CONTAINS SCOPE RETURN FOR ERRORS
\$ERTTL: .WORD 0 ; CONTAINS TOTAL ERRORS DETECTED
\$ITEMB: .BYTE 0 ; CONTAINS ITEM CONTROL BYTE
\$ERMAX: .BYTE 1 ; CONTAINS MAX. ERRORS PER TEST
\$ERRPC: .WORD 0 ; CONTAINS PC OF LAST ERROR INSTRUCTION
\$GDADR: .WORD 0 ; CONTAINS ADDRESS OF 'GOOD' DATA
\$BDADR: .WORD 0 ; CONTAINS ADDRESS OF 'BAD' DATA
\$GDAT: .WORD 0 ; CONTAINS 'GOOD' DATA
\$BDAT: .WORD 0 ; CONTAINS 'BAD' DATA
 .WORD 0 ; RESERVED--NOT TO BE USED
 .WORD 0
\$AUTOB: .BYTE 0 ; AUTOMATIC MODE INDICATOR
\$INTAG: .BYTE 0 ; INTERRUPT MODE INDICATOR
 .WORD 0
\$SWR: .WORD USWR ; ADDRESS OF SWITCH REGISTER
\$DISPLAY: .WORD DDISP ; ADDRESS OF DISPLAY REGISTER
\$TKS: 177560 ; TTY KBD STATUS
\$TKB: 177562 ; TTY KBD BUFFER
\$TPS: 177564 ; TTY PRINTER STATUS REG. ADDRESS
\$TPB: 177566 ; TTY PRINTER BUFFER REG. ADDRESS
\$NULL: .BYTE 0 ; CONTAINS NULL CHARACTER FOR FILLS
\$FILLS: .BYTE 2 ; CONTAINS # OF FILLER CHARACTERS REQUIRED
\$FILLC: .BYTE 12 ; INSERT FILL CHARS. AFTER A "LINE FEED"
\$TPFLG: .BYTE 0 ; "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
\$TMP0: .WORD 0 ; USER DEFINED
\$TMP1: .WORD 0 ; USER DEFINED
\$TMP2: .WORD 0 ; USER DEFINED
\$TMP3: .WORD 0 ; USER DEFINED
\$TMP4: .WORD 0 ; USER DEFINED
\$TIMES: 0 ; MAX. NUMBER OF ITERATIONS
\$ESCAPE: 0 ; ESCAPE ON ERROR ADDRESS
\$BELL: .ASCIZ <207><377><377> ; CODE FOR BELL
\$QUES: .ASCII /?/ ; QUESTION MARK
\$CRLF: .ASCII <15> ; CARRIAGE RETURN
\$LF: .ASCIZ <12> ; LINE FEED

.SBTTL APT MAILBOX-ETABLE

.EVEN
\$MAIL: ; APT MAILBOX
\$MSGTY: .WORD AMSGTY ; MESSAGE TYPE CODE
\$FATAL: .WORD AFATAL ; FATAL ERROR NUMBER

1320	001226	000000	\$TESTN:	.WORD	ATESTN	::	TEST NUMBER
1321	001230	000000	\$PASS:	.WORD	APASS	::	PASS COUNT
1322	001232	000000	\$DEVCT:	.WORD	ADEVCT	::	DEVICE COUNT
1323	001234	000000	\$UNIT:	.WORD	AUNIT	::	I/O UNIT NUMBER
1324	001236	000000	\$MSGAD:	.WORD	AMSGAD	::	MESSAGE ADDRESS
1325	001240	000000	\$MSGLG:	.WORD	AMSGLG	::	MESSAGE LENGTH
1326	001242		\$ETABLE:			::	APT ENVIRONMENT TABLE
1327	001242	000	\$ENV:	.BYTE	AENV	::	ENVIRONMENT BYTE
1328	001243	000	\$ENVM:	.BYTE	AENVM	::	ENVIRONMENT MODE BITS
1329	001244	000000	\$SWREG:	.WORD	ASWREG	::	APT SWITCH REGISTER
1330	001246	000000	\$USWR:	.WORD	AUSWR	::	USER SWITCHES
1331	001250	000000	\$CPUOP:	.WORD	ACPUOP	::	CPU TYPE, OPTIONS
1332			*			::	BITS 15-11=CPU TYPE
1333			*			::	11/04=01, 11/05=02, 11/20=03, 11/40=04, 11/45=05
1334			*			::	11/70=06, PDQ=07, Q=10
1335			*			::	BIT 10=REAL TIME CLOCK
1336			*			::	BIT 9=FLOATING POINT PROCESSOR
1337			*			::	BIT 8=MEMORY MANAGEMENT
1338	001252	000	\$MAMS1:	.BYTE	AMAMS1	::	HIGH ADDRESS, M.S. BYTE
1339	001253	000	\$MTYP1:	.BYTE	AMTYP1	::	MEM. TYPE, BLK#1
1340			*			::	MEM. TYPE BYTE -- (HIGH BYTE)
1341			*			::	900 NSEC CORE=001
1342			*			::	300 NSEC BIPOLAR=002
1343			*			::	500 NSEC MOS=003
1344	001254	000000	\$MADR1:	.WORD	AMADR1	::	HIGH ADDRESS, BLK#1
1345			*			::	MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
1346	001256	000	\$MAMS2:	.BYTE	AMAMS2	::	HIGH ADDRESS, M.S. BYTE
1347	001257	000	\$MTYP2:	.BYTE	AMTYP2	::	MEM. TYPE, BLK#2
1348	001260	000000	\$MADR2:	.WORD	AMADR2	::	MEM. LAST ADDRESS, BLK#2
1349	001262	000	\$MAMS3:	.BYTE	AMAMS3	::	HIGH ADDRESS, M.S. BYTE
1350	001263	000	\$MTYP3:	.BYTE	AMTYP3	::	MEM. TYPE, BLK#3
1351	001264	000000	\$MADR3:	.WORD	AMADR3	::	MEM. LAST ADDRESS, BLK#3
1352	001266	000	\$MAMS4:	.BYTE	AMAMS4	::	HIGH ADDRESS, M.S. BYTE
1353	001267	000	\$MTYP4:	.BYTE	AMTYP4	::	MEM. TYPE, BLK#4
1354	001270	000000	\$MADR4:	.WORD	AMADR4	::	MEM. LAST ADDRESS, BLK#4
1355	001272	120254	\$VECT1:	.WORD	AVECT1	::	INTERRUPT VECTOR#1, BUS PRIORITY#1
1356	001274	000000	\$VECT2:	.WORD	AVECT2	::	INTERRUPT VECTOR#2, BUS PRIORITY#2
1357	001276	176700	\$BASE:	.WORD	ABASE	::	BASE ADDRESS OF EQUIPMENT UNDER TEST
1358	001300	000000	\$DEVN:	.WORD	ADEVN	::	DEVICE MAP
1359	001302	000000	\$CDW1:	.WORD	ACDW1	::	CONTROLLER DESCRIPTION WORD#1
1360	001304	000000	\$CDW2:	.WORD	ACDW2	::	CONTROLLER DESCRIPTION WORD#2
1361	001306	000000	\$DDW0:	.WORD	ADDW0	::	DEVICE DESCRIPTOR WORD#0
1362	001310	000000	\$DDW1:	.WORD	ADDW1	::	DEVICE DESCRIPTOR WORD#1
1363	001312	000000	\$DDW2:	.WORD	ADDW2	::	DEVICE DESCRIPTOR WORD#2
1364	001314	000000	\$DDW3:	.WORD	ADDW3	::	DEVICE DESCRIPTOR WORD#3
1365	001316	000000	\$DDW4:	.WORD	ADDW4	::	DEVICE DESCRIPTOR WORD#4
1366	001320	000000	\$DDW5:	.WORD	ADDW5	::	DEVICE DESCRIPTOR WORD#5
1367	001322	000000	\$DDW6:	.WORD	ADDW6	::	DEVICE DESCRIPTOR WORD#6
1368	001324	000000	\$DDW7:	.WORD	ADDW7	::	DEVICE DESCRIPTOR WORD#7
1369	001326		\$ETEND:			::	
1370			.MEXIT			::	
1371	001326	000000	CTLFG:	.WORD	0	::	
1372						::	
1373						::	
1374						::	
1375						::	

: THE REGISTER INPUT BUFFER IS USED FOR
: STORING DRIVE STATUS

```

1376 001330          GETBUF:
1377
1378          .REGISTER INPLT BUFFER
1379 001330 000000  RMCS1I: .WORD          ; CONTROL STATUS REGISTER
1380 001332 000000  RMWCI: .WORD          ; WORD COUNT REGISTER
1381 001334 000000  RMBAI: .WORD          ; BUS ADDRESS REGISTER
1382 001336 000000  RMDAI: .WORD          ; DISK ADDRESS REGISTER
1383 001340 000000  RMCS2I: .WORD         ; CONTROLLER STATUS REGISTER
1384 001342 000000  RMDSI: .WORD          ; DRIVE STATUS REGISTER
1385 001344 000000  RMER1I: .WORD         ; ERROR REGISTER 1
1386 001346 000000  RMASI: .WORD          ; ATTENTION SUMMARY REGISTER
1387 001350 000000  RMLAI: .WORD          ; LOOK AHEAD REGISTER
1388 001352 000000  RMDBI: .WORD          ; DATA BUFFER
1389 001354 000000  RMMR1I: .WORD         ; MAINTENANCE REGISTER #1
1390 001356 000000  RMDTI: .WORD          ; DRIVE TYPE REGISTER
1391 001360 000000  RMSNI: .WORD          ; SERIAL NUMBER REGISTER
1392 001362 000000  RMOFI: .WORD          ; OFFSET REGISTER
1393 001364 000000  RMDCI: .WORD          ; DESIRED CYLINDER REGISTER
1394 001366 000000  RMCCI: .WORD          ; CURRENT CYLINDER REGISTER
1395 001370 000000  RMMR2I: .WORD         ; MAINTENANCE REGISTER #2
1396 001372 000000  RMER2I: .WORD         ; ERROR REGISTER 2
1397 001374 000000  RMEC1I: .WORD         ; ECC POSITION REGISTER
1398 001376 000000  RMEC2I: .WORD         ; ECC PATTERN REGISTER
1399
1400          ; THE REGISTER OUTPUT BUFFER IS USED FOR
1401          ; ASSEMBLING DATA GOING TO REGISTER
1402
1403 001400          PUTBUF:
1404
1405          .REGISTER OUTPUT BUFFER
1406 001400 000000  RMCS1O: .WORD         ; CONTROL STATUS REGISTER
1407 001402 000000  RMWCO: .WORD         ; WORD COUNT REGISTER
1408 001404 000000  RMBAO: .WORD         ; BUS ADDRESS REGISTER
1409 001406 000000  RMDAO: .WORD         ; DISK ADDRESS REGISTER
1410 001410 000000  RMCS2O: .WORD        ; CONTROLLER STATUS REGISTER
1411 001412 000000  RMDSO: .WORD         ; DRIVE STATUS REGISTER
1412 001414 000000  RMER1O: .WORD        ; ERROR REGISTER 1
1413 001416 000000  RMASO: .WORD         ; ATTENTION SUMMARY REGISTER
1414 001420 000000  RMLAO: .WORD         ; LOOK AHEAD REGISTER
1415 001422 000000  RMDBO: .WORD         ; DATA BUFFER
1416 001424 000000  RMMR1O: .WORD        ; MAINTENANCE REGISTER #1
1417 001426 000000  RMDTO: .WORD         ; DRIVE TYPE REGISTER
1418 001430 000000  RMSNO: .WORD         ; SERIAL NUMBER REGISTER
1419 001432 000000  RMOFO: .WORD         ; OFFSET REGISTER
1420 001434 000000  RMDCO: .WORD         ; DESIRED CYLINDER REGISTER
1421 001436 000000  RMCCO: .WORD         ; CURRENT CYLINDER REGISTER
1422 001440 000000  RMMR2O: .WORD        ; MAINTENANCE REGISTER #2
1423 001442 000000  RMER2O: .WORD        ; ERROR REGISTER 2
1424 001444 000000  RMEC1O: .WORD        ; ECC POSITION REGISTER
1425 001446 000000  RMEC2O: .WORD        ; ECC PATTERN REGISTER
1426
1427          ; EACH WORD OF THE TEST QUE CONTAINS THE DEVICE NUMBER IN
1428          ; THE LOW BYTE AND THE ATTENTION BIT IN THE HIGH BYTE. A ZERO
1429          ; WORD IS A BLANK AND REPRESENTS THE END OF THE QUE.
1430
1431 001450 000012  TSTQUE: .BLKW 10.    ; TEST QUE

```

1432			
1433			
1434			;MEDIA ENABLE IS SET IF THE BAD SECTOR FILES HAVE BEEN RECOVERED
1435	001474	000000	;FOR THE UNIT UNDER TEST, OTHERWISE IT IS ZERO.
1436			MEDENB: .WORD ;MEDIA ENABLE
1437			
1438			;LOCATIONS "ASNDC" AND "ASNDC" CONTAIN THE CYLINDER, TRACK AND SECTOR
1439	001476	000000	;ADDRESS ASSIGNED BY THE BAD SECTOR MODULE.
1440	001500	000000	ASNDC: .WORD ;ASSIGNED DESIRED CYLINDER
1441	001502	000000	ASNDA: .WORD ;ASSIGNED TRACK, AND SECTOR
1442	001504	000000	CLKADR: .WORD ;UNIBUS ADDRESS OF KW11 CLOCK
1443			CLKVCT: .WORD ;VECTOR ADDRESS OF KW11 CLOCK
1444			
1445			;THE GET INDEX TABLE CONTAINS A BYTE LIST OF REGISTERS WHICH
1446			;ARE READ BY THE GET SUBROUTINE. THE LIST IS TERMINATED BY
1447	001506	000027	;A NEGATIVE BYTE.
1448			GETINX: .BLKB 23. ;GET INDEX TABLE
1449			
1450			;THE PUT INDEX TABLE ICONTAINS A BYTE LIST OF REGISTERS WHICH
1451			;ARE WRITTEN BY THE PUT SUBROUTINE. THE LIST IS TERMINATED BY
1452	001535	000027	;A NEGATIVE BYTE.
1453			PUTINX: .BLKB 23. ;PUT INDEX TABLE
1454			
1455			;PUT TAGS HERE

ERROR POINTER TABLE

.SBTTL ERROR POINTER TABLE

;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
;*LOCATION \$ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
;*NOTE1: IF \$ITEMB IS 0 THE ONLY PERTINENT DATA IS (\$ERRPC).
;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

;* EM ;;POINTS TO THE ERROR MESSAGE
;* DH ;;POINTS TO THE DATA HEADER
;* DT ;;POINTS TO THE DATA
;* DF ;;POINTS TO THE DATA FORMAT

\$ERRTB:

1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472

001564

1473			;ERROR	1	WRONG UNIT SELECTED
1474	001564	071434		EMT1	
1475	001566	075526		EHT1	
1476	001570	075652		EDT1	
1477	001572	075742		EFT1	
1478					
1479					
1480			;ERROR	2	DEVICE WENT UNAVAILABLE
1481	001574	071440		EMT2	
1482	001576	075526		EHT1	
1483	001600	075652		EDT1	
1484	001602	075742		EFT1	
1485					
1486					
1487			;ERROR	3	DEVICE WENT NONEXISTENT
1488	001604	071446		EMT3	
1489	001606	075526		EHT1	
1490	001610	075652		EDT1	
1491	001612	075742		EFT1	
1492					
1493					
1494			;ERROR	4	CONTROLLER NOT READY
1495	001614	071454		EMT4	
1496	001616	075526		EHT1	
1497	001620	075652		EDT1	
1498	001622	075742		EFT1	
1499					
1500					
1501			;ERROR	5	DRIVE NOT READY AND GO NOT RESET
1502	001624	071462		EMT5	
1503	001626	075526		EHT1	
1504	001630	075652		EDT1	
1505	001632	075742		EFT1	
1506					
1507					
1508			;ERROR	6	UNEXPECTED VALUE FOR "ATA" STATUS
1509	001634	071470		EMT6	
1510	001636	075526		EHT1	
1511	001640	075652		EDT1	
1512	001642	075742		EFT1	
1513					
1514					
1515			;ERROR	7	BUS TIMEOUT TRYING TO READ OR WRITE REGISTER :
1516	001644	071476		EMT7	
1517	001646	000000		0	
1518	001650	000000		0	
1519	001652	000000		0	
1520					
1521					
1522			;ERROR	10	DRIVE, NOT READY BUT GO IS RESET
1523	001654	071504		EMT10	
1524	001656	075526		EHT1	
1525	001660	075652		EDT1	
1526	001662	075742		EFT1	
1527					
1528					

1529			;ERROR	11	GO NOT RESET BUT DRIVE IS READY
1530	001664	071510		EMT11	
1531	001666	075526		EHT1	
1532	001670	075652		EDT1	
1533	001672	075742		EFT1	
1534					
1535					
1536			;ERROR	12	INCORRECT JUNCTION CODE
1537	001674	071514		EMT12	
1538	001676	075526		EHT1	
1539	001700	075652		EDT1	
1540	001702	075742		EFT1	
1541					
1542					
1543			;ERROR	13	PARITY ERROR READING REMOTE REGISTERS
1544	001704	071522		EMT13	
1545	001706	075526		EHT1	
1546	001710	075652		EDT1	
1547	001712	075742		EFT1	
1548					
1549					
1550			;ERROR	14	TRANSFER ERROR IS INCORRECT
1551	001714	071534		EMT14	
1552	001716	075526		EHT1	
1553	001720	075652		EDT1	
1554	001722	075742		EFT1	
1555					
1556					
1557			;ERROR	15	INCORRECT WORD COUNT
1558	001724	071542		EMT15	
1559	001726	075526		EHT1	
1560	001730	075652		EDT1	
1561	001732	075742		EFT1	
1562					
1563					
1564			;ERROR	16	INCORRECT BUS ADDRESS
1565	001734	071550		EMT16	
1566	001736	075526		EHT1	
1567	001740	075652		EDT1	
1568	001742	075742		EFT1	
1569					
1570					
1571			;ERROR	17	INCORRECT LBT STATUS
1572	001744	071560		EMT17	
1573	001746	075526		EHT1	
1574	001750	075652		EDT1	
1575	001752	075742		EFT1	
1576					
1577					
1578			;ERROR	20	INCORRECT AOE
1579	001754	071570		EMT20	
1580	001756	075526		EHT1	
1581	001760	075652		EDT1	
1582	001762	075742		EFT1	
1583					
1584					

1585			;ERROR	21	INCORRECT DISK ADDRESS
1586	001764	071600		EMT21	
1587	001766	075526		EHT1	
1588	001770	075652		EDT1	
1589	001772	075742		EFT1	
1590					
1591					
1592			;ERROR	22	INCORRECT CYLINDER ADDRESS
1593	001774	071610		EMT22	
1594	001776	075526		EHT1	
1595	002000	075652		EDT1	
1596	002002	075742		EFT1	
1597					
1598					
1599			;ERROR	23	INCORRECT WLE STATUS
1600	002004	071620		EMT23	
1601	002006	075526		EHT1	
1602	002010	075652		EDT1	
1603	002012	075742		EFT1	
1604					
1605					
1606			;ERROR	24	INCORRECT UPE STATUS
1607	002014	071630		EMT24	
1608	002016	075526		EHT1	
1609	002020	075652		EDT1	
1610	002022	075742		EFT1	
1611					
1612					
1613			;ERROR	25	INCORRECT WCF STATUS
1614	002024	071640		EMT25	
1615	002026	075526		EHT1	
1616	002030	075652		EDT1	
1617	002032	075742		EFT1	
1618					
1619					
1620			;ERROR	26	INCORRECT WCE STATUS
1621	002034	071650		EMT26	
1622	002036	075526		EHT1	
1623	002040	075652		EDT1	
1624	002042	075742		EFT1	
1625					
1626					
1627			;ERROR	27	INCORRECT MDPE STATUS
1628	002044	071660		EMT27	
1629	002046	075526		EHT1	
1630	002050	075652		EDT1	
1631	002052	075742		EFT1	
1632					
1633					
1634			;ERROR	30	INCORRECT DCK STATUS
1635	002054	071670		EMT30	
1636	002056	075526		EHT1	
1637	002060	075652		EDT1	
1638	002062	075742		EFT1	
1639					
1640					

1641			;ERROR	31	INCORRECT ECH STATUS
1642	002064	071700		EMT31	
1643	002066	075526		EHT1	
1644	002070	075652		EDT1	
1645	002072	075742		EFT1	
1646					
1647					
1648			;ERROR	32	DLT SHOULD NOT BE SET
1649	002074	071710		EMT32	
1650	002076	075526		EHT1	
1651	002100	075652		EDT1	
1652	002102	075742		EFT1	
1653					
1654					
1655			;ERROR	33	MXF SHOULD NOT BE SET
1656	002104	071720		EMT33	
1657	002106	075526		EHT1	
1658	002110	075652		EDT1	
1659	002112	075742		EFT1	
1660					
1661					
1662			;ERROR	34	DTE SHOULD NOT BE SET
1663	002114	071730		EMT34	
1664	002116	075526		EHT1	
1665	002120	075652		EDT1	
1666	002122	075742		EFT1	
1667					
1668					
1669			;ERROR	35	INCORRECT HCRC STATUS
1670	002124	071740		EMT35	
1671	002126	075526		EHT1	
1672	002130	075652		EDT1	
1673	002132	075742		EFT1	
1674					
1675					
1676			;ERROR	36	INCORRECT HCE STATUS
1677	002134	071750		EMT36	
1678	002136	075526		EHT1	
1679	002140	075652		EDT1	
1680	002142	075742		EFT1	
1681					
1682					
1683			;ERROR	37	INCORRECT FER STATUS
1684	002144	071760		EMT37	
1685	002146	075526		EHT1	
1686	002150	075652		EDT1	
1687	002152	075742		EFT1	
1688					
1689					
1690			;ERROR	40	DPE SHOULD NOT BE SET (NOT A DATA COMMAND)
1691	002154	071770		EMT40	
1692	002156	075526		EHT1	
1693	002160	075652		EDT1	
1694	002162	075742		EFT1	
1695					
1696					

1697			;ERROR 41	LOST "MOL" DURING PACK ACKNOWLEDGE
1698	002164	071776	EMT41	
1699	002166	075526	EHT1	
1700	002170	075652	EDT1	
1701	002172	075742	EFT1	
1702				
1703				
1704			;ERROR 42	UNSAFE ERROR DURING PACK ACKNOWLEDGE
1705	002174	072006	EMT42	
1706	002176	075526	EHT1	
1707	002200	075652	EDT1	
1708	002202	075742	EFT1	
1709				
1710				
1711			;ERROR 43	"OPI" ERROR DURING PACK ACKNOWLEDGE
1712	002204	072020	EMT43	
1713	002206	075526	EHT1	
1714	002210	075652	EDT1	
1715	002212	075742	EFT1	
1716				
1717				
1718			;ERROR 44	"RMR" ERROR DURING PACK ACKNOWLEDGE
1719	002214	072030	EMT44	
1720	002216	075526	EHT1	
1721	002220	075652	EDT1	
1722	002222	075742	EFT1	
1723				
1724				
1725			;ERROR 45	"ILR" ERROR DURING PACK ACKNOWLEDGE
1726	002224	072040	EMT45	
1727	002226	075526	EHT1	
1728	002230	075652	EDT1	
1729	002232	075742	EFT1	
1730				
1731				
1732			;ERROR 46	"ILF" ERROR DURING PACK ACKNOWLEDGE
1733	002234	072050	EMT46	
1734	002236	075526	EHT1	
1735	002240	075652	EDT1	
1736	002242	075742	EFT1	
1737				
1738				
1739			;ERROR 47	COMPOSITE ERROR STATUS IS INCORRECT
1740	002244	072060	EMT47	
1741	002246	075526	EHT1	
1742	002250	075652	EDT1	
1743	002252	075742	EFT1	
1744				
1745				
1746			;ERROR 50	PARITY ERROR WRITING REMOTE REGISTERS
1747	002254	072066	EMT50	
1748	002256	075526	EHT1	
1749	002260	075652	EDT1	
1750	002262	075742	EFT1	
1751				
1752				

1753			;ERROR	51	INCORRECT IAE STATUS DURING SEEK COMMAND
1754	002264	072076		EMT51	
1755	002266	075526		EHT1	
1756	002270	075652		EDT1	
1757	002272	075742		EFT1	
1758					
1759					
1760			;ERROR	52	OPI ERROR DURING SEEK - MEDIUM IS NOT ON LINE
1761	002274	072110		EMT52	
1762	002276	075526		EHT1	
1763	002300	075652		EDT1	
1764	002302	075742		EFT1	
1765					
1766					
1767			;ERROR	53	OPI ERROR DURING SEEK - MEDIUM IS ON LINE, ASSUME ON CYLINDER LATCH DIDN'T RESET
1768			:		
1769	002304	072126		EMT53	
1770	002306	075526		EHT1	
1771	002310	075652		EDT1	
1772	002312	075742		EFT1	
1773					
1774					
1775			;ERROR	54	SEEK INCOMPLETE ERROR DURING SEEK COMMAND
1776	002314	072144		EMT54	
1777	002316	075526		EHT1	
1778	002320	075652		EDT1	
1779	002322	075742		EFT1	
1780					
1781					
1782			;ERROR	55	DEVICE CHECK DURING SEEK COMMAND
1783	002324	072154		EMT55	
1784	002326	075526		EHT1	
1785	002330	075652		EDT1	
1786	002332	075742		EFT1	
1787					
1788					
1789			;ERROR	56	PIP IS STILL SET AFTER SEEK - SKI IS RESET
1790	002334	072166		EMT56	
1791	002336	075526		EHT1	
1792	002340	075652		EDT1	
1793	002342	075742		EFT1	
1794					
1795					
1796			;ERROR	57	ATA DID NOT SET DURING SEEK COMMAND
1797	002344	072204		EMT57	
1798	002346	075526		EHT1	
1799	002350	075652		EDT1	
1800	002352	075742		EFT1	
1801					
1802					
1803			;ERROR	60	IVC ERROR DURING SEEK COMMAND - LOST VOLUME VALID
1804			:		
1805	002354	072214		EMT60	
1806	002356	075526		EHT1	
1807	002360	075652		EDT1	
1808	002362	075742		EFT1	

1809				
1810				
1811			;ERROR 61	ERRONEOUS IVC ERROR DURING SEEK COMMAND -
1812			:	VOLUME VALID IS STILL SET
1813	002364	072232		
1814	002366	075526	EMT61	
1815	002370	075652	EHT1	
1816	002372	075742	EDT1	
1817			EFT1	
1818				
1819			;ERROR 62	MOL IS ZERO, BUT OPI WAS NOT
1820			:	REPORTED DURING SEEK COMMAND
1821	002374	072252	EMT62	
1822	002376	075526	EHT1	
1823	002400	075652	EDT1	
1824	002402	075742	EFT1	
1825				
1826				
1827			;ERROR 63	UNUSED
1828	002404	000000	0	
1829	002406	000000	0	
1830	002410	000000	0	
1831	002412	000000	0	
1832				
1833				
1834			;ERROR 64	DRIVE DID NOT DETECT "IVC" ERROR DURING SEEK
1835	002414	072270	EMT64	
1836	002416	075526	EHT1	
1837	002420	075652	EDT1	
1838	002422	075742	EFT1	
1839				
1840				
1841			;ERROR 65	DRIVE EXECUTED A SEEK WITH ERROR SET
1842	002424	072310	EMT65	
1843	002426	075526	EHT1	
1844	002430	075652	EDT1	
1845	002432	075742	EFT1	
1846				
1847				
1848			;ERROR 66	UNEXPECTED ERROR SET IN RMER1
1849	002434	072330	EMT66	
1850	002436	075526	EHT1	
1851	002440	075652	EDT1	
1852	002442	075742	EFT1	
1853				
1854				
1855			;ERROR 67	UNEXPECTED ERROR SET IN RMER2
1856	002444	072342	EMT67	
1857	002446	075526	EHT1	
1858	002450	075652	EDT1	
1859	002452	075742	EFT1	
1860				
1861				
1862			;ERROR 70	ERRONEOUS "IAE" ERROR DURING RECALIBRATE
1863	002454	072354	EMT70	
1864	002456	075526	EHT1	

1865	002460	075652	EDT1	
1866	002462	075742	EFT1	
1867				
1868				
1869			;ERROR	71 "ILF" ERROR DURING RECALIBRATE
1870	002464	072364	EMT71	
1871	002466	075526	EHT1	
1872	002470	075652	EDT1	
1873	002472	075742	EFT1	
1874				
1875				
1876			;ERROR	72 "OPI" ERROR DURING RECALIBRATE DUE TO "MOL" = 0
1877	002474	072374	EMT72	
1878	002476	075526	EHT1	
1879	002500	075652	EDT1	
1880	002502	075742	EFT1	
1881				
1882				
1883			;ERROR	73 "OPI" ERROR DURING RECALIBRATE BECAUSE ON
1884			:	CYLINDER DIDNT DROP
1885	002504	072412	EMT73	
1886	002506	075526	EHT1	
1887	002510	075652	EDT1	
1888	002512	075742	EFT1	
1889				
1890				
1891			;ERROR	74 "IVC" ERROR DURING RECALIBRATE - "VV" = 0
1892	002514	072430	EMT74	
1893	002516	075526	EHT1	
1894	002520	075652	EDT1	
1895	002522	075742	EFT1	
1896				
1897				
1898			;ERROR	75 ERRONEOUS "IVC" ERROR DURING RECALIBRATE - "VV" = 1
1899	002524	072440	EMT75	
1900	002526	075526	EHT1	
1901	002530	075652	EDT1	
1902	002532	075742	EFT1	
1903				
1904				
1905			;ERROR	76 "SKI" ERROR DURING RECALIBRATE
1906	002534	072460	EMT76	
1907	002536	075526	EHT1	
1908	002540	075652	EDT1	
1909	002542	075742	EFT1	
1910				
1911				
1912			;ERROR	77 "DVC" OCCURRED DURING RECALIBRATE
1913	002544	072470	EMT77	
1914	002546	075526	EHT1	
1915	002550	075652	EDT1	
1916	002552	075742	EFT1	
1917				
1918				
1919			;ERROR	100 LOST "MOL" DURING RECALIBRATE - "OPI" = 0
1920	002554	072502	EMT100	

1921	002556	075526		EHT1	
1922	002560	075652		EDT1	
1923	002562	075742		EFT1	
1924					
1925					
1926			;ERROR	101	LOST "VV" DURING RECALIBRATE - "IVC" = 0
1927	002564	072520		EMT101	
1928	002566	075526		EHT1	
1929	002570	075652		EDT1	
1930	002572	075742		EFT1	
1931					
1932					
1933			;ERROR	102	"ATA" DID NOT SET DURING RECALIBRATE
1934	002574	072536		EMT102	
1935	002576	075526		EHT1	
1936	002600	075652		EDT1	
1937	002602	075742		EFT1	
1938					
1939					
1940			;ERROR	103	"OM" DID NOT RESET DURING RECALIBRATE
1941	002604	072546		EMT103	
1942	002606	075526		EHT1	
1943	002610	075652		EDT1	
1944	002612	075742		EFT1	
1945					
1946					
1947			;ERROR	104	"PIP" IS STIL SET AFTER RECALIBRATE
1948	002614	072560		EMT104	
1949	002616	075526		EHT1	
1950	002620	075652		EDT1	
1951	002622	075742		EFT1	
1952					
1953					
1954			;ERROR	105	UNEXPECTED "ILR" ERROR DURING RECALIBRATE
1955	002624	072576		EMT105	
1956	002626	075526		EHT1	
1957	002630	075652		EDT1	
1958	002632	075742		EFT1	
1959					
1960					
1961			;ERROR	106	UNEXPECTED "RMR" ERROR DURING RECALIBRATE
1962	002634	072606		EMT106	
1963	002636	075526		EHT1	
1964	002640	075652		EDT1	
1965	002642	075742		EFT1	
1966					
1967					
1968			;ERROR	107	"UNS" ERROR DURING RECALIBRATE - AC POWER IS LOW
1969	002644	072616		EMT107	
1970	002646	075526		EHT1	
1971	002650	075652		EDT1	
1972	002652	075742		EFT1	
1973					
1974					
1975			;ERROR	110	CANNOT ACCESS MASSBUS CONTROLLER VIA UNIBUS
1976	002654	072636		EMT110	

1977	002656	075550		EHT110	
1978	002660	075670		EDT110	
1979	002662	075760		EFT110	
1980					
1981					
1982			;ERROR	111	NONEXISTENT DEVICE
1983	002664	072650		EMT111	
1984	002666	075554		EHT111	
1985	002670	075672		EDT111	
1986	002672	075762		EFT111	
1987					
1988					
1989			;ERROR	112	DEVICE NOT AVAILABLE
1990	002674	072656		EMT112	
1991	002676	075554		EHT111	
1992	002700	075672		EDT111	
1993	002702	075762		EFT111	
1994					
1995					
1996			;ERROR	113	BUS TIMEOUT-NED STATUS FAILURE
1997	002704	072664		EMT113	
1998	002706	000000		0	
1999	002710	000000		0	
2000	002712	000000		0	
2001					
2002					
2003			;ERROR	114	DEVICE NOT AN RM03
2004	002714	072700		EMT114	
2005	002716	075560		EHT114	
2006	002720	075674		EDT114	
2007	002722	075764		EFT114	
2008					
2009					
2010			;ERROR	115	RMCS1 NOT INITIALIZED BY UNIBUS
2011	002724	072706		EMT115	
2012	002726	075526		EHT1	
2013	002730	075652		EDT1	
2014	002732	075742		EFT1	
2015					
2016					
2017			;ERROR	116	RMBA NOT INITIALIZED BY UNIBUS
2018	002734	072716		EMT116	
2019	002736	075526		EHT1	
2020	002740	075652		EDT1	
2021	002742	075742		EFT1	
2022					
2023					
2024			;ERROR	117	RMCS2 NOT INITIALIZED BY UNIBUS
2025	002744	072726		EMT117	
2026	002746	075526		EHT1	
2027	002750	075652		EDT1	
2028	002752	075742		EFT1	
2029					
2030					
2031			;ERROR	120	RMER1 NOT INITIALIZED BY UNIBUS
2032	002754	072736		EMT120	

2033	002756	075526	EHT1	
2034	002760	075652	EDT1	
2035	002762	075742	EFT1	
2036				
2037				
2038				;ERROR 121 RMAS NOT INITIALIZED BY UNIBUS
2039	002764	072746	EMT121	
2040	002766	075526	EHT1	
2041	002770	075652	EDT1	
2042	002772	075742	EFT1	
2043				
2044				
2045				;ERROR 122 RMMR1 NOT INITIALIZED BY UNIBUS
2046	002774	072756	EMT122	
2047	002776	075526	EHT1	
2048	003000	075652	EDT1	
2049	003002	075742	EFT1	
2050				
2051				
2052				;ERROR 123 RMD5 NOT INITIALIZED BY UNIBUS
2053	003004	072766	EMT123	
2054	003006	075526	EHT1	
2055	003010	075652	EDT1	
2056	003012	075742	EFT1	
2057				
2058				
2059				;ERROR 124 RMEC2 NOT INITIALIZED BY UNIBUS
2060	003014	072776	EMT124	
2061	003016	075526	EHT1	
2062	003020	075652	EDT1	
2063	003022	075742	EFT1	
2064				
2065				
2066				;ERROR 125 RMMR2 NOT INITIALIZED BY UNIBUS
2067	003024	073006	EMT125	
2068	003026	075526	EHT1	
2069	003030	075652	EDT1	
2070	003032	075742	EFT1	
2071				
2072				
2073				;ERROR 126 RMCS1 NOT CLEARED BY CONTROLLER CLEAR
2074	003034	073016	EMT126	
2075	003036	075526	EHT1	
2076	003040	075652	EDT1	
2077	003042	075742	EFT1	
2078				
2079				
2080				;ERROR 127 RMBA NOT CLEARED BY CONTROLLER CLEAR
2081	003044	073030	EMT127	
2082	003046	075526	EHT1	
2083	003050	075652	EDT1	
2084	003052	075742	EFT1	
2085				
2086				
2087				;ERROR 130 RMCS2 NOT CLEARED BY CONTROLLER CLEAR
2088	003054	073042	EMT130	

2089	003056	075526	EHT1	
2090	003060	075652	EDT1	
2091	003062	075742	EFT1	
2092				
2093				
2094			;ERROR	131 RMER1 NOT CLEARED BY CONTROLLER CLEAR
2095	003064	073054	EMT131	
2096	003066	075526	EHT1	
2097	003070	075652	EDT1	
2098	003072	075742	EFT1	
2099				
2100				
2101			;ERROR	132 RMAS NOT CLEARED BY CONTROLLER CLEAR
2102	003074	073066	EMT132	
2103	003076	075526	EHT1	
2104	003100	075652	EDT1	
2105	003102	075742	EFT1	
2106				
2107				
2108			;ERROR	133 RMMR1 NOT CLEARED BY CONTROLLER CLEAR
2109	003104	073100	EMT133	
2110	003106	075526	EHT1	
2111	003110	075652	EDT1	
2112	003112	075742	EFT1	
2113				
2114				
2115			;ERROR	134 RMDS NOT CLEARED BY CONTROLLER CLEAR
2116	003114	073112	EMT134	
2117	003116	075526	EHT1	
2118	003120	075652	EDT1	
2119	003122	075742	EFT1	
2120				
2121				
2122			;ERROR	135 RMEC2 NOT CLEARED BY CONTROLLER CLEAR
2123	003124	073124	EMT135	
2124	003126	075526	EHT1	
2125	003130	075652	EDT1	
2126	003132	075742	EFT1	
2127				
2128				
2129			;ERROR	136 RMMR2 NOT CLEARED BY CONTROLLER CLEAR
2130	003134	073136	EMT136	
2131	003136	075526	EHT1	
2132	003140	075652	EDT1	
2133	003142	075742	EFT1	
2134				
2135				
2136			;ERROR	137 RMCS1 NOT CLEARED BY ERROR CLEAR
2137	003144	073150	EMT137	
2138	003146	075526	EHT1	
2139	003150	075652	EDT1	
2140	003152	075742	EFT1	
2141				
2142				
2143			;ERROR	140 RMCS2 NOT CLEARED BY ERROR CLEAR
2144	003154	073160	EMT140	

2145	003156	075526	EHT1	
2146	003160	075652	EDT1	
2147	003162	075742	EFT1	
2148				
2149				
2150				;ERROR 141 RMCS1 NOT CLEARED BY DRIVE CLEAR
2151	003164	073170	EMT141	
2152	003166	075526	EHT1	
2153	003170	075652	EDT1	
2154	003172	075742	EFT1	
2155				
2156				
2157				;ERROR 142 RMDS NOT CLEARED BY DRIVE CLEAR
2158	003174	073200	EMT142	
2159	003176	075526	EHT1	
2160	003200	075652	EDT1	
2161	003202	075742	EFT1	
2162				
2163				
2164				;ERROR 143 RMER1 NOT CLEARED BY DRIVE CLEAR
2165	003204	073210	EMT143	
2166	003206	075526	EHT1	
2167	003210	075652	EDT1	
2168	003212	075742	EFT1	
2169				
2170				
2171				;ERROR 144 RMAS NOT CLEARED BY DRIVE CLEAR
2172	003214	073220	EMT144	
2173	003216	075526	EHT1	
2174	003220	075652	EDT1	
2175	003222	075742	EFT1	
2176				
2177				
2178				;ERROR 145 RMMR1 NOT CLEARED BY DRIVE CLEAR
2179	003224	073230	EMT145	
2180	003226	075526	EHT1	
2181	003230	075652	EDT1	
2182	003232	075742	EFT1	
2183				
2184				
2185				;ERROR 146 RMMR2 NOT CLEARED BY DRIVE CLEAR
2186	003234	073240	EMT146	
2187	003236	075526	EHT1	
2188	003240	075652	EDT1	
2189	003242	075742	EFT1	
2190				
2191				
2192				;ERROR 147 RMER2 NOT CLEARED BY DRIVE CLEAR
2193	003244	073250	EMT147	
2194	003246	075526	EHT1	
2195	003250	075652	EDT1	
2196	003252	075742	EFT1	
2197				
2198				
2199				;ERROR 150 RMEC2 NOT CLEARED BY DRIVE CLEAR
2200	003254	073260	EMT150	

2201	003256	075526	EHT1	
2202	003260	075652	EDT1	
2203	003262	075742	EFT1	
2204				
2205				
2206				;ERROR 151 MEDIUM NOT ON LINE
2207	003264	073270	EMT151	
2208	003266	075526	EHT1	
2209	003270	075652	EDT1	
2210	003272	075742	EFT1	
2211				
2212				
2213				;ERROR 152 DRIVE FAULT
2214	003274	073302	EMT152	
2215	003276	075526	EHT1	
2216	003300	075652	EDT1	
2217	003302	075742	EFT1	
2218				
2219				
2220				;ERROR 153 UNSAFE SHOULD BE SET BECAUSE DVC IS SET
2221	003304	073314	EMT153	
2222	003306	075526	EHT1	
2223	003310	075652	EDT1	
2224	003312	075742	EFT1	
2225				
2226				
2227				;ERROR 154 UNSAFE SHOULD NOT BE SET, AC IS LOW
2228	003314	073332	EMT154	
2229	003316	075526	EHT1	
2230	003320	075652	EDT1	
2231	003322	075742	EFT1	
2232				
2233				
2234				;ERROR 155 VOLUME VALID NOT SET BY PACK ACK
2235	003324	073350	EMT155	
2236	003326	075526	EHT1	
2237	003330	075652	EDT1	
2238	003332	075742	EFT1	
2239				
2240				
2241				;ERROR 156 OFFSET MODE NOT SET BY OFFSET COMMAND
2242	003334	073362	EMT156	
2243	003336	075526	EHT1	
2244	003340	075652	EDT1	
2245	003342	075742	EFT1	
2246				
2247				
2248				;ERROR 157 OFFSET MODE NOT RESET BY RTC COMMAND
2249	003344	073374	EMT157	
2250	003346	075526	EHT1	
2251	003350	075652	EDT1	
2252	003352	075742	EFT1	
2253				
2254				
2255				;ERROR 160 RMOF NOT RESET BY RIP COMMAND
2256	003354	073406	EMT160	

2257	003356	075526	EHT1	
2258	003360	075652	EDT1	
2259	003362	075742	EFT1	
2260				
2261				
2262				;ERROR 161 RMDA NOT RESET BY RIP COMMAND
2263	003364	073416	EMT161	
2264	003366	075526	EHT1	
2265	003370	075652	EDT1	
2266	003372	075742	EFT1	
2267				
2268				
2269				;ERROR 162 RMDC NOT RESET BY RIP COMMAND
2270	003374	073430	EMT162	
2271	003376	075526	EHT1	
2272	003400	075652	EDT1	
2273	003402	075742	EFT1	
2274				
2275				
2276				;ERROR 163 DATA WAS ECC CORRECTED BUT DOES NOT COMPARE WITH
2277				; WRITE BUFFER
2278	003404	075322	EMT336	
2279	003406	075610	EHT336	
2280	003410	075706	EDY336	
2281	003412	075776	EFT336	
2282				
2283				
2284				;ERROR 164 OPI SHOULD NOT BE SET
2285	003414	073452	EMT164	
2286	003416	075526	EHT1	
2287	003420	075652	EDT1	
2288	003422	075742	EFT1	
2289				
2290				
2291				;ERROR 165 IVC SHOULD NOT BE SET
2292	003424	073460	EMT165	
2293	003426	075526	EHT1	
2294	003430	075652	EDT1	
2295	003432	075742	EFT1	
2296				
2297				
2298				;ERROR 166 IAE SHOULD NOT BE SET
2299	003434	073466	EMT166	
2300	003436	075526	EHT1	
2301	003440	075652	EDT1	
2302	003442	075742	EFT1	
2303				
2304				
2305				;ERROR 167 NEM SHOULD NOT BE SET
2306	003444	073474	EMT167	
2307	003446	075526	EHT1	
2308	003450	075652	EDT1	
2309	003452	075742	EFT1	
2310				
2311				
2312				;ERROR 170 UNUSED

2313	003454	000000	0	
2314	003456	000000	0	
2315	003460	000000	0	
2316	003462	000000	0	
2317				
2318				
2319				;ERROR 171 "ATA" NOT SET DURING RETURN TO CENTERLINE
2320	003464	073512	EMT171	
2321	003466	075526	EHT1	
2322	003470	075652	EDT1	
2323	003472	075742	EFT1	
2324				
2325				
2326				;ERROR 172 "ATA" NOT SET BY OFFSET COMMAND
2327	003474	073522	EMT172	
2328	003476	075526	EHT1	
2329	003500	075652	EDT1	
2330	003502	075742	EFT1	
2331				
2332				
2333				;ERROR 173 RMR2 NOT INITIALIZED BY UNIBUS INIT
2334	003504	073532	EMT173	
2335	003506	075526	EHT1	
2336	003510	075652	EDT1	
2337	003512	075742	EFT1	
2338				
2339				
2340				;ERROR 174 RMR2 NOT INITIALIZED BY CONTROLLER CLEAR
2341	003514	073542	EMT174	
2342	003516	075526	EHT1	
2343	003520	075652	EDT1	
2344	003522	075742	EFT1	
2345				
2346				
2347				;ERROR 175 SELECTED DEVICE IS IN WRITE PROTECT
2348	003524	073554	EMT175	
2349	003526	075526	EHT1	
2350	003530	075652	EDT1	
2351	003532	075742	EFT1	
2352				
2353				
2354				;ERROR 176 CANNOT SET DIAGNOSTIC MODE
2355	003534	073562	EMT176	
2356	003536	075526	EHT1	
2357	003540	075652	EDT1	
2358	003542	075742	EFT1	
2359				
2360				
2361				;ERROR 177 INCORRECT "MOL" STATUS DURING DIAGNOSTIC MODE
2362	003544	073570	EMT177	
2363	003546	075526	EHT1	
2364	003550	075652	EDT1	
2365	003552	075742	EFT1	
2366				
2367				
2368				;ERROR 200 INCORRECT "PIP" STATUS DURING DIAGNOSTIC MODE

2369	003554	073602		EMT200	
2370	003556	075526		EHT1	
2371	003560	075652		EDT1	
2372	003562	075742		EFT1	
2373					
2374					
2375			;ERROR	201	INCORRECT "WRL" STATUS DURING DIAGNOSTIC MODE
2376	003564	073614		EMT201	
2377	003566	075526		EHT1	
2378	003570	075652		EDT1	
2379	003572	075742		EFT1	
2380					
2381					
2382			;ERROR	202	INCORRECT "SKI" STATUS DURING DIAGNOSTIC MODE
2383	003574	073626		EMT202	
2384	003576	075526		EHT1	
2385	003600	075652		EDT1	
2386	003602	075742		EFT1	
2387					
2388					
2389			;ERROR	203	INCORRECT "DVC" STATUS DURING DIAGNOSTIC MODE
2390	003604	073640		EMT203	
2391	003606	075526		EHT1	
2392	003610	075652		EDT1	
2393	003612	075742		EFT1	
2394					
2395					
2396			;ERROR	204	"VV" WAS NOT RESET BY MAINTENANCE UNIT READY
2397	003614	073652		EMT204	
2398	003616	075526		EHT1	
2399	003620	075652		EDT1	
2400	003622	075742		EFT1	
2401					
2402					
2403			;ERROR	205	SELECTED DEVICE HAS A PERSISTENT "SKI" ERROR
2404	003624	073670		EMT205	
2405	003626	075526		EHT1	
2406	003630	075652		EDT1	
2407	003632	075742		EFT1	
2408					
2409					
2410			;ERROR	206	"LBC" DID NOT SET DURING DIAGNOSTIC MODE
2411	003634	073700		EMT206	
2412	003636	075526		EHT1	
2413	003640	075652		EDT1	
2414	003642	075742		EFT1	
2415					
2416					
2417			;ERROR	207	UNEXPECTED LOSS OF "MOL" - MEDIUM IS OFF LINE
2418	003644	073710		EMT207	
2419	003646	075526		EHT1	
2420	003650	075652		EDT1	
2421	003652	075742		EFT1	
2422					
2423					
2424			;ERROR	210	UNEXPECTED LOSS OF VOLUME VALID - "VV" = 0

2425	003654	073722		EMT210	
2426	003656	075526		EHT1	
2427	003660	075652		EDT1	
2428	003662	075742		EFT1	
2429					
2430					
2431			;ERROR	211	UNEXPECTED MECHANICAL MOTION - "PIP" = 1
2432	003664	073730		EMT211	
2433	003666	075526		EHT1	
2434	003670	075652		EDT1	
2435	003672	075742		EFT1	
2436					
2437					
2438			;ERROR	212	UNEXPECTED DEVICE FAULT - "DVC" = 1
2439	003674	073744		EMT212	
2440	003676	075526		EHT1	
2441	003700	075652		EDT1	
2442	003702	075742		EFT1	
2443					
2444					
2445			;ERROR	213	UNEXPECTED SEEK INCOMPLETE ERROR - "SKI" = 1
2446	003704	073760		EMT213	
2447	003706	075526		EHT1	
2448	003710	075652		EDT1	
2449	003712	075742		EFT1	
2450					
2451					
2452			;ERROR	214	DRIVE EXECUTED A RECALIBRATE WITH ERROR SET
2453	003714	073770		EMT214	
2454	003716	075540		EHT2	
2455	003720	075662		EDT2	
2456	003722	075752		EFT2	
2457					
2458					
2459			;ERROR	215	DRIVE DID NOT DETECT "IVC" ERROR DURING RECALIBRATE
2460	003724	074010		EMT215	
2461	003726	075540		EHT2	
2462	003730	075662		EDT2	
2463	003732	075752		EFT2	
2464					
2465					
2466			;ERROR	216	INCORRECT "IVC" STATUS
2467	003734	074022		EMT216	
2468	003736	075526		EHT1	
2469	003740	075652		EDT1	
2470	003742	075742		EFT1	
2471					
2472					
2473			;ERROR	217	INCORRECT "IAE" STATUS
2474	003744	074032		EMT217	
2475	003746	075526		EHT1	
2476	003750	075652		EDT1	
2477	003752	075742		EFT1	
2478					
2479					
2480			;ERROR	220	INCORRECT "WLE" STATUS

25181	003754	074042		EMT220	
25182	003756	075526		EHT1	
25183	003760	075652		EDT1	
25184	003762	075742		EFT1	
25185					
25186					
25187			; ERROR	221	INCORRECT "OPI" STATUS
25188	003764	074052		EMT221	
25189	003766	075526		EHT1	
25190	003770	075652		EDT1	
25191	003772	075742		EFT1	
25192					
25193					
25194			; ERROR	222	RM03 DID NOT DETECT RMR ERROR
25195	003774	074062		EMT222	
25196	003776	075526		EHT1	
25197	004000	075652		EDT1	
25198	004002	075742		EFT1	
25199					
25200					
25201			; ERROR	223	RM03 DID NOT DETECT PARITY ERROR ON MASSBUS CONTROL BUS
25202	004004	074072		EMT223	
25203	004006	075564		EHT223	
25204	004010	075676		EDT223	
25205	004012	075766		EFT223	
25206					
25207					
25208			; ERROR	224	UNUSED
25209	004014	000000		0	
25210	004016	000000		0	
25211	004020	000000		0	
25212	004022	000000		0	
25213					
25214					
25215			; ERROR	225	UNUSED
25216	004024	000000		0	
25217	004026	000000		0	
25218	004030	000000		0	
25219	004032	000000		0	
25220					
25221					
25222			; ERROR	226	UNUSED
25223	004034	000000		0	
25224	004036	000000		0	
25225	004040	000000		0	
25226	004042	000000		0	
25227					
25228					
25229			; ERROR	227	UNUSED
25230	004044	000000		0	
25231	004046	000000		0	
25232	004050	000000		0	
25233	004052	000000		0	
25234					
25235					
25236			; ERROR	230	UNUSED

2537	004054	000000	0	
2538	004056	000000	0	
2539	004060	000000	0	
2540	004062	000000	0	
2541				
2542				
2543				
2544	004064	000000		; ERROR 231 UNUSED
2545	004066	000000	0	
2546	004070	000000	0	
2547	004072	000000	0	
2548				
2549				
2550				
2551	004074	000000		; ERROR 232 UNUSED
2552	004076	000000	0	
2553	004100	000000	0	
2554	004102	000000	0	
2555				
2556				
2557				
2558	004104	000000		; ERROR 233 UNUSED
2559	004106	000000	0	
2560	004110	000000	0	
2561	004112	000000	0	
2562				
2563				
2564				
2565	004114	000000		; ERROR 234 UNUSED
2566	004116	000000	0	
2567	004120	000000	0	
2568	004122	000000	0	
2569				
2570				
2571				
2572	004124	000000		; ERROR 235 UNUSED
2573	004126	000000	0	
2574	004130	000000	0	
2575	004132	000000	0	
2576				
2577				
2578				
2579	004134	000000		; ERROR 236 UNUSED
2580	004136	000000	0	
2581	004140	000000	0	
2582	004142	000000	0	
2583				
2584				
2585				
2586	004144	000000		; ERROR 237 UNUSED
2587	004146	000000	0	
2588	004150	000000	0	
2589	004152	000000	0	
2590				
2591				
2592				
				; ERROR 240 UNUSED

2593	004154	000000	0	
2594	004156	000000	0	
2595	004160	000000	0	
2596	004162	000000	0	
2597				
2598				
2599				
2600	004164	000000	; ERROR 241	UNUSED
2601	004166	000000	0	
2602	004170	000000	0	
2603	004172	000000	0	
2604				
2605				
2606			; ERROR 242	UNUSED
2607	004174	000000	0	
2608	004176	000000	0	
2609	004200	000000	0	
2610	004202	000000	0	
2611				
2612				
2613			; ERROR 243	UNUSED
2614	004204	000000	0	
2615	004206	000000	0	
2616	004210	000000	0	
2617	004212	000000	0	
2618				
2619				
2620			; ERROR 244	UNUSED
2621	004214	000000	0	
2622	004216	000000	0	
2623	004220	000000	0	
2624	004222	000000	0	
2625				
2626				
2627			; ERROR 245	UNUSED
2628	004224	000000	0	
2629	004226	000000	0	
2630	004230	000000	0	
2631	004232	000000	0	
2632				
2633				
2634			; ERROR 246	"ATA" NOT RESET BY GO WHEN "ERR" = 0
2635	004234	074146	EMT246	
2636	004236	075526	EHT1	
2637	004240	075652	EDT1	
2638	004242	075742	EFT1	
2639				
2640				
2641			; ERROR 247	"ATA" NOT RESET BY WRITING RMAS
2642	004244	074156	EMT247	
2643	004246	075526	EHT1	
2644	004250	075652	EDT1	
2645	004252	075742	EFT1	
2646				
2647				
2648			; ERROR 250	"ATA" WAS RESET BY GO WHEN "ERR" = 1

2649	004254	074170		EMT250	
2650	004256	075526		EHT1	
2651	004260	075652		EDT1	
2652	004262	075742		EFT1	
2653					
2654					
2655			;ERROR	251	PROGRAM INTERRUPT WAS NOT GENERATED
2656	004264	074204		EMT251	
2657	004266	075540		EHT2	
2658	004270	075662		EDT2	
2659	004272	075752		EFT2	
2660					
2661					
2662			;ERROR	252	PROGRAM INTERRUPT SHOULD NOT HAVE BEEN GENERATED
2663	004274	074212		EMT252	
2664	004276	075540		EHT2	
2665	004300	075662		EDT2	
2666	004302	075752		EFT2	
2667					
2668					
2669			;ERROR	253	OFFSET MODE WAS NOT RESET BY WRITING RMDC
2670	004304	074220		EMT253	
2671	004306	075526		EHT1	
2672	004310	075652		EDT1	
2673	004312	075742		EFT1	
2674					
2675					
2676			;ERROR	254	INCORRECT "ILF" STATUS
2677	004314	074236		EMT254	
2678	004316	075526		EHT1	
2679	004320	075652		EDT1	
2680	004322	075742		EFT1	
2681					
2682					
2683			;ERROR	255	INCORRECT "ATA" STATUS
2684	004324	074246		EMT255	
2685	004326	075526		EHT1	
2686	004330	075652		EDT1	
2687	004332	075742		EFT1	
2688					
2689					
2690			;ERROR	256	INCORRECT "ILP" STATUS
2691	004334	074256		EMT256	
2692	004336	075576		EHT256	
2693	004340	075676		EDT223	
2694	004342	075766		EFT223	
2695					
2696					
2697			;ERROR	257	INVALID IAE STATUS DURING SEARCH COMMAND
2698	004344	074266		EMT257	
2699	004346	075526		EHT1	
2700	004350	075652		EDT1	
2701	004352	075742		EFT1	
2702					
2703					
2704			;ERROR	260	"IVC" WAS NOT DETECTED DURING SEARCH COMMAND

2705	004354	074300	EMT260	
2706	004356	075526	EHT1	
2707	004360	075652	EDT1	
2708	004362	075742	EFT1	
2709				
2710				
2711			;ERROR 261	DRIVE EXECUTED SEARCH WITH ERROR SET
2712	004364	074312	EMT261	
2713	004366	075526	EHT1	
2714	004370	075652	EDT1	
2715	004372	075742	EFT1	
2716				
2717				
2718			;ERROR 262	"LBC" ERROR NOT SET DURING DIAGNOSTIC MODE
2719	004374	074332	EMT262	
2720	004376	075526	EHT1	
2721	004400	075652	EDT1	
2722	004402	075742	EFT1	
2723				
2724				
2725			;ERROR 263	"SKI" ERROR DURING SEARCH COMMAND
2726	004404	074342	EMT263	
2727	004406	075526	EHT1	
2728	004410	075652	EDT1	
2729	004412	075742	EFT1	
2730				
2731				
2732			;ERROR 264	"IVC" ERROR DURING SEARCH - LOST VOLUME VALID
2733	004414	074352	EMT264	
2734	004416	075526	EHT1	
2735	004420	075652	EDT1	
2736	004422	075742	EFT1	
2737				
2738				
2739			;ERROR 265	ERRONEOUS IVC ERROR DURING SEARCH - VOLUME IS VALID
2740	004424	074372	EMT265	
2741	004426	075526	EHT1	
2742	004430	075652	EDT1	
2743	004432	075742	EFT1	
2744				
2745				
2746			;ERROR 266	DEVICE FAULT (DVC) DURING SEARCH
2747	004434	074412	EMT266	
2748	004436	075526	EHT1	
2749	004440	075652	EDT1	
2750	004442	075742	EFT1	
2751				
2752				
2753			;ERROR 267	SKI SHOULD HAVE BEEN SET BECAUSE CYLINDER ADDRESS IS TOO LARGE
2754			:	
2755	004444	074424	EMT267	
2756	004446	075526	EHT1	
2757	004450	075652	EDT1	
2758	004452	075742	EFT1	
2759				
2760				

2761			;ERROR	270	OPI ERROR DURING SEARCH BECAUSE MOL = 0
2762	004454	074442		EMT270	
2763	004456	075526		EHT1	
2764	004460	075652		EDT1	
2765	004462	075742		EFT1	
2766					
2767					
2768			;ERROR	271	OPI ERROR DURING SEARCH BECAUSE ON CYLINDER
2769			:		DIDN'T DROP
2770	004464	074456		EMT271	
2771	004466	075526		EHT1	
2772	004470	075652		EDT1	
2773	004472	075742		EFT1	
2774					
2775					
2776			;ERROR	272	LOST MOL DURING SEARCH, OPI IS NOT SET
2777	004474	074474		EMT272	
2778	004476	075526		EHT1	
2779	004500	075652		EDT1	
2780	004502	075742		EFT1	
2781					
2782					
2783			;ERROR	273	PIP STIL SET AFTER SEARCH
2784	004504	074512		EMT273	
2785	004506	075526		EHT1	
2786	004510	075652		EDT1	
2787	004512	075742		EFT1	
2788					
2789					
2790			;ERROR	274	PARITY ERROR OCCURRED WHILE WRITING REMOTE
2791			:		REGISTERS BUT MXF DID NOT SET
2792	004514	074530		EMT274	
2793	004516	075526		EHT1	
2794	004520	075652		EDT1	
2795	004522	075742		EFT1	
2796					
2797					
2798			;ERROR	275	MXF ERROR - COMPOSITE ERROR OCCURRED BEFORE DATA
2799			:		COMMAND STARTED
2800	004524	074546		EMT275	
2801	004526	075526		EHT1	
2802	004530	075652		EDT1	
2803	004532	075742		EFT1	
2804					
2805					
2806			;ERROR	276	"OPI" ERROR DURING DATA TRANSFER BECAUSE "MOL" WAS
2807			:		ZERO
2808	004534	074560		EMT276	
2809	004536	075526		EHT1	
2810	004540	075652		EDT1	
2811	004542	075742		EFT1	
2812					
2813					
2814			;ERROR	277	"OPI" ERROR DURING DATA TRANSFER BECAUSE 1) ON
2815			:		CYLINDER DIDN'T DROP OR 2) SEARCH TIMED OUT OR
2816			:		3) RUN TIMED OUT

2817	004544	074574	EMT277	
2818	004546	075526	EHT1	
2819	004550	075652	EDT1	
2820	004552	075742	EFT1	
2821				
2822				
2823				; ERROR 300 "IVC" ERROR DURING DATA TRANSFER BECAUSE VOLUME
2824				;
2825	004554	074612	EMT300	
2826	004556	075526	EHT1	
2827	004560	075652	EDT1	
2828	004562	075742	EFT1	
2829				
2830				
2831				; ERROR 301 ERRONEOUS "IVC" ERROR DURING DATA TRANSFER - VOLUME
2832				;
2833	004564	074632	EMT301	
2834	004566	075526	EHT1	
2835	004570	075652	EDT1	
2836	004572	075742	EFT1	
2837				
2838				
2839				; ERROR 302 "ILR" ERROR DURING DATA TRANSFER
2840	004574	074654	EMT302	
2841	004576	075526	EHT1	
2842	004600	075652	EDT1	
2843	004602	075742	EFT1	
2844				
2845				
2846				; ERROR 303 "ILF" ERROR DURING DATA TRANSFER
2847	004604	074666	EMT303	
2848	004606	075526	EHT1	
2849	004610	075652	EDT1	
2850	004612	075742	EFT1	
2851				
2852				
2853				; ERROR 304 "RMR" ERROR DURING DATA TRANSFER
2854	004614	074700	EMT304	
2855	004616	075526	EHT1	
2856	004620	075652	EDT1	
2857	004622	075742	EFT1	
2858				
2859				
2860				; ERROR 305 INCORRECT "IAE" STATUS DURING DATA TRANSFER
2861	004624	074712	EMT305	
2862	004626	075526	EHT1	
2863	004630	075652	EDT1	
2864	004632	075742	EFT1	
2865				
2866				
2867				; ERROR 306 "SKI" ERROR DURING DATA TRANSFER
2868	004634	074724	EMT306	
2869	004636	075526	EHT1	
2870	004640	075652	EDT1	
2871	004642	075742	EFT1	
2872				

2873				
2874			;ERROR	307 DRIVE DID NOT DETECT SKI ERROR DUE TO CYLINDER
2875	004644	074734		EMT307
2876	004646	075526		EHT1
2877	004650	075652		EDT1
2878	004652	075742		EFT1
2879				
2880				
2881			;ERROR	310 DEVICE FAULT DURING DATA TRANSFER
2882	004654	074754		EMT310
2883	004656	075526		EHT1
2884	004660	075652		EDT1
2885	004662	075742		EFT1
2886				
2887				
2888			;ERROR	311 LOSS OF BIT CLOCK DURING DATA TRANSFER
2889	004664	074766		EMT311
2890	004666	075526		EHT1
2891	004670	075652		EDT1
2892	004672	075742		EFT1
2893				
2894				
2895			;ERROR	312 LOSS OF SYSTEM CLOCK DURING DATA TRANSFER
2896	004674	075000		EMT312
2897	004676	075526		EHT1
2898	004700	075652		EDT1
2899	004702	075742		EFT1
2900				
2901				
2902			ERROR	313 UNSAFE ERROR DURING DATA TRANSFER (DVC = 0)
2903	004704	075012		EMT313
2904	004706	075526		EHT1
2905	004710	075652		EDT1
2906	004712	075742		EFT1
2907				
2908				
2909			;ERROR	314 DRIVE TIMING ERROR DURING DATA TRANSFER
2910	004714	075032		EMT314
2911	004716	075526		EHT1
2912	004720	075652		EDT1
2913	004722	075742		EFT1
2914				
2915				
2916			;ERROR	315 WRITE LOCK ERROR
2917	004724	075044		EMT315
2918	004726	075526		EHT1
2919	004730	075652		EDT1
2920	004732	075742		EFT1
2921				
2922				
2923			;ERROR	316 ERRONEOUS WRITE LOCK ERROR
2924	004734	075056		EMT316
2925	004736	075526		EHT1
2926	004740	075652		EDT1
2927	004742	075742		EFT1
2928				

2929					
2930			;ERROR	317	HEADER CRC ERROR DURING DATA TRANSFER
2931	004744	075070		EMT317	
2932	004746	075526		EHT1	
2933	004750	075652		EDT1	
2934	004752	075742		EFT1	
2935					
2936					
2937			;ERROR	320	FORMAT ERROR DURING DATA TRANSFER
2938	004754	075100		EMT320	
2939	004756	075526		EHT1	
2940	004760	075652		EDT1	
2941	004762	075742		EFT1	
2942					
2943					
2944			;ERROR	321	HEADER COMPARE ERROR DURING DATA TRANSFER
2945	004764	075110		EMT321	
2946	004766	075526		EHT1	
2947	004770	075652		EDT1	
2948	004772	075742		EFT1	
2949					
2950					
2951			;ERROR	322	HEADER ERRORS SHOULD NOT BE SET
2952	004774	075120		EMT322	
2953	004776	075526		EHT1	
2954	005000	075652		EDT1	
2955	005002	075742		EFT1	
2956					
2957					
2958			;ERROR	323	DATA CHECK ERROR DURING DATA TRANSFER
2959	005004	075126		EMT323	
2960	005006	075526		EHT1	
2961	005010	075652		EDT1	
2962	005012	075742		EFT1	
2963					
2964					
2965			;ERROR	324	CORRECTABLE DATA CHECK ERROR DURING DATA TRANSFER
2966	005014	075136		EMT324	
2967	005016	075526		EHT1	
2968	005020	075652		EDT1	
2969	005022	075742		EFT1	
2970					
2971					
2972			;ERROR	325	UNCORRECTABLE DATA CHECK ERROR DURING DATA TRANSFER
2973	005024	075150		EMT325	
2974	005026	075526		EHT1	
2975	005030	075652		EDT1	
2976	005032	075742		EFT1	
2977					
2978					
2979			;ERROR	326	DATA PARITY ERROR DURING READ COMMAND
2980	005034	075162		EMT326	
2981	005036	075526		EHT1	
2982	005040	075652		EDT1	
2983	005042	075742		EFT1	
2984					

2985				
2986			;ERROR 327	OFFSET MODE NOT RESET BY WRITE COMMAND
2987	005044	075200	EMT327	
2988	005046	075526	EHT1	
2989	005050	075652	EDT1	
2990	005052	075742	EFT1	
2991				
2992				
2993			;ERROR 330	DATA PARITY ERROR DURING WRITE COMMAND
2994	005054	075212	EMT330	
2995	005056	075526	EHT1	
2996	005060	075652	EDT1	
2997	005062	075742	EFT1	
2998				
2999				
3000			;ERROR 331	WRITE CLOCK FAILURE DURING WRITE COMMAND
3001	005064	075222	EMT331	
3002	005066	075526	EHT1	
3003	005070	075652	EDT1	
3004	005072	075742	EFT1	
3005				
3006				
3007			;ERROR 332	DATA LATE ERROR DURING DATA TRANSFER
3008	005074	075234	EMT332	
3009	005076	075526	EHT1	
3010	005100	075652	EDT1	
3011	005102	075742	EFT1	
3012				
3013				
3014			;ERROR 333	PIP STIL SET AFTER DATA TRANSFER - SKI = 0
3015	005104	075246	EMT333	
3016	005106	075526	EHT1	
3017	005110	075652	EDT1	
3018	005112	075742	EFT1	
3019				
3020				
3021			;ERROR 334	LOST MOL DURING DATA TRANSFER - OPI = 0
3022	005114	075264	EMT334	
3023	005116	075526	EHT1	
3024	005120	075652	EDT1	
3025	005122	075742	EFT1	
3026				
3027				
3028			;ERROR 335	LOST VOLUME VALID DURING DATA TRANSFER - IVC = 0
3029	005124	075302	EMT335	
3030	005126	075526	EHT1	
3031	005130	075652	EDT1	
3032	005132	075742	EFT1	
3033				
3034				
3035			;ERROR 336	DATA READ DOES NOT COMPARE WITH DATA WRITTEN
3036	005134	075322	EMT336	
3037	005136	075610	EHT336	
3038	005140	075706	EDT336	
3039	005142	075776	EFT336	
3040				

3041				
3042			;ERROR	337 WRITE CHECK ERROR NOT DETECTED
3043	005144	075332		EMT337
3044	005146	075622		EHT337
3045	005150	075716		EDT337
3046	005152	076006		EFT337
3047				
3048				
3049			;ERROR	340 WRITE CHECK ERROR AT UNEXPECTED ADDRESS
3050	005154	075342		EMT340
3051	005156	075610		EHT336
3052	005160	075706		EDT336
3053	005162	075776		EFT336
3054				
3055				
3056			;ERROR	341 INCORRECT DATA DURING WRITE CHECK ERROR
3057	005164	075354		EMT341
3058	005166	075610		EHT336
3059	005170	075706		EDT336
3060	005172	075776		EFT336
3061				
3062				
3063			;ERROR	342 "IVC" ERROR NOT DETECTED DURING DATA TRANSFER
3064	005174	075362		EMT342
3065	005176	075526		EHT1
3066	005200	075652		EDT1
3067	005202	075742		EFT1
3068				
3069				
3070			;ERROR	343 "FER" NOT DETECTED DURING DATA TRANSFER
3071	005204	075374		EMT343
3072	005206	075526		EHT1
3073	005210	075652		EDT1
3074	005212	075742		EFT1
3075				
3076				
3077			;ERROR	344 "HCE" NOT DETECTED DURING DATA TRANSFER
3078	005214	075406		EMT344
3079	005216	075634		EHT344
3080	005220	075726		EDT344
3081	005222	076016		EFT344
3082				
3083				
3084			;ERROR	345 "BSE" NOT DETECTED DURING DATA TRANSFER
3085	005224	075420		EMT345
3086	005226	075526		EHT1
3087	005230	075652		EDT1
3088	005232	075742		EFT1
3089				
3090				
3091			;ERROR	346 HEADER ERROR WAS DETECTED W/ HCI SET
3092	005234	075430		EMT346
3093	005236	075526		EHT1
3094	005240	075652		EDT1
3095	005242	075742		EFT1
3096				

3097				
3098			;ERROR 347	DATA TRANSFER NOT ABORTED W/ COMP ERROR SET
3099	005244	075444	EMT347	
3100	005246	075526	EHT1	
3101	005250	075652	EDT1	
3102	005252	075742	EFT1	
3103				
3104				
3105			;ERROR 350	LOST VOLUME VALID DURING SEARCH - "IVC" = 0
3106	005254	075456	EMT350	
3107	005256	075526	EHT1	
3108	005260	075652	EDT1	
3109	005262	075742	EFT1	
3110				
3111				
3112			;ERROR 351	"ATA" DID NOT SET DURING SEARCH
3113	005264	075474	EMT351	
3114	005266	075526	EHT1	
3115	005270	075652	EDT1	
3116	005272	075742	EFT1	
3117				
3118				
3119			;ERROR 352	PROGRAM TIMEOUT WHILE TESTING RMLA
3120	005274	075504	EMT352	
3121	005276	000000	0	
3122	005300	000000	0	
3123	005302	000000	0	
3124				
3125				
3126			;ERROR 353	LOOK AHEAD TEST FAILS
3127	005304	075510	EMT353	
3128	005306	075646	EHT353	
3129	005310	075740	EDT353	
3130	005312	076026	EFT353	
3131				
3132				
3133			;ERROR 354	BSE SHOULD NOT BE SET
3134	005314	075520	EMT354	
3135	005316	075526	EHT1	
3136	005320	075652	EDT1	
3137	005322	075742	EFT1	
3138				
3139				
3140			;PUT ERROR TABLE HERE	
3141			.SBTTL ERROR TABLE USAGE	
3142				
3143			;THE ERROR TABLE ABOVE CONSISTS OF FOUR WORD ENTRIES FOR EACH ERROR	
3144			NUMBER, I.E.,	
3145			:	
3146			EMT - ERROR MESSAGE TABLE ADDRESS	
3147			EHT - ERROR HEADER TABLE ADDRESS	
3148			EDT - ERROR DATA TABLE ADDRESS	
3149			EFT - ERROR FORMAT TABLE ADDRESS	
3150			:	
3151			;THE EMT ENTRY IS THE ADDRESS OF THE TABLE OF ERROR MESSAGE STRINGS	
3152			;FOR THE PARTICULAR ERROR. EACH ERROR MESSAGE TABLE LISTS THE ADDRESS	

3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171

: OF ONE OR MORE ERROR MESSAGE STRINGS WHICH ARE TO BE FORMATTED AND
: TYPED BY THE ERROR TYPE SUBROUTINE. IF THE EMT ENTRY IS ZERO, THERE IS
: NO MESSAGE TO BE TYPED FOR THE ERROR.

: SIMILARLY, THE EHT, EDT, AND EFT ENTRIES ARE ADDRESSES OF TABLES
: OF HEADER, DATA AND FORMAT INFORMATION FOR A GIVEN ERROR. EACH ENTRY
: IN THE ERROR HEADER TABLE MAY OR MAY NOT HAVE AN ASSOCIATED LINE OF
: DATA, HOWEVER, EACH DATA LINE MUST HAVE AN ASSOCIATED FORMAT AND
: HEADER. THAT IS, A HEADER LINE MAY BE PRINTED WITHOUT ANY DATA,
: BUT A DATA LINE IS NOT PRINTED WITHOUT A HEADER, AND EACH DATA LINE
: MUST ALSO HAVE A FORMAT.

: IN SUMMARY,

: EACH NONZERO ENTRY IS THE ADDRESS OF A TABLE,
: EACH TABLE IS A LIST OF ADDRESSES WHICH DEFINES THE LOCATIONS
: OF MESSAGE STRINGS, HEADERS, DATA OR FORMAT.

```

3172 .SBTTL START OF PROGRAM
3173
3174
3175 005324 START:
3176
3177 :CLEAR AND SETUP FOR TEST EXECUTION
3178 005324 000240 NOP
3179 005326 000005 RESET ;INITIALIZE THE SYSTEM
3180 005330 013746 000300 MOV PR6, -(SP) ;PUT NEW PS ON STACK
3181 005334 012746 005342 MOV #64$, -(SP) ;PUT NEW PC ON STACK
3182 005340 000002 RTI ;POP NEW PC AND PS
3183 005342
3184
3185 .SBTTL INITIALIZE THE COMMON TAGS
3186 ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
3187 005342 012706 001114 MOV #CMTAG, R6 ;FIRST LOCATION TO BE CLEARED
3188 005346 005026 CLR (R6)+ ;CLEAR MEMORY LOCATION
3189 005350 022706 001154 CMP #SWR, R6 ;;DONE?
3190 005354 001374 BNE -6 ;LOOP BACK IF NO
3191 005356 012706 001100 MOV #STACK, SP ;SETUP THE STACK POINTER
3192 ;;INITIALIZE A FEW VECTORS
3193 005362 012737 064600 000020 MOV #SCOPE, @IOTVEC ;IOT VECTOR FOR SCOPE ROUTINE
3194 005370 012737 000340 000022 MOV #340, @IOTVEC+2 ;LEVEL 7
3195 005376 012737 065174 000030 MOV #ERROR, @EMTVEC ;EMT VECTOR FOR ERROR ROUTINE
3196 005404 012737 000340 000032 MOV #340, @EMTVEC+2 ;LEVEL 7
3197 005412 012737 066734 000034 MOV #TRAP, @TRAPVEC ;TRAP VECTOR FOR TRAP CALLS
3198 005420 012737 000340 000036 MOV #340, @TRAPVEC+2 ;LEVEL 7
3199 005426 012737 067042 000024 MOV #SPWRDN, @PWRVEC ;POWER FAILURE VECTOR
3200 005434 012737 000340 000026 MOV #340, @PWRVEC+2 ;LEVEL 7
3201 005442 013737 037042 037034 MOV $ENDCT, $EOPCT ;SETUP END-OF-PROGRAM COUNTER
3202 005450 005037 001206 CLR $TIMES ;INITIALIZE NUMBER OF ITERATIONS
3203 005454 005037 001210 CLR $ESCAPE ;CLEAR THE ESCAPE ON ERROR ADDRESS
3204 005460 112737 000001 001131 MOV #1, $ERMAX ;ALLOW ONE ERROR PER TEST
3205 005466 012737 005466 001122 MOV #., $LPADR ;INITIALIZE THE LOOP ADDRESS FOR SCOPE
3206 005474 012737 005474 001124 MOV #., $LPERR ;SETUP THE ERROR LOOP ADDRESS
3207 ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
3208 ;;EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
3209 005502 013746 000004 MOV @ERRVEC, -(SP) ;SAVE ERROR VECTOR
3210 005506 012737 005542 000004 MOV #65$, @ERRVEC ;SET UP ERROR VECTOR
3211 005514 012737 177570 001154 MOV #DSWR, SWR ;SETUP FOR A HARDWARE SWITCH REGISTER
3212 005522 012737 177570 001156 MOV #DDISP, DISPLAY ;AND A HARDWARE DISPLAY REGISTER
3213 005530 022777 177777 173416 CMP #-1, @SWR ;TRY TO REFERENCE HARDWARE SWR
3214 005536 001012 BNE 67$ ;BRANCH IF NO TIMEOUT TRAP OCCURRED
3215 ;AND THE HARDWARE SWR IS NOT = -1
3216 005540 000403 BR 66$ ;BRANCH IF NO TIMEOUT
3217 005542 012716 005550 65$: MOV #66$, (SP) ;SET UP FOR TRAP RETURN
3218 005546 000002 RTI
3219 005550 012737 000176 001154 66$: MOV #SWREG, SWR ;POINT TO SOFTWARE SWR
3220 005556 012737 000174 001156 MOV #DISPREG, DISPLAY
3221 005564 012637 000004 67$: MOV (SP)+, @ERRVEC ;RESTORE ERROR VECTOR
3222
3223 005570 005037 001230 CLR $PASS ;CLEAR PASS COUNT
3224 005574 132737 000200 001243 BITB #APTSIZE, $ENVM ;TEST USER SIZE UNDER APT
3225 005602 001403 BEQ 68$ ;YES, USE NON-APT SWITCH
3226 005604 012737 001244 001154 MOV #SSWREG, SWR ;NO, USE APT SWITCH REGISTER
3227 005612 68$:

```



```

3228 .SBTTL TYPE PROGRAM NAME
3229 ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
3230 005612 005227 177777 INC #1 ;;FIRST TIME?
3231 005616 001052 BNE 69$ ;;BRANCH IF NO
3232 005620 022737 037176 000042 CMP #SENDAD, @#42 ;;ACT-11?
3233 005626 001446 BEQ 69$ ;;BRANCH IF YES
3234 005630 104401 005676 TYPE 70$ TYPE ASCIZ STRING
3235 .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
3236 005634 005737 000042 TST @#42 ;;ARE WE RUNNING UNDER XXDP/ACT?
3237 005640 001012 BNE 71$ ;;BRANCH IF YES
3238 005642 123727 001242 000001 CMPB $ENV, #1 ;;ARE WE RUNNING UNDER APT?
3239 005650 001406 BEQ 71$ ;;BRANCH IF YES
3240 005652 023727 001154 000176 CMP SWR, #SWREG ;;SOFTWARE SWITCH REG SELECTED?
3241 005660 001005 BNE 72$ ;;BRANCH IF NO
3242 005662 104407 GTSWR ;;GET SOFT-SWR SETTINGS
3243 005664 000403 BR 72$
3244 005666 112737 000001 001150 71$: MOVB #1, $AUTOB ;;SET AUTO-MODE INDICATOR
3245 005674 72$:
3246 005674 000423 BR 69$ ;;GET OVER THE ASCIZ
3247 ;;70$: .ASCIZ <CRLF>@RM03/RM02 FUNCTIONAL TEST, PART 3 @<CRLF>
3248 69$:
3249
3250
3251 ;FIND OUT IF PROGRAM IS RUNNING IN STANDALONE MODE
3252 005744 122737 000077 000041 CMPB #77, @#41 ;LOAD FROM THE RM03 FOR XXDP
3253 005752 001003 BNE 5$ ;BRANCH IF NOT
3254 005754 104401 070565 TYPE ,XDPMG ;TYPE CHANGE PACK MESSAGE
3255 005760 000000 HALT ;THEN HALT
3256 005762 5$:
3257 005762 005737 000042 TST 42 ;IS LOC 42 ZERO ??
3258 005766 001003 BNE 10$ ;NO - NOT IN STANDALONE
3259 005770 105737 001242 TSTB $ENV ;IS APT ENVIRONMENT ZERO ??
3260 005774 001451 BEQ STANDALONE ;YES - PROGRAM IN STANDALONE
3261 005776 10$:
3262
3263 ;PROGRAM NOT RUNNING IN STANDALONE - SEE IF SIZING IS ALLOWED
3264 005776 132737 000200 001243 XSIZ: BITB #BIT7, $ENV ;SIZING ALLOWED ??
3265 006004 001043 BNE 20$ ;NO
3266 ; MOV #377, $DEV ;YES - SET DEVICE MAP FOR ALL DEVICES
3267 006006 005037 001300 CLR $DEV ;CLEAR THE BIT MAP
3268 006012 005001 R1 ;START FROM THE DRIVE 0
3269 006014 012704 000001 MOV #BIT0, R4 ;BIT MAP FOR DRIVE 0
3270 006020 013700 001276 MOV $BASE, R0 ;LOAD THE BASE ADDRESS
3271 006024 012760 000040 000010 15$: MOV #CLR, RMCS2(R0) ;MASS BUS CLEAR
3272 006032 010160 000010 MOV R1, RMCS2(R0) ;LOAD THE DRIVE ADDRESS
3273 006036 016003 000010 MOV RMCS2(R0), R3 ;READ STATUS REG TO SEE NED BIT
3274 006042 032703 010000 BIT #NED, R3 ;NED BIT SET ?
3275 006046 001010 BNE 16$ ;BRANCH IF SO
3276 006050 016003 000000 MOV RMCS1(R0), R3 ;READ DVA BIT
3277 006054 032703 004000 BIT #DVA, R3 ;DVA SET
3278 006060 001403 BEQ 16$ ;BRANCH IF NOT
3279 006062 050437 001300 BIS R4, $DEV ;SET THE DEVICE MAP
3280 006066 000405 BR 17$ ;NOT TYPE THE NON-EXIST MESS
3281 006070 16$:
3282 006070 104401 070652 TYPE ,NOTEX ;TYPE NOT EXIST DRIVE
3283 006074 010146 MOV R1, -(SP) ;DRIVE NUMBER

```

F06

CZRMEBO RM03/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 70
GET VALUE FOR SOFTWARE SWITCH REGISTER

SEQ 0070

3284	006076	104403	
3285	006100	006	
3286	006101	000	
3287	006102	005201	
3288	006104	006304	
3289	006106	022701	000007
3290	006112	103344	
3291	006114		
3292			
3293			
3294	006114	000137	006746

	TYP0S		
	.BYTE	6	
	.BYTE	0	
17\$:	INC	R1	; INCREMENT THE DRIVE ADDRESS
	ASL	R4	; SET UP BIT MAP FOR THE NEXT DRIVE
	CMP	#7 R1	; ALL DRIVES ARE CHECKED /
	BHIS	15\$; BRANCH IF NOT
20\$:			
	; GO TO COMMON START CODE		
	JMP	CMNSTART	

```

3295 006120          STANDALONE:
3296 006120 004737 065404      JSR      PC,$TKINT      ;INITIALIZE CONSOLE
3297
3298          ;SEE IF THIS IS THE FIRST START AFTER PROGRAM WAS LOADED
3299 006124 005327 000001      DEC      #1           ;FIRST START ??
3300 006130 100024          BPL      10$         ;YES !!
3301
3302
3303          ;SEE IF THE USER WANTS TO KEEP SAME DEVICES FOR TESTING
3304 006132 104401 070074      5$:      TYPE      ,CNSLOO      ;MAINTAIN PREVIOUS PARAMETERS??
3305 006136 104411          RDCHR          ;GET RESPONSE
3306 006140 012637 001176      MOV      (SP)+,$TMP1    ;ECHO RESPONSE
3307 006144 104401 001176      TYPE      $TMP1
3308 006150 123727 001176 000131  CMPB     $TMP1,#'Y      ;YES RESPONSE??
3309 006156 001002          BNE      6$
3310 006160 000137 007100      JMP      READY        ;KEEP PREVIOUS PARAMETERS
3311 006164 123727 001176 000116 6$:      CMPB     $TMP1,#'N      ;NO RESPONSE??
3312 006172 001420          BEQ      20$         ;GET NEW PARAMETERS
3313 006174 104401 067605      TYPE      ,QSTMRK     ;NOT YES OR NO, TYPE ""
3314 006200 000754          BR       5$          ;RETRY
3315 006202          10$:
3316
3317          ;SEE IF OPERATOR WANTS HELP FILE
3318 006202 104401 067607      TYPE      ,HELPOST     ;WANT HELP ??
3319 006206 104411          RDCHR          ;GET RESPONSE
3320 006210 012637 001176      MOV      (SP)+,$TMP1    ;SAVE AND ECHO RESPONSE
3321 006214 104401 001176      TYPE      $TMP1
3322 006220 123727 001176 000131  CMPB     $TMP1,#'Y      ;WAS IT A YES RESPONSE ??
3323 006226 001002          BNE      20$         ;NO - DONT TYPE HELP
3324 006230 104401 107550      TYPE      ,HELP       ;YES - TYPE HELP TEXT
3325 006234          20$:
3326
3327          ;SEE IF USER WANTS TO CHANGE RMO3 UNIBUS ADDRESS
3328 006234 104401 067775      TYPE      ,UBUSQST     ;WANT TO CHANGE ADDRESS ??
3329 006240 104411          RDCHR          ;GET RESPONSE
3330 006242 012637 001176      MOV      (SP)+,$TMP1    ;SAVE AND ECHO RESPONSE
3331 006246 104401 001176      TYPE      $TMP1
3332 006252 104411          RDCHR          ;READ ONE MORE CHARACTER
3333 006254 005726          TST      (SP)+        ;CLEAR THE STACK POINT
3334 006256 123727 001176 000131  CMPB     $TMP1,#'Y      ;WAS IT A YES RESPONSE ??
3335 006264 001137          BNE      30$         ;NO !!
3336 006266          30$:
3337
3338          ;DIALOGUE TO CHANGE THE UNIBUS ADDRESS, AND INTERRUPT VECTOR
3339 006266 104401 070133      TYPE      ,CNSLO1     ;TYPE CURRENT BUS ADDRESS
3340 006272 013746 001276      MOV      $BASE,-(SP)   ;SAVE $BASE FOR TYPEOUT
3341 006276 104402          TYPOC          ;GO TYPE--OCTAL ASCII(ALL DIGITS)
3342 006300 104401 067576      TYPE      ,CLSPRN
3343 006304 104413          RDOCT          ;GET NEW BUS ADDRESS
3344 006306 012637 001176      MOV      (SP)+,$TMP1    ;CARRIAGE RETURN??
3345 006312 001412          BEQ      50$         ;YES-SKIP TO NEXT ENTRY
3346 006314 022737 160000 001176  CMP      #160000,$TMP1 ;BASE ADDRESS IN I/O PAGE??
3347 006322 101403          BLOS     40$         ;YES
3348 006324 104401 070160      TYPE      ,CNSLO2     ;TYPE WARNING MESSAGE
3349 006330 000756          BR       30$         ;RETRY
3350 006332 013737 001176 001276 40$:      MOV      $TMP1,$BASE   ;STORE NEW BUS ADDRESS

```

H06

CZRMEBO RM03/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 72
GET VALUE FOR SOFTWARE SWITCH REGISTER

SEG 0072

3351	006340	113737	001272	001176	50\$:	MOVB	\$VECT1,\$TMP1	;TYPE CURRENT VECTOR ADDRESS
3352	006346	105037	001177			CLRB	\$TMP1+1	
3353	006352	104401	070241			TYPE	,CNSLO3	
3354	006356	013746	001176			MOV	\$TMP1,-(SP)	::SAVE \$TMP1 FOR TYPEOUT
3355	006362	104403				TYPOS		::GO TYPE--OCTAL ASCII
3356	006364	003				.BYTE	3	::TYPE 3 DIGIT(S)
3357	006365	000				.BYTE	0	::SUPPRESS LEADING ZEROS
3358	006366	104401	067576			TYPE	,CLSPRN	
3359	006372	104413				RDOCT		;GET NEW VECTOR ADDRESS
3360	006374	012637	001176			MOV	(SP)+,\$TMP1	;CARRIAGE RETURN??
3361	006400	001412				BEQ	70\$;YES-SKIP TO NEXT ENTRY
3362	006402	022737	001000	001176		CMP	#1000,\$TMP1	;VECTOR ADDRESS < 1000??
3363	006410	101003				BHI	60\$;YES!!
3364	006412	104401	070271			TYPE	,CNSLO4	;TYPE WARNING MESSAGE
3365	006416	000750				BR	50\$;RETRY
3366	006420	113737	001176	001272	60\$:	MOVB	\$TMP1,\$VECT1	;STORE NEW VECTOR ADDRESS
3367	006426	113737	001273	001176	70\$:	MOVB	\$VECT1+1,\$TMP1	;TYPE CURRENT PRIORITY
3368	006434	006237	001176			ASR	\$TMP1	
3369	006440	006237	001176			ASR	\$TMP1	
3370	006444	006237	001176			ASR	\$TMP1	
3371	006450	006237	001176			ASR	\$TMP1	
3372	006454	006237	001176			ASR	\$TMP1	
3373	006460	105037	001177			CLRB	\$TMP1+1	
3374	006464	104401	070345			TYPE	,CNSLOS	
3375	006470	013746	001176			MOV	\$TMP1,-(SP)	::SAVE \$TMP1 FOR TYPEOUT
3376	006474	104403				TYPOS		::GO TYPE--OCTAL ASCII
3377	006476	001				.BYTE	1	::TYPE 1 DIGIT(S)
3378	006477	000				.BYTE	0	::SUPPRESS LEADING ZEROS
3379	006500	104401	067576			TYPE	,CLSPRN	
3380	006504	104413				RDOCT		;GET NEW PRIORITY
3381	006506	012637	001176			MOV	(SP)+,\$TMP1	;CARRIAGE RETURN??
3382	006512	001424				BEQ	90\$;YES-SKIP TO NEXT ENTRY
3383	006514	023727	001176	000007		CMP	\$TMP1,#7	;LEGAL PRIORITY??
3384	006522	002403				BLT	80\$;YES!!
3385	006524	104401	070401			TYPE	,CNSLO6	;TYPE WARNING MESSAGE
3386	006530	000736				BR	70\$;RETRY
3387	006532				80\$:			;STORE NEW PRIORITY
3388	006532	006337	001176			ASL	\$TMP1	
3389	006536	006337	001176			ASL	\$TMP1	
3390	006542	006337	001176			ASL	\$TMP1	
3391	006546	006337	001176			ASL	\$TMP1	
3392	006552	006337	001176			ASL	\$TMP1	
3393	006556	113737	001176	001273		MOVB	\$TMP1,\$VECT1+1	
3394	006564				90\$:			
3395								
3396								
3397	006564	005037	001300			CLR	\$DEVN	;CLEAR DEVICE MAP
3398	006570	104401	070426			TYPE	,CNSLO7	;TYPE INPUT INSTRUCTIONS
3399	006574	104401	067601			TYPE	,PROMPT	;TYPE PROMPTING CHARACTER
3400	006600	104411				RDOCHR		;GET RESPONSE
3401	006602	012637	001176			MOV	(SP)+,\$TMP1	;ECHO RESPONSE
3402	006606	104401	001176			TYPE	\$TMP1	
3403	006612	023727	001176	000101		CMP	\$TMP1,#'A	;TEST ALL DRIVES??
3404	006620	001021				BNE	110\$;NO
3405	006622	000137	005776			JMP	XSIZ	;AUTO SIZE
3406	006626	012737	000377	001300		MOV	#377,\$DEVN	;TEST ALL DEVICES

; DIALOGUE TO INPUT DEVICE NUMBERS

```

3407 006634 000444          BR      140$      ;SKIP TO NEXT ENTRY
3408 006636 104401 067601 100$:    TYPE    ,PROMPT    ;TYPE PROMPTING CHARACTER
3409 006642 104411          ROCHR          ;GET RESPONSE
3410 006644 012637 001176    MOV     (SP)+,$TMP1 ;ECHO RESPONSE
3411 006650 104401 001176    TYPE    $TMP1
3412 006654 023727 001176 000015  CMP     $TMP1,#CR   ;CARRIAGE RETURN??
3413 006662 001431          BEQ     140$
3414 006664 023727 001176 000060 110$:  CMP     $TMP1,#'0   ;NUMBER < 0??
3415 006672 002404          BLI     120$        ;YES!!
3416 006674 023727 001176 000067  CMP     $TMP1,#'7   ;NUMBER > 7??
3417 006702 003403          BLE     130$        ;NO!!
3418 006704 104401 067605 120$:  TYPE    QSTMRK     ;TYPE "?"
3419 006710 000752          BR      100$        ;RETRY
3420 006712 013701 001176 130$:  MOV     $TMP1,R1    ;R1=DRIVE NUMBER
3421 006716 042701 177770    BIC     #1C7,R1
3422 006722 116102 070776    MOVB   ATNTBL(R1),R2 ;DECODE DEVICE NUMBER
3423 006726 042702 177400    BIC     #1C377,R2   ;CLEAR UNUSED BITS
3424 006732 050237 001300    BIS     R2,$DEVN    ;SET DEVICE # IN MAP
3425 006736 122737 000377 001300  CMPB   #377,$DEVN  ;DONE ??
3426 006744 101334          BHI     100$        ;NO
3427 006746
3428

```

```

3429 006746 CMNSTART:
3430
3431 ;ASSEMBLE TEST QUE FROM DEVICE MAP
3432 006746 013700 001300 MOV $DEV,RO ;RO = DEVICE MAP
3433 006752 012701 001452 MOV #TSTQUE+2,R1 ;R1 = ADDRESS OF FIRST ENTRY IN QUE
3434 006756 010137 001450 MOV R1,TSTQUE ;INITIALIZE ENTRY POINTER
3435 006762 012702 000001 MOV #1,R2 ;R2 = DEVICE POINTER
3436 006766 005003 CLR R3 ;R3 = DEVICE NUMBER
3437 006770 030200 10$: BIT R2,RO ;IS THIS DEVICE IN MAP ??
3438 006772 001406 BEQ 20$ ;NO !!
3439 006774 010311 MOV R3,(R1) ;YES - ENTER DEVICE NUMBER IN QUE
3440 006776 116361 070776 000001 MOVB ATNTBL(R3),1(R1) ;ENTER ATTENTION BIT IN QUE
3441 007004 062701 070002 ADD #2,R1 ;ADVANCE ENTRY POINTER
3442 007010 006302 20$: ASL R2 ;ADVANCE DEVICE POINTER
3443 007012 105702 TSTB R2 ;DONE ALL DEVICES ??
3444 007014 001402 BEQ 25$ ;YES
3445 007016 005203 INC R3 ;ADVANCE DEVICE NUMBER
3446 007020 000763 BR 10$ ;ENTER NEXT DEVICE
3447 007022 005011 25$: CLR (R1) ;TERMINATE TEST QUE
3448
3449 ;SIZE FOR CLOCK
3450 007024 004737 044224 JSR PC,SIZCLK ;SEE IF CLOCK PRESENT
3451 007030 000403 BR 40$ ;YES - CLOCK IS PRESENT
3452 007032 104000 30$: ERROR ;NO CLOCK
3453 007034 000000 HALT
3454 007036 000775 BR 30$
3455 007040 40$:
3456 .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
3457 007040 005737 000042 TST #42 ;ARE WE RUNNING UNDER XXDP/ACT?
3458 007044 001012 BNE 64$ ;BRANCH IF YES
3459 007046 123727 001242 000001 CMPB $ENV,#1 ;ARE WE RUNNING UNDER APT?
3460 007054 001406 BEQ 64$ ;BRANCH IF YES
3461 007056 023727 001154 000176 CMP SWR,#SWREG ;SOFTWARE SWITCH REG SELECTED?
3462 007064 001005 BNE 65$ ;BRANCH IF NO
3463 007066 104407 GTSWR ;GET SOFT-SWR SETTINGS
3464 007070 000403 BR 65$
3465 007072 112737 000001 001150 64$: MOVB #1,$AUTOB ;SET AUTO-MODE INDICATOR
3466 007100 65$:
3467 007100 000240 READY: NOP ;READY TO START TEST
3468 007102 005037 001326 CLR CTLFG ;CLEAR THE CONTROL-C FLAG
3469 007106 004737 065404 JSR PC,$TKINT ;INITIALIZE TTY
3470 007112 013746 000300 MOV PR6,-(SP) ;PUT NEW PS ON STACK
3471 007116 012746 007124 MOV #64$,-(SP) ;PUT NEW PC ON STACK
3472 007122 000002 RTI ;POP NEW PC AND PS
3473 007124
3474 007124 117737 172320 001234 64$: MOVB @TSTQUE,$UNIT ;LOAD UNIT NUMBER
3475 007132 005037 001474 CLR MEDENB ;CLEAR MEDIA ENABLE

```

```

3476
3477
3478
3479 007136
3480 007136 000240
3481 007140 012737 007154 001122
3482 007146 012737 007154 001124
3483 007154
3484 007154 012706 001100
3485 007160 013700 001276
3486 007164 013701 001450
3487 007170 012737 000001 001226
3488 007176 005001
3489 007200 013746 000004
3490 007204 013746 000006
3491 007210 012737 007302 000004
3492 007216 012737 000300 000006
3493
3494 007224 110160 000001
3495 007230 010160 000002
3496 007234 016002 000002
3497 007240 010160 000004
3498 007244 016002 000004
3499 007250 010160 000010
3500 007254 016002 000010
3501 007260 010160 000022
3502 007264 016002 000022
3503 007270 012637 000006
3504 007274 012637 000004
3505 007300 000415
3506
3507 007302 022626
3508 007304 012637 000006
3509 007310 012637 000004
3510 007314 104110
3511 007316 005737 000042
3512 007322 001002
3513 007324 000137 005324
3514 007330 000137 037006
3515 007334
3516
3517
3518
3519
3520
3521 007334
3522 007334 000004
3523 007336 000240
3524 007340 012706 001100
3525 007344 013700 001276
3526 007350 013701 001450
3527 007354 012737 000002 001226
3528
3529 007362 004737 052700
3530 007366 000404
3531 007370 000240

```

```

*****
; *TEST 1 CONTROLLER ACCESS TEST
*****
↑ST1:
NOP ;START OF TEST
MOV #1$, $LPADR
MOV #1$, $LPERR
1$:
MOV #STACK, SP ;INITIALIZE STACK POINTER
MOV $BASE, R0 ;R0=UNIBUS ADDRESS
MOV TSTQUE, R1 ;(R1) = DEVICE BEING TESTED
MOV #1, $TESTN ;;SET TEST NUMBER IN APT MAIL BOX
CLR R1
MOV ERRVEC, -(SP) ;;PUSH ERRVEC ON STACK
MOV ERRVEC+2, -(SP) ;;PUSH ERRVEC+2 ON STACK
MOV #3$, ERRVEC
MOV #PR6, ERRVEC+2
MOVB R1, RMCS1+1(R0) ;MOVE HI BYTE TO RMCS1
MOV R1, RMWC(R0) ;MOVE WORD COUNT REGISTER
MOV RMWC(R0), R2
MOV R1, RMBA(R0) ;MOVE BUS ADDRESS REGISTER
MOV RMBA(R0), R2
MOV R1, RMCS2(R0) ;MOVE CONTROL STATUS REGISTER
MOV RMCS2(R0), R2
MOV R1, RMDB(R0) ;MOVE DATA BUFFER
MOV RMDB(R0), R2
MOV (SP)+, ERRVEC+2 ;;POP STACK INTO ERRVEC+2
MOV (SP)+, ERRVEC ;;POP STACK INTO ERRVEC
BR 7$ ;NO BUS TIMEOUT OCCURRED
3$:
CMP (SP)+, (SP)+ ;ADJUST STACK
MOV (SP)+, ERRVEC+2 ;;POP STACK INTO ERRVEC+2
MOV (SP)+, ERRVEC ;;POP STACK INTO ERRVEC
ERROR 110 ;CANNOT ACCESS MASSBUS CONTROLLER
TST 42 ;STAND ALONE MODE??
BNE 5$ ;NO!!
JMP START ;YES-GO GET $BASE
5$:
JMP $EOP ;GO TO END OF PASS HANDLER
7$:
*****
; *TEST 2 DEVICE AVAILABLE TEST
*****
↑ST2:
SCOPE ;SCOPE CALL
NOP ;START OF TEST
MOV #STACK, SP ;INITIALIZE STACK POINTER
MOV $BASE, R0 ;R0=UNIBUS ADDRESS
MOV TSTQUE, R1 ;(R1) = DEVICE BEING TESTED
MOV #2, $TESTN ;;SET TEST NUMBER IN APT MAIL BOX
JSR PC, CNTCLR
BR 2$ ;GO TO 2$ IF NO ERROR
NOP ;RETURN HERE IF ERROR

```

```

3532 007372 104000          ERROR          ;ERROR NUMBER DEFINED BY SUB
3533 007374 000137 007514 2$: JMP 7$          ;GO TO 7$ IF ERROR WAS FOUND
3534 007400
3535 007400 013746 000004  MOV ERRVEC, -(SP)      ;; PUSH ERRVEC ON STACK
3536 007404 013746 000006  MOV ERRVEC+2, -(SP)    ;; PUSH ERRVEC+2 ON STACK
3537 007410 012737 007500 000004  MOV #5$, ERRVEC
3538 007416 013737 000300 000006  MOV PR6, ERRVEC+2
3539
3540 007424 016037 000000 001176  MOV RMCS1(RO), $TMP1  ;GET DVA STATUS
3541 007432 016037 000010 001174  MOV RMCS2(RO), $TMP0  ;GET NED STATUS
3542 007440 012637 000006  MOV (SP)+, ERRVEC+2   ; POP STACK INTO ERRVEC+2
3543 007444 012637 000004  MOV (SP)+, ERRVEC     ; POP STACK INTO ERRVEC
3544 007450 032737 010000 001174  BIT #NED, $TMP0      ;NONEXISTENT DEVICE??
3545 007456 001402          BEQ 3$              ;NO!!
3546 007460 104111          ERROR 111          ;NONEXISTENT DEVICE
3547 007462 000414          BR 7$
3548 007464 032737 004000 001176 3$: BIT #DVA, $TMP1     ;DEVICE AVAILABLE??
3549 007472 001012          BNE 9$             ;YES!!
3550 007474 104112          ERROR 112          ;DEVICE NOT AVAILABLE
3551 007476 000406          BR 7$
3552
3553 007500 022626          5$: CMP (SP)+, (SP)+    ;ADJUST STACK
3554 007502 012637 000006  MOV (SP)+, ERRVEC+2  ; POP STACK INTO ERRVEC+2
3555 007506 012637 000004  MOV (SP)+, ERRVEC    ; POP STACK INTO ERRVEC
3556 007512 104113          ERROR 113          ;BUS TIMEOUT (04 TRAP)
3557 007514 000137 036752 7$: JMP $EOSP
3558
3559 007520          9$:
3560
3561          ;*****
3562          ;*TEST 3          DRIVE TYPE TEST
3563          ;*****
3564          ;*****
3565 007520          †ST3:
3566 007520 000004          SCOPE          ;SCOPE CALL
3567 007522 000240          NOP          ;START OF TEST
3568 007524 012706 001100  MOV #STACK, SP      ;INITIALIZE STACK POINTER
3569 007530 013700 001276  MOV $BASE, RO       ;RO=UNIBUS ADDRESS
3570 007534 013701 001450  MOV TSTQUE, R1     ;(R1) = DEVICE BEING TESTED
3571 007540 012737 000003 0C1226  MOV #3, $TESTN     ;SET TEST NUMBER IN APT MAIL BOX
3572
3573 007546 004737 052700  JSR PC, CNTCLR
3574 007552 000404          BR 2$          ;GO TO 2$ IF NO ERROR
3575 007554 000240          NOP          ;RETURN HERE IF ERROR
3576 007556 104000          ERROR          ;ERROR NUMBER DEFINED BY SUB
3577 007560 000137 007676  JMP 4$          ;GO TO 4$ IF ERROR WAS FOUND
3578 007564
3579 007564 112737 000026 001506 2$: MOVB #RMDT, GETINX  ;SETUP GET INDEX TABLE
3580 007572 112737 000200 001507  MOVB #200, GETINX+1
3581 007600 012737 007702 001356  MOV #5$, RMDTI     ;RMDT INPUT BUFFER = 5$
3582 007606 004737 043536  JSR PC, GET        ;GO GET DRIVE TYPE
3583 007612 000402          BR 3$          ;GO TO 3$ IF NO ERROR
3584 007614 000240          NOP          ;RETURN HERE IF ERROR
3585 007616 104000          ERROR          ;ERROR NUMBER SPECIFIED BY GET
3586 007620 022737 020024 001356 3$: CMP #SNGPRT, RMDTI ;SINGLE PORT RMO3??
3587 007626 001425          BEQ 5$          ;YES!!

```



```

3588 007630 022737 024024 001356      CMP      #DULPRT,RMDTI      ;DUAL PORT RM03??
3589 007636 001421                    BEQ      5$                ;YES!!
3590 007640 022737 020025 001356      CMP      #SNGPRT!BIT0,RMDTI ;SINGLE PROT RM02 ?
3591 007646 001415                    BEQ      5$                ;
3592 007650 022737 024025 001356      CMP      #DULPRT!BIT0,RMDTI ;DUAL PORT RM02 ?
3593 007656 001411                    BEQ      5$                ;
3594 007660 012737 020024 001176      MOV      #SNGPRT,$TMP1
3595 007666 012737 024024 001200      MOV      #DULPRT,$TMP2
3596 007674 104114                    ERROR    114                ;NOT AN RM03
3597 007676 000137 036752          4$:      JMP      SEOSP            ;GO TO SUBPASS HANDLER.
3598
3599 007702          5$:
3600      ;*****
3601      ;*TEST 4      WRITE, READ ZEROS
3602      ;*****
3603      †ST4:
3604      SCOPE                    ;SCOPE CALL
3605      NOP                      ;START OF TEST
3606 007704 000004                    MOV      #STACK,$P        ;INITIALIZE STACK POINTER
3607 007706 012706 001100                    MOV      $BASE,$R0        ;R0=UNIBUS ADDRESS
3608 007712 013700 001276                    MOV      TSTQUE,$R1       ;($R1) = DEVICE BEING TESTED
3609 007716 013701 001450                    MOV      #4,$TESTN        ;SET TEST NUMBER IN APT MAIL BOX
3610 007722 012737 000004 001226
3611      ;*****
3612      ;LOOP #1      FORMAT,WRITE,READ
3613
3614 007730          10$:
3615
3616      ;PREPARE THE DEVICE FOR FORMAT OPERATION
3617 007730 004737 040020                    JSR      PC,TSTPRP        ;PREPARE DEVICE FOR TEST
3618 007734 054130                    .WORD   054130 ;TASK DESCRIPTOR
3619 007736 000404                    BR       20$              ;GO TO 20$ IF NO ERROR
3620 007740 000240                    NOP
3621 007742 104000                    ERROR    ;RETURN HERE IF ERROR
3622 007744 000137 010724                    JMP      350$             ;ERROR # DEFINED BY TSTPRP SUBROUTINE
3623 007750                    ;GO TO 350$ IF ERROR WAS FOUND
3624
3625          20$:
3626 007750 012737 000000 001434                    ;LOAD PARAMETERS AND GENERATE DATA BUFFER
3627 007756 012737 000000 001406                    MOV      #0,$RMDCO        ;CYLINDER = 0
3628 007764 012737 107550 001404                    MOV      #0,$RMDAO        ;TRACK = 0, SECTOR = 0
3629 007772 012737 177376 001402                    MOV      #BUFONE,$RMBAO   ;BUS ADDRESS
3630 010000 012737 010000 001432                    MOV      #(<↑C<2+256.>+1),$RMWCO ;WORD COUNT = 1 SECTOR
3631 010006 012737 000063 001400                    MOV      #FMT16,$RMOFO    ;16 BIT FORMAT
3632                    ;WRITE HEADER AND DATA COMMAND
3633
3634          ;VERIFY SECTOR(S) NOT IN BAD SECTOR FILE
3635 010014 004737 040734                    JSR      PC,BADSCCT       ;CALL BAD SECTOR MODULE
3636 010020 000405                    BR       25$              ;GO TO 25$ IF NO ERROR
3637 010022 104401 067466                    TYPE    ,SCTMSG          ;TYPE BAD SECTOR MESSAGE
3638 010026 104000                    ERROR    ;ERROR # DEFINED BY BADSCCT SUBROUTINE
3639 010030 000137 010724                    JMP      350$             ;GO TO 350$ IF ERROR WAS FOUND
3640
3641 010034 012737 071110 001174          25$:      MOV      #ZEROS,$TMP0     ;STARTING ADDRESS OF PATTERN
3642 010042 012737 000001 001176                    MOV      #1,$TMP1        ;RANGE OF PATTERN
3643 010050 004737 042640                    JSR      PC,$GENBUF       ;GO GENERATE BUFFER FOR FORMAT

```

```

3644
3645
3646 010054 012702 001535
3647 010060 112722 000034
3648 010064 112722 000006
3649 010070 112722 000004
3650 010074 112722 000002
3651 010100 112722 000032
3652 010104 112722 000000
3653 010110 112712 000200
3654 010114
3655
3656
3657 010114 004737 044006
3658 010120 000404
3659 010122 000240
3660 010124 104000
3661 010126 000137 010724
3662 010132
3663
3664
3665 010132 004737 043452
3666
3667
3668 010136 004737 044346
3669
3670
3671 010142 004737 043536
3672 010146 000404
3673 010150 000240
3674 010152 104000
3675 010154 000137 010724
3676 010160
3677
3678
3679 010160 004737 057036
3680 010164 000405
3681 010166 000240
3682 010170 104000
3683 010172 004736
3684 010174 000137 010724
3685 010200
3686
3687
3688 010200 012737 010216 001122
3689 010206 012737 010216 001124
3690 010214 000410
3691
3692
3693
3694
3695 010216
3696
3697
3698
3699 010216 004737 040020

;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATIION
MOV #PUTINX,R2 ;R2 = BYTE ENTRY POSITION
MOVB #RMDC,(R2)+
MOVB #RMDA,(R2)+
MOVB #RMBA,(R2)+
MOVB #RMWC,(R2)+
MOVB #RMOF,(R2)+
MOVB #RMCS1,(R2)+
MOVB #200,(R2) ;TERMINATE TABLE
30$:
;FORMAT THE DRIVE
JSR PC,PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
BR 40$ ;GO TO 40$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY PUT SUBROUTINE
JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
40$:
;SETUP GET REGISTER INDEX TABLE FOR READING STATUS
JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
;WAIT FOR THE FORMAT TO COMPLETE
JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
;GO READ STATUS FOR FORMAT OPERATION
JSR PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
BR 50$ ;GO TO 50$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY GET SUBROUTINE
JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
50$:
;VERIFY NO ERRORS DURING FORMAT
JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
BR 60$ ;GO TO 60$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
JSR PC,2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
60$:
;MOVE LOOP ADDRESSES TO NEXT OPERATION
MOV #70$,SLPADR
MOV #70$,SLPERR
BR 80$ ;SKIP TO WRITE OPERATION
;*****
;LOOP #2 WRITE,READ
70$:
;PREPARE DEVICE FOR WRITE OPERATION
JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST

```

```

3700 010222 054130 .WORD 054130 ;TASK DESCRIPTOR
3701 010224 000404 BR 80$ ;GO TO 80$ IF NO ERROR
3702 010226 000240 NOP ;RETURN HERE IF ERROR
3703 010230 104000 ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
3704 010232 000137 010724 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
3705 010236 80$:
3706
3707
3708 010236 120$:
3709
3710 ;WRITE DATA TO THE DRIVE
3711 010236 012737 107554 001404 MOV #BUFONE+4,RMBA0 ;MOVE MEMORY ADDRESS
3712 010244 012737 177400 001402 MOV #(<C256.+1>,RMWCO ;CHANGE WORD COUNT
3713 010252 012737 000061 001400 MOV #WD!GO,RMCS10 ;WRITE DATA COMMAND
3714 010260 012702 001535 MOV #PUTINX,R2 ;LOAD PUT REGISTER INDEX TABLE
3715 010264 112722 000006 MOVB #RMDA,(R2)+
3716 010270 112722 000034 MOVB #RMOC,(R2)+
3717 010274 112722 000032 MOVB #RMOF,(R2)+
3718 010300 112722 000004 MOVB #RMBA,(R2)+
3719 010304 112722 000002 MOVB #RMWC,(R2)+
3720 010310 112722 000000 MOVB #RMCS1,(R2)+
3721 010314 112722 000200 MOVB #200,(R2)+ ;TERMINATE TABLE
3722
3723 010320 004737 044006 JSR PC,PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
3724 010324 000404 BR 130$ ;GO TO 130$ IF NO ERROR
3725 010326 000240 NOP ;RETURN HERE IF ERROR
3726 010330 104000 ERROR ;ERROR # DEFINED BY PUT SUBROUTINE
3727 010332 000137 010724 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
3728 010336 130$:
3729
3730 ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
3731 010336 004737 043452 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
3732
3733 ;WAIT FOR WRITE COMMAND TO COMPLETE
3734 010342 004737 044346 JSR PC,TIMOUT ;GO TO TIMEOUT SUBROUTINE
3735
3736 ;GO READ STATUS FOR WRITE COMMAND
3737 010346 004737 043536 JSR PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
3738 010352 000404 BR 140$ ;GO TO 140$ IF NO ERROR
3739 010354 000240 NOP ;RETURN HERE IF ERROR
3740 010356 104000 ERROR ;ERROR # DEFINED BY GET SUBROUTINE
3741 010360 000137 010724 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
3742 010364 140$:
3743
3744 ;CHECK FOR ERRORS DURING WRITE OPERATION
3745 010364 004737 044532 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
3746 010370 000405 BR 150$ ;GO TO 150$ IF NO ERROR
3747 010372 000240 NOP ;RETURN HERE IF ERROR
3748 010374 104000 ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
3749 010376 004736 JSR PC,2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
3750 010400 000137 010724 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
3751 010404 150$:
3752 010404 004737 057036 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
3753 010410 000405 BR 160$ ;GO TO 160$ IF NO ERROR
3754 010412 000240 NOP ;RETURN HERE IF ERROR
3755 010414 104000 ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE

```

```

3756 010416 004736          JSR    PC,(SP)+      ;GO BACK FOR MORE ERROR CHECKS
3757 010420 000137 010724  JMP    350$          ;GO TO 350$ IF ERROR WAS FOUND
3758 010424          160$:
3759
3760 010424          170$:
3761 010424 004737 045364    JSR    PC,SECERR     ;GO CHECK FOR SECONDARY ERRORS
3762 010430 000405          BR     180$          ;GO TO 180$ IF NO ERROR
3763 010432 000240          NOP
3764 010434 104000          ERROR  ;RETURN HERE IF ERROR
3765 010436 004736          JSR    PC,(SP)+     ;ERROR # DEFINED BY SECERR SUBROUTINE
3766 010440 000137 010724  JMP    350$          ;GO BACK FOR MORE ERROR CHECKS
3767 010444          180$:          ;GO TO 350$ IF ERROR WAS FOUND
3768
3769          ;CHANGE LOOP ADDRESSES
3770 010444 012737 010462 001122  MOV    #190$,$LPADR
3771 010452 012737 010462 001124  MOV    #190$,$LPERR
3772 010460 000410          BR     200$          ;SKIP TO NEXT OPERATION
3773
3774          ;*****
3775          ;LOOP #3      READ
3776
3777 010462          190$:
3778
3779          ;PREPARE DEVICE FOR READ OPERATION
3780 010462 004737 040020    JSR    PC,TSTPRP    ;PREPARE DEVICE FOR TEST
3781 010466 054130          .WORD 054130 ;TASK DESCRIPTOR
3782 010470 000404          BR     200$          ;GO TO 200$ IF NO ERROR
3783 010472 000240          NOP
3784 010474 104000          ERROR  ;RETURN HERE IF ERROR
3785 010476 000137 010724  JMP    350$          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
3786 010502          200$:          ;GO TO 350$ IF ERROR WAS FOUND
3787
3788 010502          240$:
3789
3790          ;READ DATA FROM DEVICE
3791 010502 012737 110560 001404  MOV    #BUFTWO+4,RMBA0 ;CHANGE MEMORY ADDRESS
3792 010510 012737 000071 001400  MOV    #RD!GO,RMCS10  ;READ DATA COMMAND
3793 010516 012702 001535          MOV    #PUTINX,R2    ;LOAD PUT REGISTER INDEX TABLE
3794 010522 112722 000006          MOVB  #RMDA,(R2)+
3795 010526 112722 000032          MOVB  #RMOF,(R2)+
3796 010532 112722 000034          MOVB  #RMDC,(R2)+
3797 010536 112722 000004          MOVB  #RMBA,(R2)+
3798 010542 112722 000002          MOVB  #RMWC,(R2)+
3799 010546 112722 000000          MOVB  #RMCS1,(R2)+
3800 010552 112712 000200          MCVB  #200,(R2)
3801
3802 010556 004737 044006    JSR    PC,PUT      ;GO WRITE REGISTERS WITH PUT SUBROUTINE
3803 010562 000404          BR     250$          ;GO TO 250$ IF NO ERROR
3804 010564 000240          NOP
3805 010566 104000          ERROR  ;RETURN HERE IF ERROR
3806 010570 000137 010724  JMP    350$          ;ERROR # DEFINED BY PUT SUBROUTINE
3807 010574          250$:          ;GO TO 350$ IF ERROR WAS FOUND
3808
3809          ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
3810 010574 004737 043452    JSR    PC,GETSTS   ;GO TO GETSTS SUBROUTINE
3811

```

```

3812 ;WAIT FOR READ OPERATION TO COMPLETE
3813 010600 004737 044346 JSR PC,TIMOUT ;GO TO TIMEOUT SUBROUTINE
3814
3815 ;GO READ STATUS FOR READ OPERATION
3816 010604 004737 043536 JSR PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
3817 010610 000404 BR 260$ ;GO TO 260$ IF NO ERROR
3818 010612 000240 NOP ;RETURN HERE IF ERROR
3819 010614 104000 ERROR ;ERROR # DEFINED BY GET SUBROUTINE
3820 010616 000137 010724 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
3821 010622
3822
3823 ;CHECK FOR ERRORS DURING READ OPERATION
3824 010622 004737 044532 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
3825 010626 000405 BR 270$ ;GO TO 270$ IF NO ERROR
3826 010630 000240 NOP ;RETURN HERE IF ERROR
3827 010632 104000 ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
3828 010634 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
3829 010636 000137 010724 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
3830
3831 010642 004737 057036 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
3832 010646 000405 BR 280$ ;GO TO 280$ IF NO ERROR
3833 010650 000240 NOP ;RETURN HERE IF ERROR
3834 010652 104000 ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
3835 010654 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
3836 010656 000137 010724 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
3837
3838
3839 010662
3840 010662 004737 045364 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
3841 010666 000405 BR 300$ ;GO TO 300$ IF NO ERROR
3842 010670 000240 NOP ;RETURN HERE IF ERROR
3843 010672 104000 ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
3844 010674 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
3845 010676 000137 010724 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
3846
3847 010702 004737 043104 JSR PC,CMPBUF ;GO COMPARE WRITE, READ DATA BUFFERS
3848 010706 107554 .WORD BUFONE+4 ;STARTING ADDRESS OF WRITE BUFFER
3849 010710 110560 .WORD BUFTWO+4 ;STARTING ADDRESS OF READ BUFFER
3850 010712 000404 BR 310$ ;GO TO 310$ IF NO ERROR
3851 010714 000240 NOP ;RETURN HERE IF ERROR
3852 010716 104000 ERROR ;ERROR # DEFINED BY CMPBUF SUBROUTINE
3853 010720 000137 010724 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
3854
3855
3856 010724
3857
3858
3859
3860 010724
3861 010724 000004
3862 010726 000240
3863 010730 012706 001100
3864 010734 013700 001276
3865 010740 013701 001450
3866 010744 012737 000005 001226
3867

;*****
;TEST 5 WRITE, WRITE CHECK ZEROS
;*****
TST5: SCOPE ;SCOPE CALL
NOP ;START OF TEST
MOV #STACK,SP ;INITIALIZE STACK POINTER
MOV $BASE,R0 ;R0=UNIBUS ADDRESS
MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
MOV #5,$TESTN ;SET TEST NUMBER IN APT MAIL BOX

```

E07

CZRMEBO RM03/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 82
TS WRITE, WRITE CHECK ZEROS

SEQ 0082

```

3868 ;*****
3869 ;LOOP #1          FORMAT,WRITE,WRITE CHECK DATA
3870
3871 010752          10$:
3872
3873 ;PREPARE THE DEVICE FOR FORMAT OPERATION
3874 010752 004737 040020      JSR      PC,TSTPRP      ;PREPARE DEVICE FOR TEST
3875 010756 054130              .WORD      054130      ;TASK DESCRIPTOR
3876 010760 000404              BR        20$          ;GO TO 20$ IF NO ERROR
3877 010762 000240              NOP                    ;RETURN HERE IF ERROR
3878 010764 104000              ERROR        ;ERROR # DEFINED BY TSTPRP SUBROUTINE
3879 010766 000137 011716      JMP        350$        ;GO TO 350$ IF ERROR WAS FOUND
3880 010772
3881
3882 ;LOAD PARAMETERS AND GENERATE DATA BUFFER
3883 010772 012737 000000 001434      MOV      #0,RMDCO      ;CYLINDER = 0
3884 011000 012737 000000 001406      MOV      #0,RMDAO      ;TRACK = 0, SECTOR = 0
3885 011006 012737 107550 001404      MOV      #BUFONE,RMBA0 ;BUS ADDRESS
3886 011014 012737 177376 001402      MOV      #(<C(2+256.)+1>),RMWCO ;WORD COUNT = 1 SECTOR
3887 011022 012737 010000 001432      MOV      #FMT16,RMOFO  ;16 BIT FORMAT
3888 011030 012737 000063 001400      MOV      #WH!GO,RMCS10 ;WRITE HEADER AND DATA COMMAND
3889
3890 ;VERIFY SECTOR(S) NOT IN BAD SECTOR FILE
3891
3892 011036 004737 040734      JSR      PC,BADSCCT    ;CALL BAD SECTOR MODULE
3893 011042 000405              BR        25$          ;GO TO 25$ IF NO ERROR
3894 011044 104401 067466      TYPE      ,SCTMSG      ;TYPE BAD SECTOR MESSAGE
3895 011050 104000              ERROR        ;ERROR # DEFINED BY BADSCCT SUBROUTINE
3896 011052 000137 011716      JMP        350$        ;GO TO 350$ IF ERROR WAS FOUND
3897 011056
3898 011056 012737 071110 001174      MOV      #ZEROS,$TMP0  ;STARTING ADDRESS OF PATTERN
3899 011064 012737 000001 001176      MOV      #1,$TMP1      ;RANGE OF PATTERN
3900 011072 004737 042640      JSR      PC,GENBUF     ;GO GENERATE BUFFER FOR FORMAT
3901
3902 ;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATION
3903 011076 012702 001535      MOV      #PUTINX,R2    ;R2 = BYTE ENTRY POSITION
3904 011102 112722 000034      MOV      #RMDC,(R2)+
3905 011106 112722 000006      MOV      #RMDA,(R2)+
3906 011112 112722 000004      MOV      #RMBA,(R2)+
3907 011116 112722 000002      MOV      #RMWC,(R2)+
3908 011122 112722 000032      MOV      #RMOF,(R2)+
3909 011126 112722 000000      MOV      #RMCS1,(R2)+
3910 011132 112712 000200      MOV      #200,(R2)    ;TERMINATE TABLE
3911 011136
3912
3913 ;FORMAT THE DRIVE
3914 011136 004737 044006      JSR      PC,PUT        ;GO WRITE REGISTERS WITH PUT SUBROUTINE
3915 011142 000404              BR        40$          ;GO TO 40$ IF NO ERROR
3916 011144 000240              NOP                    ;RETURN HERE IF ERROR
3917 011146 104000              ERROR        ;ERROR # DEFINED BY PUT SUBROUTINE
3918 011150 000137 011716      JMP        350$        ;GO TO 350$ IF ERROR WAS FOUND
3919 011154
3920
3921 ;SETUP GET REGISTER INDEX TABLE FOR READING STATUS
3922 011154 004737 043452      JSR      PC,GETSTS     ;GO TO GETSTS SUBROUTINE
3923

```

F07

CZRMEBO RMO3/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 83
TS WRITE, WRITE CHECK ZEROS

SEG 0083

```

3924 ;WAIT FOR THE FORMAT TO COMPLETE
3925 011160 004737 044346 JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
3926
3927 ;GO READ STATUS FOR FORMAT OPERATION
3928 011164 004737 043536 JSR PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
3929 011170 000404 BR 50$ ;GO TO 50$ IF NO ERROR
3930 011172 000240 NOP ;RETURN HERE IF ERROR
3931 011174 104000 ERROR ;ERROR # DEFINED BY GET SUBROUTINE
3932 011176 000137 011716 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
3933 011202
3934
3935 ;VERIFY NO ERRORS DURING FORMAT
3936 011202 004737 057036 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
3937 011206 000405 BR 60$ ;GO TO 60$ IF NO ERROR
3938 011210 000240 NOP ;RETURN HERE IF ERROR
3939 011212 104000 ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
3940 011214 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
3941 011216 000137 011716 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
3942 011222
3943
3944 ;MOVE LOOP ADDRESSES TO NEXT OPERATION
3945 011222 012737 011240 001122 MOV #70$,SLPADR
3946 011230 012737 011240 001124 MOV #70$,SLPERR
3947 011236 000410 BR 80$ ;SKIP TO WRITE OPERATION
3948
3949 ;*****
3950 ;LOOP #2 WRITE,WRITE CHECK DATA
3951
3952 011240
3953
3954
3955 ;PREPARE DEVICE FOR WRITE OPERATION
3956 011240 004737 040020 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
3957 011244 054130 .WORD 054130 ;TASK DESCRIPTOR
3958 011246 000404 BR 80$ ;GO TO 80$ IF NO ERROR
3959 011250 000240 NOP ;RETURN HERE IF ERROR
3960 011252 104000 ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
3961 011254 000137 011716 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
3962 011260
3963
3964
3965 011260
3966
3967 ;WRITE DATA TO THE DRIVE
3968 011260 012737 107554 001404 MOV #BUFONE+4,RMBA0 ;CHANGE MEMORY ADDRESS
3969 011266 012737 177400 001402 MOV #(<C256.+1>,RMWCO ;CHANGE WORD COUNT
3970 011274 012737 000061 001400 MOV #WD!GO,RMCS10 ;WRITE DATA COMMAND
3971 011302 012702 001535 MOV #PUTINX,R2 ;LOAD PUT REGISTER INDEX TABLE
3972 011306 112722 000006 MOVB #RMDA,(R2)+
3973 011312 112722 000034 MUVB #RMDC,(R2)+
3974 011316 112722 000032 MOVB #RMOF,(R2)+
3975 011322 112722 000004 MOVB #RMBA,(R2)+
3976 011326 112722 000002 MOVB #RMWC,(R2)+
3977 011332 112722 000000 MOVB #RMCS1,(R2)+
3978 011336 112722 000200 MOVB #200,(R2)+ ;TERMINATE TABLE
3979

```

```

3980 011342 004737 044006 JSR PC,PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
3981 011346 000404 BR 130$ ;GO TO 130$ IF NO ERROR
3982 011350 000240 NOP ;RETURN HERE IF ERROR
3983 011352 104000 ERROR ;ERROR # DEFINED BY PUT SUBROUTINE
3984 011354 000137 011716 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
3985 011360
130$:
3986
3987 ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
3988 011360 004737 043452 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
3989
3990 ;WAIT FOR WRITE COMMAND TO COMPLETE
3991 011364 004737 044346 JSR PC,TIMOUT ;GO TO TIMEOUT SUBROUTINE
3992
3993 ;GO READ STATUS FOR WRITE COMMAND
3994 011370 004737 043536 JSR PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
3995 011374 000404 BR 140$ ;GO TO 140$ IF NO ERROR
3996 011376 000240 NOP ;RETURN HERE IF ERROR
3997 011400 104000 ERROR ;ERROR # DEFINED BY GET SUBROUTINE
3998 011402 000137 011716 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
3999 011406
140$:
4000
4001 ;CHECK FOR ERRORS DURING WRITE OPERATION
4002 011406 004737 044532 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
4003 011412 000405 BR 150$ ;GO TO 150$ IF NO ERROR
4004 011414 000240 NOP ;RETURN HERE IF ERROR
4005 011416 104000 ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
4006 011420 004736 JSR PC,2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
4007 011422 000137 011716 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
4008 011426
150$:
4009 011426 004737 057036 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
4010 011432 000405 BR 160$ ;GO TO 160$ IF NO ERROR
4011 011434 000240 NOP ;RETURN HERE IF ERROR
4012 011436 104000 ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
4013 011440 004736 JSR PC,2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
4014 011442 000137 011716 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
4015 011446
160$:
4016
4017 170$:
4018 011446 004737 045364 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
4019 011452 000405 BR 180$ ;GO TO 180$ IF NO ERROR
4020 011454 000240 NOP ;RETURN HERE IF ERROR
4021 011456 104000 ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
4022 011460 004736 JSR PC,2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
4023 011462 000137 011716 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
4024 011466
180$:
4025
4026 ;CHANGE LOOP ADDRESSES
4027 011466 012737 011504 001122 MOV #190$,SLPADR
4028 011474 012737 011504 001124 MOV #190$,SLPERR
4029 011502 000410 BR 200$ ;SKIP TO NEXT OPERATION
4030
4031 ;*****
4032 ;LOOP #3 WRITE CHECK DATA
4033
4034 C11504
4035 190$:

```



```

4036 ;PREPARE DEVICE FOR WRITE CHECK DATA OPERATION
4037 011504 004737 040020 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
4038 011510 054130 .WORD 054130 ;TASK DESCRIPTOR
4039 011512 000404 BR 200$ ;GO TO 200$ IF NO ERROR
4040 011514 000240 NOP ;RETURN HERE IF ERROR
4041 011516 104000 ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
4042 011520 000137 011716 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
4043 011524
4044
4045 011524 200$:
4046
4047
4048 011524 012737 000051 001400 ;WRITE CHECK DATA DATA FROM DEVICE
4049 011532 012702 001535 MOV #WCD!GO, RMCS10 ;WRITE CHECK DATA DATA COMMAND
4050 011536 112722 000006 MOV #PUTINX, R2 ;LOAD PUT REGISTER INDEX TABLE
4051 011542 112722 000032 MOVB #RMDA, (R2)+
4052 011546 112722 000034 MOVB #RMOF, (R2)+
4053 011552 112722 000004 MOVB #RMDC, (R2)+
4054 011556 112722 000002 MOVB #RMBA, (R2)+
4055 011562 112722 000000 MOVB #RMWC, (R2)+
4056 011566 112712 000200 MOVB #RMCS1, (R2)+
4057 #200, (R2)
4058 011572 004737 044006 JSR PC,PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
4059 011576 000404 BR 250$ ;GO TO 250$ IF NO ERROR
4060 011600 000240 NOP ;RETURN HERE IF ERROR
4061 011602 104000 ERROR ;ERROR # DEFINED BY PUT SUBROUTINE
4062 011604 000137 011716 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
4063 011610 250$:
4064
4065 ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
4066 011610 004737 043452 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
4067
4068 ;WAIT FOR WRITE CHECK DATA OPERATION TO COMPLETE
4069 011614 004737 044346 JSR PC,TIMOUT ;GO TO TIMEOUT SUBROUTINE
4070
4071 ;GO READ STATUS FOR WRITE CHECK DATA OPERATION
4072 011620 004737 043536 JSR PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
4073 011624 000404 BR 260$ ;GO TO 260$ IF NO ERROR
4074 011626 000240 NOP ;RETURN HERE IF ERROR
4075 011630 104000 ERROR ;ERROR # DEFINED BY GET SUBROUTINE
4076 011632 000137 011716 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
4077 011636 260$:
4078
4079 ;CHECK FOR ERRORS DURING WRITE CHECK DATA OPERATION
4080 011636 004737 044532 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
4081 011642 000405 BR 270$ ;GO TO 270$ IF NO ERROR
4082 011644 000240 NOP ;RETURN HERE IF ERROR
4083 011646 104000 ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
4084 011650 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
4085 011652 000137 011716 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
4086 011656 270$:
4087 011656 004737 057036 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
4088 011662 000405 BR 280$ ;GO TO 280$ IF NO ERROR
4089 011664 000240 NOP ;RETURN HERE IF ERROR
4090 011666 104000 ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
4091 011670 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS

```

```

4092 011672 000137 011716          JMP      350$          ;GO TO 350$ IF ERROR WAS FOUND
4093 011676          280$:
4094
4095 011676          290$:
4096 011676 004737 045364          JSR      PC, SECERR   ;GO CHECK FOR SECONDARY ERRORS
4097 011702 000405          BR       300$        ;GO TO 300$ IF NO ERROR
4098 011704 000240          NOP
4099 011706 104000          ERROR   ;RETURN HERE IF ERROR
4100 011710 004736          JSR      PC, 2(SP)+  ;ERROR # DEFINED BY SECERR SUBROUTINE
4101 011712 000137 011716          JMP      350$        ;GO BACK FOR MORE ERROR CHECKS
4102 011716          300$:
4103
4104 011716          350$:
4105 ;*****
4106 ;*TEST 6 WRITE, WRITE CHECK ZEROS W/ WCE ERROR
4107 ;*****
4108 011716          TST6:
4109 011716 000004          NOP          ;SCOPE CALL
4110 011720 000240          MOV          ;START OF TEST
4111 011722 012706 001100        MOV      #STACK, SP ;INITIALIZE STACK POINTER
4112 011726 013700 001276        MOV      $BASE, RO   ;RO=UNIBUS ADDRESS
4113 011732 013701 001450        MOV      TSTQUE, R1  ;(R1) = DEVICE BEING TESTED
4114 011736 012737 000006 001226  MOV      #6, $TESTN  ;;SET TEST NUMBER IN APT MAIL BOX
4115
4116 ;*****
4117 ;LOOP #1 FORMAT,WRITE,WRITE CHECK DATA
4118
4119 011744          10$:
4120
4121          ;PREPARE THE DEVICE FOR FORMAT OPERATION
4122 011744 004737 040020          JSR      PC, TSTPRP  ;PREPARE DEVICE FOR TEST
4123 011750 054130          .WORD   054130     ;TASK DESCRIPTOR
4124 011752 000404          BR       20$        ;GO TO 20$ IF NO ERROR
4125 011754 000240          NOP
4126 011756 104000          ERROR   ;RETURN HERE IF ERROR
4127 011760 000137 013134          JMP      350$        ;ERROR # DEFINED BY TSTPRP SUBROUTINE
4128 011764          20$:
4129
4130          ;LOAD PARAMETERS AND GENERATE DATA BUFFER
4131 011764 012737 000000 001434        MOV      #0, RMDCO   ;CYLINDER = 0
4132 011772 012737 000000 001406        MOV      #0, RMDAO   ;TRACK = 0, SECTOR = 0
4133 012000 012737 107550 001404        MOV      #BUFONE, RMBAO ;BUS ADDRESS
4134 012006 012737 177376 001402        MOV      <↑C(2+256.)+1>, RMWCO ;WORD COUNT = 1 SECTOR
4135 012014 012737 010000 001432        MOV      #FMT16, RMOFO ;16 BIT FORMAT
4136 012022 012737 000063 001400        MOV      #WH!GO, RMCS10 ;WRITE HEADER AND DATA COMMAND
4137
4138          ;VERIFY SECTOR(S) NOT IN BAD SECTOR FILE
4139
4140 012030 004737 040734          JSR      PC, BADSCT  ;CALL BAD SECTOR MODULE
4141 012034 000405          BR       25$        ;GO TO 25$ IF NO ERROR
4142 012036 104401 067466          TYPE    ,SCTMSG     ;TYPE BAD SECTOR MESSAGE
4143 012042 104000          ERROR   ;ERROR # DEFINED BY BADSCT SUBROUTINE
4144 012044 000137 013134          JMP      350$        ;GO TO 350$ IF ERROR WAS FOUND
4145 012050          25$:
4146 012050 012737 071110 001174        MOV      #ZEROS, $TMP0 ;STARTING ADDRESS OF PATTERN
4147 012056 012737 000001 001176        MOV      #1, $TMP1   ;RANGE OF PATTERN

```

```

4148 012064 004737 042640      JSR      PC,GENBUF      ;GO GENERATE BUFFER FOR FORMAT
4149
4150      ;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATIION
4151 012070 012702 001535      MOV      #PUTINX,R2      ;R2 = BYTE ENTRY POSITION
4152 012074 112722 000034      MOV      #RMDC,(R2)+
4153 012100 112722 000006      MOV      #RMDA,(R2)+
4154 012104 112722 000004      MOV      #RMBA,(R2)+
4155 012110 112722 000002      MOV      #RMWC,(R2)+
4156 012114 112722 000032      MOV      #RMOF,(R2)+
4157 012120 112722 000000      MOV      #RMCS1,(R2)+
4158 012124 112712 000200      MOV      #200,(R2)      ;TERMINATE TABLE
4159 012130
4160
4161      ;FORMAT THE DRIVE
4162 012130 004737 044006      JSR      PC,PUT        ;GO WRITE REGISTERS WITH PUT SUBROUTINE
4163 012134 000404      BR      40$          ;GO TO 40$ IF NO ERROR
4164 012136 000240      NOP
4165 012140 104000      ERROR   ;RETURN HERE IF ERROR
4166 012142 000137 013134      JMP      350$        ;ERROR # DEFINED BY PUT SUBROUTINE
4167 012146
4168
4169      ;GO TO 350$ IF ERROR WAS FOUND
4170
4171      40$:
4172      ;SETUP GET REGISTER INDEX TABLE FOR READING STATUS
4173 012146 004737 043452      JSR      PC,GETSTS    ;GO TO GETSTS SUBROUTINE
4174
4175      ;WAIT FOR THE FORMAT TO COMPLETE
4176 012152 004737 044346      JSR      PC,TIMOUT   ;GO TO TIMEOUT SUBROUTINE
4177
4178      ;GO READ STATUS FOR FORMAT OPERATION
4179 012156 004737 043536      JSR      PC,GET      ;GO READ REGISTERS WITH GET SUBROUTINE
4180 012162 000404      BR      50$          ;GO TO 50$ IF NO ERROR
4181 012164 000240      NOP
4182 012166 104000      ERROR   ;RETURN HERE IF ERROR
4183 012170 000137 013134      JMP      350$        ;ERROR # DEFINED BY GET SUBROUTINE
4184 012174
4185
4186      50$:
4187      ;VERIFY NO ERRORS DURING FORMAT
4188 012174 004737 057036      JSR      PC,DTASTS   ;GO VERIFY RESULTS OF DATA TRANSFER
4189 012200 000405      BR      60$          ;GO TO 60$ IF NO ERROR
4190 012202 000240      NOP
4191 012204 104000      ERROR   ;RETURN HERE IF ERROR
4192 012206 004736      ERROR   ;ERROR # DEFINED BY DTASTS SUBROUTINE
4193 012210 000137 013134      JSR      PC,@(SP)+   ;GO BACK FOR MORE ERROR CHECKS
4194 012214      JMP      350$        ;GO TO 350$ IF ERROR WAS FOUND
4195
4196      60$:
4197      ;MOVE LOOP ADDRESSES TO NEXT OPERATION
4198 012214 012737 012232 001122      MOV      #70$,SLPADR
4199 012222 012737 012232 001124      MOV      #70$,SLPERR
4200 012230 000410      BR      80$          ;SKIP TO WRITE OPERATION
4201
4202      ;*****
4203      ;LOOP #2      WRITE,WRITE CHECK DATA
4204
4205      70$:
4206
4207      ;PREPARE DEVICE FOR WRITE OPERATION

```

K07

CZRMEBO RM03/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23MACY11 30(1046) 23-NOV-77 12:49 PAGE 88
T6 WRITE, WRITE CHECK ZEROS W/ WCE ERROR

SEQ 0088

```

4204 012232 004737 040020      JSR    PC,TSTPRP      ;PREPARE DEVICE FOR TEST
4205 012236 054130              .WORD  054130      ;TASK DESCRIPTOR
4206 012240 000404              BR     80$          ;GO TO 80$ IF NO ERROR
4207 012242 000240              NOP                    ;RETURN HERE IF ERROR
4208 012244 104000              ERROR  ;ERROR # DEFINED BY TSTPRP SUBROUTINE
4209 012246 000137 013134      JMP    350$         ;GO TO 350$ IF ERROR WAS FOUND
4210 012252
4211
4212
4213 012252
4214
4215
4216 012252 012737 177400 001402 ;WRITE DATA TO THE DRIVE
4217 012260 012737 107554 001404   MOV    #(<C256.+1>,RMWCO ;CHANGE WORD COUNT
4218 012266 012737 000061 001400   MOV    #BUFONE+4,RMBA0 ;CHANGE MEMORY ADDRESS
4219 012274 012702 001535   MOV    #WD!GO,RMCS10 ;WRITE DATA COMMAND
4220 012300 112722 000006   MOV    #PUTINX,R2 ;LOAD PUT REGISTER INDEX TABLE
4221 012304 112722 000034   MOV    #RMDA,(R2)+
4222 012310 112722 000032   MOV    #RMDC,(R2)+
4223 012314 112722 000004   MOV    #RMOF,(R2)+
4224 012320 112722 000002   MOV    #RMBA,(R2)+
4225 012324 112722 000000   MOV    #RMWC,(R2)+
4226 012330 112722 000200   MOV    #RMCS1,(R2)+
4227
4228 012334 004737 044006      JSR    PC,PUT        ;GO WRITE REGISTERS WITH PUT SUBROUTINE
4229 012340 000404              BR     130$         ;GO TO 130$ IF NO ERROR
4230 012342 000240              NOP                    ;RETURN HERE IF ERROR
4231 012344 104000              ERROR  ;ERROR # DEFINED BY PUT SUBROUTINE
4232 012346 000137 013134      JMP    350$         ;GO TO 350$ IF ERROR WAS FOUND
4233 012352
4234
4235
4236 012352 004737 043452      ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
4237
4238
4239 012356 004737 044346      JSR    PC,GETSTS    ;GO TO GETSTS SUBROUTINE
4240
4241
4242
4243
4244
4245
4246
4247 012400
4248
4249
4250 012400 004737 044532      ;WAIT FOR WRITE COMMAND TO COMPLETE
4251 012404 000405              JSR    PC,TIMOUT    ;GO TO TIMOUT SUBROUTINE
4252 012406 000240
4253 012410 104000
4254 012412 004736
4255 012414 000137 013134      ;GO READ STATUS FOR WRITE COMMAND
4256 012420
4257 012420 004737 057036      JSR    PC,GET      ;GO READ REGISTERS WITH GET SUBROUTINE
4258 012424 000405              BR     140$         ;GO TO 140$ IF NO ERROR
4259 012426 000240              NOP                    ;RETURN HERE IF ERROR
                          ERROR  ;ERROR # DEFINED BY GET SUBROUTINE
                          JMP    350$         ;GO TO 350$ IF ERROR WAS FOUND
4250 012400 004737 044532      ;CHECK FOR ERRORS DURING WRITE OPERATION
4251 012404 000405              JSR    PC,PRIERR    ;GO CHECK FOR PRIMARY ERRORS
4252 012406 000240              BR     150$         ;GO TO 150$ IF NO ERROR
4253 012410 104000              NOP                    ;RETURN HERE IF ERROR
4254 012412 004736              ERROR  ;ERROR # DEFINED BY PRIERR SUBROUTINE
4255 012414 000137 013134      JSR    PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
4256 012420
4257 012420 004737 057036      JMP    350$         ;GO TO 350$ IF ERROR WAS FOUND
4258 012424 000405              JSR    PC,DTASTS    ;GO VERIFY RESULTS OF DATA TRANSFER
4259 012426 000240              BR     160$         ;GO TO 160$ IF NO ERROR
                          NOP                    ;RETURN HERE IF ERROR

```

L07

CZRME80 RM03/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 89
T6 WRITE, WRITE CHECK ZEROS W/ WCE ERROR

SEQ 0089

```

4260 012430 104000          ERROR          ;ERROR # DEFINED BY DTASTS SUBROUTINE
4261 012432 004736          JSR          PC,2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
4262 012434 000137 013134  JMP          350$      ;GO TO 350$ IF ERROR WAS FOUND
4263 012440          160$:
4264
4265 012440          170$:
4266 012440 004737 045364      JSR          PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
4267 012444 000405          BR          180$      ;GO TO 180$ IF NO ERROR
4268 012446 000240          NOP          ;RETURN HERE IF ERROR
4269 012450 104000          ERROR          ;ERROR # DEFINED BY SECERR SUBROUTINE
4270 012452 004736          JSR          PC,2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
4271 012454 000137 013134  JMP          350$      ;GO TO 350$ IF ERROR WAS FOUND
4272 012460          180$:
4273
4274          ;CHANGE LOOP ADDRESSES
4275 012460 012737 012502 001122  MOV          #190$, $LPADR
4276 012466 012737 012502 001124  MOV          #190$, $LPERR
4277 012474 012737 012506 001210  MOV          #195$, $ESCAPE ;;ESCAPE TO 195$ ON ERROR
4278
4279          ;*****
4280          ;LOOP #3      WRITE CHECK DATA
4281
4282 012502          190$:
4283
4284 012502 012703 000001      MOV          #1,R3      ;R3+WCE BIT POSITION
4285 012506 050337 110552      195$:  BIS          R3,BUFTWO-2 ;CHANGE LAST WORD OF BUFFER
4286
4287          ;PREPARE DEVICE FOR WRITE CHECK DATA OPERATION
4288 012512 004737 040020      JSR          PC,TSTPRP ;PREPARE DEVICE FOR TEST
4289 012516 054130          .WORD      054130 ;TASK DESCRIPTOR
4290 012520 000404          BR          200$      ;GO TO 200$ IF NO ERROR
4291 012522 000240          NOP          ;RETURN HERE IF ERROR
4292 012524 104000          ERROR          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
4293 012526 000137 013134  JMP          350$      ;GO TO 350$ IF ERROR WAS FOUND
4294 012532          200$:
4295
4296 012532          240$:
4297
4298          ;READ DATA FROM DEVICE
4299 012532 012737 000051 001400  MOV          #WCD!GO, RMCS10 ;WRITE CHECK DATA DATA COMMAND
4300 012540 012702 001535          MOV          #PUTINX, R2 ;LOAD PUT REGISTER INDEX TABLE
4301 012544 112722 000006          MOVB         #RMDA, (R2)+
4302 012550 112722 000032          MOVB         #RMOF, (R2)+
4303 012554 112722 000034          MOVB         #RMDC, (R2)+
4304 012560 112722 000004          MOVB         #RMA, (R2)+
4305 012564 112722 000002          MOVB         #RMWC, (R2)+
4306 012570 112722 000000          MOVB         #RMCS1, (R2)+
4307 012574 112712 000200          MOVB         #200, (R2)
4308
4309 012600 004737 044006      JSR          PC,PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
4310 012604 000404          BR          250$      ;GO TO 250$ IF NO ERROR
4311 012606 000240          NOP          ;RETURN HERE IF ERROR
4312 012610 104000          ERROR          ;ERROR # DEFINED BY PUT SUBROUTINE
4313 012612 000137 013134  JMP          350$      ;GO TO 350$ IF ERROR WAS FOUND
4314 012616          250$:
4315

```

```

4316 ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
4317 012616 004737 043452 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
4318
4319 ;WAIT FOR WRITE CHECK DATA OPERATION TO COMPLETE
4320 012622 004737 044346 JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
4321
4322 ;GO READ STATUS FOR WRITE CHECK DATA OPERATION
4323 012626 004737 043536 JSR PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
4324 012632 000404 BR 260$ ;GO TO 260$ IF NO ERROR
4325 012634 000240 NOP ;RETURN HERE IF ERROR
4326 012636 104000 ERROR ;ERROR # DEFINED BY GET SUBROUTINE
4327 012640 000137 013134 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
4328 012644
260$:
4329
4330 ;CHECK FOR ERRORS DURING WRITE CHECK DATA OPERATION
4331 012644 004737 044532 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
4332 012650 000405 BR 270$ ;GO TO 270$ IF NO ERROR
4333 012652 000240 NOP ;RETURN HERE IF ERROR
4334 012654 104000 ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
4335 012656 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
4336 012660 000137 013134 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
4337 012664
270$:
4338 012664 032737 040000 001340 BIT #WCE, RMCS2I ;WAS "WCE" DETECTED??
4339 012672 001030 BNE 285$ ;YES!!
4340 012674 004737 057036 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
4341 012700 000405 BR 280$ ;GO TO 280$ IF NO ERROR
4342 012702 000240 NOP ;RETURN HERE IF ERROR
4343 012704 104000 ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
4344 012706 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
4345 012710 000137 013134 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
4346 012714
280$:
4347 012714 013737 001340 001140 MOV RMCS2I,$GDDAT ;EXPECTED STATUS
4348 012722 052737 040000 001140 BIS #WCE,$GDDAT
4349 012730 013737 001340 001142 MOV RMCS2I,$BDDAT ;RECEIVED STATUS
4350 012736 010337 001174 MOV R3,$TMP0 ;FAILING BIT POSITION
4351 012742 012737 110552 001176 MOV #BUFTWO-2,$TMP1 ;FAILING ADDRESS
4352 012750 104337 ERROR 337 ;WCE NOT DETECTED
4353 012752 000470 BR 350$
4354
285$:
4355 012754
4356 012754 112737 000022 001506 MOVB #RMD8,GETINX ;SETUP GET INDEX TABLE
4357 012762 112737 000200 001507 MOVB #200,GETINX+1
4358 012770 004737 043536 JSR PC,GET ;GO GET THE CONTENTS OF THE DATA BUFFER
4359 012774 000404 BR 290$ ;GO TO 290$ IF NO ERROR
4360 012776 000240 NOP ;RETURN HERE IF ERROR
4361 013000 104000 ERROR ;ERROR DEFINED BY GET SUBROUTINE
4362 013002 000137 013134 JMP 350$ ;GO TO 350$ IF ERROR
4363
290$:
4364 013006
4365 013006 013737 001352 001142 MOV RMDBI,$BDDAT ;RECEIVED DATA
4366 013014 013737 110552 001140 MOV BUFTWO-2,$GDDAT ;EXPECTED DATA
4367 013022 040337 001140 BIC R3,$GDDAT
4368 013026 012737 110552 001134 MOV #BUFTWO-2,$GDADR ;EXPECTED ADDRESS
4369 013034 013737 001334 001136 MOV RMBAI,$BDADR ;RECEIVED ADDRESS
4370 013042 162737 000002 001136 SUB #2,$BDADR ;ADJUST BUS ADDRESS
4371 013050 023737 001134 001136 CMP $GDADR,$BDADR ;ADDRESSES OK??

```

```

4372 013056 001402 BEQ 295$ ;YES!!
4373 013060 104340 ERROR 340 ;WCE AT UNEXPECTED ADDRESS
4374 013062 000424 BR 350$
4375 013064 023737 001140 001142 295$: CMP $GDDAT,$BDDAT ;DATA OK??
4376 013072 001402 BEQ 296$ ;YES!!
4377 013074 104341 ERROR 341 ;UNEXPECTED WCE DATA
4378 013076 000416 BR 350$
4379 013100 296$:
4380
4381 013100 004737 045364 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
4382 013104 000405 BR 300$ ;GO TO 300$ IF NO ERROR
4383 013106 000240 NOP ;RETURN HERE IF ERROR
4384 013110 104000 ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
4385 013112 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
4386 013114 000137 013134 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
4387 013120 300$:
4388
4389 013120 040337 110552 BIC R3,BUFTWO-2 ;RESTORE DATA PATTERN
4390 013124 006303 ASL R3 ;SHIFT TO NEXT BIT
4391 013126 001402 BEQ 350$ ;EXIT IF DONE
4392 013130 000137 012506 JMP 195$ ;REPEAT TEST FOR NEXT DATA BIT
4393 013134 350$:
4394 013134 012737 000000 001210 MOV #0,$ESCAPE ;;ESCAPE TO 0 ON ERROR
4395 013142 012737 011744 001122 MOV #10,$SLPADR ;CHANGE LOOP TO START OF TEST
4396 013150 012737 011744 001124 MOV #10,$SLPERR
4397
4398 ;*****
4399 ;*TEST 7 WRITE, READ ONES
4400 ;*****
4401 ;TST7:
4402 SCOPE ;SCOPE CALL
4403 NOP ;START OF TEST
4404 MOV #STACK,SP ;INITIALIZE STACK POINTER
4405 MOV $BASE,R0 ;R0=UNIBUS ADDRESS
4406 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
4407 MOV #7,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
4408
4409 ;*****
4410 ;LOOP #1 FORMAT,WRITE,READ
4411
4412 10$:
4413 ;PREPARE THE DEVICE FOR FORMAT OPERATION
4414 013204 004737 040020 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
4415 013210 054130 .WORD 054130 ;TASK DESCRIPTOR
4416 013212 000404 BR 20$ ;GO TO 20$ IF NO ERROR
4417 013214 000240 NOP ;RETURN HERE IF ERROR
4418 013216 104000 ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
4419 013220 000137 014200 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
4420 013224 20$:
4421
4422 ;LOAD PARAMETERS AND GENERATE DATA BUFFER
4423 013224 012737 000000 001434 MOV #0,RMDCO ;CYLINDER = 0
4424 013232 012737 000000 001406 MOV #0,RMDAO ;TRACK = 0, SECTOR = 0
4425 013240 012737 107550 001404 MOV #BUFONE,RMBAO ;BUS ADDRESS
4426 013246 012737 177400 001402 MOV #(<C256.+1>),RMWCO ;WORD COUNT = 1 SECTOR
4427 013254 012737 010000 001432 MOV #FMT16,RMFO ;16 BIT FORMAT

```

```

4428 013262 012737 000063 001400      MOV      #WH!GO,RMCS10      ;WRITE HEADER AND DATA COMMAND
4429
4430      ;VERIFY SECTOR(S) NOT IN BAD SECTOR FILE
4431
4432 013270 004737 040734      JSR      PC,BADSCT        ;CALL BAD SECTOR MODULE
4433 013274 000405                BR      25$              ;GO TO 25$ IF NO ERROR
4434 013276 104401 067466      TYPE    ,SCTMSG          ;TYPE BAD SECTOR MESSAGE
4435 013302 104000                ERROR   #                ;ERROR # DEFINED BY BADSCT SUBROUTINE
4436 013304 000137 014200      JMP     350$            ;GO TO 350$ IF ERROR WAS FOUND
4437 013310
4438 013310 012737 071046 001174 25$:      MOV     #ONES,$TMPD      ;STARTING ADDRESS OF PATTERN
4439 013316 012737 000001 001176      MOV     #1,$TMP1         ;RANGE OF PATTERN
4440 013324 004737 042640      JSR    PC,GENBUF         ;GO GENERATE BUFFER FOR FORMAT
4441
4442      ;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATIION
4443      MOV     #PUTINX,R2      ;R2 = BYTE ENTRY POSITION
4444 013330 012702 001535      MOV     #RMDC,(R2)+
4445 013334 112722 000034      MOV     #RMDA,(R2)+
4446 013340 112722 000006      MOV     #RMDA,(R2)+
4447 013344 112722 000004      MOV     #RMDA,(R2)+
4448 013350 112722 000002      MOV     #RMDA,(R2)+
4449 013354 112722 000032      MOV     #RMOF,(R2)+
4450 013360 112722 000000      MOV     #RMCS1,(R2)+
4451 013364 112712 000200      MOV     #200,(R2)      ;TERMINATE TABLE
4452
4453 30$:
4454      ;FORMAT THE DRIVE
4455 013370 004737 044006      JSR    PC,PUT           ;GO WRITE REGISTERS WITH PUT SUBROUTINE
4456 013374 000404                BR      40$              ;GO TO 40$ IF NO ERROR
4457 013376 000240                NOP
4458 013400 104000                ERROR   #                ;RETURN HERE IF ERROR
4459 013402 000137 014200      JMP     350$            ;ERROR # DEFINED BY PUT SUBROUTINE
4460
4461      ;GO TO 350$ IF ERROR WAS FOUND
4462 40$:
4463      ;SETUP GET REGISTER INDEX TABLE FOR READING STATUS
4464      JSR    PC,GETSTS      ;GO TO GETSTS SUBROUTINE
4465
4466      ;WAIT FOR THE FORMAT TO COMPLETE
4467      JSR    PC,TIMOUT      ;GO TO TIMOUT SUBROUTINE
4468
4469      ;GO READ STATUS FOR FORMAT OPERATION
4470      JSR    PC,GET         ;GO READ REGISTERS WITH GET SUBROUTINE
4471 013416 004737 043536      BR      50$              ;GO TO 50$ IF NO ERROR
4472 013422 000404                NOP
4473 013424 000240                ERROR   #                ;RETURN HERE IF ERROR
4474 013426 104000                ERROR   #                ;ERROR # DEFINED BY GET SUBROUTINE
4475 013430 000137 014200      JMP     350$            ;GO TO 350$ IF ERROR WAS FOUND
4476
4477 50$:
4478      ;VERIFY NO ERRORS DURING FORMAT
4479      JSR    PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
4480 013434 004737 057036      BR      60$              ;GO TO 60$ IF NO ERROR
4481 013440 000405                NOP
4482 013442 000240                ERROR   #                ;RETURN HERE IF ERROR
4483 013444 104000                ERROR   #                ;ERROR # DEFINED BY DTASTS SUBROUTINE
4484 013446 004736      JSR    PC,@(SP)+        ;GO BACK FOR MORE ERROR CHECKS
4485 013450 000137 014200      JMP     350$            ;GO TO 350$ IF ERROR WAS FOUND

```



```

4484 ;MOVE LOOP ADDRESSES TO NEXT OPERATION
4485 013454 012737 013472 001122 MOV #70$, $LPADR
4486 013462 012737 013472 001124 MOV #70$, $LPERR
4487 013470 000410 BR 80$ ;SKIP TO WRITE OPERATION
4488
4489 ;*****
4490 ;LOOP #2 WRITE,READ
4491
4492 013472 70$:
4493
4494 ;PREPARE DEVICE FOR WRITE OPERATION
4495 JSR PC, TSTPRP ;PREPARE DEVICE FOR TEST
4496 013472 004737 040020 .WORD 054130 ;TASK DESCRIPTOR
4497 013476 054130 BR 80$ ;GO TO 80$ IF NO ERROR
4498 013500 000404 NOP ;RETURN HERE IF ERROR
4499 013502 000240 ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
4500 013504 104000 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
4501 013506 000137 014200
4502 013512 80$:
4503
4504
4505 013512 120$:
4506
4507 ;WRITE DATA TO THE DRIVE
4508 013512 012737 107554 001404 MOV #BUFONE+4, RMBAO ;CHANGE MEMORY ADDRESS
4509 013520 012737 177400 001402 MOV #(<C256.+1>, RMWCC ;CHANGE WORD COUNT
4510 013526 012737 000061 001400 MOV #WD!GO, RMCS10 ;WRITE DATA COMMAND
4511 013534 012702 001535 MOV #PUTINX, R2 ;LOAD PUT REGISTER INDEX TABLE
4512 013540 112722 000006 MOV #RMDA, (R2)+
4513 013544 112722 000034 MOV #RMDC, (R2)+
4514 013550 112722 000032 MOV #RMOF, (R2)+
4515 013554 112722 000004 MOV #RMBB, (R2)+
4516 013560 112722 000002 MOV #RMWC, (R2)+
4517 013564 112722 000000 MOV #RMCS1, (R2)+
4518 013570 112722 000200 MOV #200, (R2)+ ;TERMINATE TABLE
4519
4520 013574 004737 044006 JSR PC, PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
4521 013600 000404 BR 130$ ;GO TO 130$ IF NO ERROR
4522 013602 000240 NOP ;RETURN HERE IF ERROR
4523 013604 104000 ERROR ;ERROR # DEFINED BY PUT SUBROUTINE
4524 013606 000137 014200 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
4525 013612 130$:
4526
4527 ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
4528 013612 004737 043452 JSR PC, GETSTS ;GO TO GETSTS SUBROUTINE
4529
4530 ;WAIT FOR WRITE COMMAND TO COMPLETE
4531 013616 004737 044346 JSR PC, TIMOUT ;GO TO TIMOUT SUBROUTINE
4532
4533 ;GO READ STATUS FOR WRITE COMMAND
4534 013622 004737 043536 JSR PC, GET ;GO READ REGISTERS WITH GET SUBROUTINE
4535 013626 000404 BR 140$ ;GO TO 140$ IF NO ERROR
4536 013630 000240 NOP ;RETURN HERE IF ERROR
4537 013632 104000 ERROR ;ERROR # DEFINED BY GET SUBROUTINE
4538 013634 000137 014200 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
4539 013640 140$:

```

```

4540
4541 ;CHECK FOR ERRORS DURING WRITE OPERATION
4542 013640 004737 044532 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
4543 013644 000405 BR 150$ ;GO TO 150$ IF NO ERROR
4544 013646 000240 NOP ;RETURN HERE IF ERROR
4545 013650 104000 ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
4546 013652 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
4547 013654 000137 014200 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
4548 013660
4549 013660 004737 057036 150$: JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
4550 013664 000405 BR 160$ ;GO TO 160$ IF NO ERROR
4551 013666 000240 NOP ;RETURN HERE IF ERROR
4552 013670 104000 ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
4553 013672 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
4554 013674 000137 014200 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
4555 013700
4556
4557 013700
4558 013700 004737 045364 170$: JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
4559 013704 000405 BR 180$ ;GO TO 180$ IF NO ERROR
4560 013706 000240 NOP ;RETURN HERE IF ERROR
4561 013710 104000 ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
4562 013712 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
4563 013714 000137 014200 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
4564 013720
4565
4566 ;CHANGE LOOP ADDRESSES
4567 013720 012737 013736 001122 MOV #190$,SLPADR
4568 013726 012737 013736 001124 MOV #190$,SLPERR
4569 013734 000410 BR 200$ ;SKIP TO NEXT OPERATION
4570
4571 ;*****
4572 ;LOOP #3 READ
4573
4574 013736
4575
4576
4577 ;PREPARE DEVICE FOR READ OPERATION
4578 013736 004737 040020 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
4579 013742 054130 .WORD 054130 ;TASK DESCRIPTOR
4580 013744 000404 BR 200$ ;GO TO 200$ IF NO ERROR
4581 013746 000240 NOP ;RETURN HERE IF ERROR
4582 013750 104000 ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
4583 013752 000137 014200 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
4584
4585 013756
4586
4587
4588 ;READ DATA FROM DEVICE
4589 013756 012737 110560 001404 MOV #BUFTWO+4,RMBA0 ;CHANGE MEMORY ADDRESS
4590 013772 012702 001535 001400 MOV #RD!GO,RMCS10 ;READ DATA COMMAND
4591 013776 112722 000006 MOV #PUTINX,R2 ;LOAD PUT REGISTER INDEX TABLE
4592 014002 112722 000032 MOVB #RMDA,(R2)+
4593 014006 112722 000034 MOVB #RMOF,(R2)+
4594 014012 112722 000004 MOVB #RMDC,(R2)+
4595 014016 112722 000002 MOVB #RMBA,(R2)+

```

```

4596 014022 112722 000000      MOVB   #RMCS1,(R2)+
4597 014026 112712 000200      MOVB   #200,(R2)
4598
4599 014032 004737 044006      JSR    PC,PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
4600 014036 000404          BR     250$   ;GO TO 250$ IF NO ERROR
4601 014040 000240          NOP                    ;RETURN HERE IF ERROR
4602 014042 104000          ERROR          ;ERROR # DEFINED BY PUT SUBROUTINE
4603 014044 000137 014200      JMP    350$   ;GO TO 350$ IF ERROR WAS FOUND
4604 014050
4605
4606
4607 014050 004737 043452      ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
4608          JSR    PC,GETSTS ;GO TO GETSTS SUBROUTINE
4609
4610          ;WAIT FOR READ OPERATION TO COMPLETE
4610 014054 004737 044346      JSR    PC,TIMOUT ;GO TO TIMEOUT SUBROUTINE
4611
4612
4613          ;GO READ STATUS FOR READ OPERATION
4613 014060 004737 043536      JSR    PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
4614 014064 000404          BR     260$   ;GO TO 260$ IF NO ERROR
4615 014066 000240          NOP                    ;RETURN HERE IF ERROR
4616 014070 104000          ERROR          ;ERROR # DEFINED BY GET SUBROUTINE
4617 014072 000137 014200      JMP    350$   ;GO TO 350$ IF ERROR WAS FOUND
4618 014076
4619
4620
4621          ;CHECK FOR ERRORS DURING READ OPERATION
4621 014076 004737 044532      JSR    PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
4622 014102 000405          BR     270$   ;GO TO 270$ IF NO ERROR
4623 014104 000240          NOP                    ;RETURN HERE IF ERROR
4624 014106 104000          ERROR          ;ERROR # DEFINED BY PRIERR SUBROUTINE
4625 014110 004736          JSR    PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
4626 014112 000137 014200      JMP    350$   ;GO TO 350$ IF ERROR WAS FOUND
4627 014116
4628          270$:
4628 014116 004737 057036      JSR    PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
4629 014122 000405          BR     280$   ;GO TO 280$ IF NO ERROR
4630 014124 000240          NOP                    ;RETURN HERE IF ERROR
4631 014126 104000          ERROR          ;ERROR # DEFINED BY DTASTS SUBROUTINE
4632 014130 004736          JSR    PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
4633 014132 000137 014200      JMP    350$   ;GO TO 350$ IF ERROR WAS FOUND
4634 014136
4635          280$:
4636          290$:
4637 014136 004737 045364      JSR    PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
4638 014142 000405          BR     300$   ;GO TO 300$ IF NO ERROR
4639 014144 000240          NOP                    ;RETURN HERE IF ERROR
4640 014146 104000          ERROR          ;ERROR # DEFINED BY SECERR SUBROUTINE
4641 014150 004736          JSR    PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
4642 014152 000137 014200      JMP    350$   ;GO TO 350$ IF ERROR WAS FOUND
4643 014156
4644          300$:
4644 014156 004737 043104      JSR    PC,CMPBUF ;GO COMPARE WRITE, READ DATA BUFFERS
4645 014162 107554          .WORD  BUFOFF+4 ;STARTING ADDRESS OF WRITE BUFFER
4646 014164 110560          .WORD  BUFTWO+4 ;STARTING ADDRESS OF READ BUFFER
4647 014166 000404          BR     310$   ;GO TO 310$ IF NO ERROR
4648 014170 000240          NOP                    ;RETURN HERE IF ERROR
4649 014172 104000          ERROR          ;ERROR # DEFINED BY CMPBUF SUBROUTINE
4650 014174 000137 014200      JMP    350$   ;GO TO 350$ IF ERROR WAS FOUND
4651 014200
4651          310$.

```

```

4652
4653 014200
4654
4655
4656
4657 014200
4658 014200 000004
4659 014202 000240
4660 014204 012706 001100
4661 014210 013700 001276
4662 014214 013701 001450
4663 014220 012737 000010 001226
4664
4665
4666
4667
4668 014226
4669
4670
4671 014226 004737 040020
4672 014232 054130
4673 014234 000404
4674 014236 000240
4675 014240 104000
4676 014242 000137 015172
4677 014246
4678
4679
4680 014246 012737 000000 001434
4681 014254 012737 000000 001406
4682 014262 012737 107550 001404
4683 014270 012737 177376 001402
4684 014276 012737 010000 001432
4685 014304 012737 000063 001400
4686
4687
4688
4689 014312 004737 040734
4690 014316 000405
4691 014320 104401 067466
4692 014324 104000
4693 014326 000137 015172
4694 014332
4695 014332 012737 071046 001174
4696 014340 012737 000001 001176
4697 014346 004737 042640
4698
4699
4700 014352 012702 001535
4701 014356 112722 000034
4702 014362 112722 000006
4703 014366 112722 000004
4704 014372 112722 000002
4705 014376 112722 000032
4706 014402 112722 000000
4707 014406 112712 000200

```

```

350$:
;*****
;TEST 10 WRITE, WRITE CHECK ONES
;*****
TST10:
SCOPE                                ;SCOPE CALL
NOP                                  ;START OF TEST
MOV #STACK, SP                       ;INITIALIZE STACK POINTER
MOV $BASE, R0                         ;R0=UNIBUS ADDRESS
MOV TSTQUE, R1                        ;(R1) = DEVICE BEING TESTED
MOV #10, $TESTN                       ;;SET TEST NUMBER IN APT MAIL BOX

;*****
;LOOP #1 FORMAT, WRITE, WRITE CHECK DATA

10$:
;PREPARE THE DEVICE FOR FORMAT OPERATION
JSR PC, TSTPRP                        ;PREPARE DEVICE FOR TEST
WORD 054130 ;TASK DESCRIPTOR
BR 20$                                ;GO TO 20$ IF NO ERROR
NOP                                    ;RETURN HERE IF ERROR
ERROR # DEFINED BY TSTPRP SUBROUTINE ;ERROR # DEFINED BY TSTPRP SUBROUTINE
JMP 350$                               ;GO TO 350$ IF ERROR WAS FOUND

20$:
;LOAD PARAMETERS AND GENERATE DATA BUFFER
MOV #0, RMDCO                          ;CYLINDER = 0
MOV #0, RMDAO                          ;TRACK = 0, SECTOR = 0
MOV #BUFONE, RMBAO                     ;BUS ADDRESS
MOV #(<C<2+256.>+1), RMWCO             ;WORD COUNT = 1 SECTOR
MOV #FMT16, RMOFO                      ;16 BIT FORMAT
MOV #WH!GO, RMCS10                     ;WRITE HEADER AND DATA COMMAND

;VERIFY SECTOR(S) NOT IN BAD SECTOR FILE
JSR PC, BADSCT                          ;CALL BAD SECTOR MODULE
BR 25$                                  ;GO TO 25$ IF NO ERROR
TYPE , SCTMSG                           ;TYPE BAD SECTOR MESSAGE
ERROR # DEFINED BY BADSCT SUBROUTINE ;ERROR # DEFINED BY BADSCT SUBROUTINE
JMP 350$                               ;GO TO 350$ IF ERROR WAS FOUND

25$:
MOV #ONES, $TMP0                        ;STARTING ADDRESS OF PATTERN
MOV #1, $TMP1                            ;RANGE OF PATTERN
JSR PC, GENBUF                          ;GO GENERATE BUFFER FOR FORMAT

;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATIION
MOV #PUTINX, R2                         ;R2 = BYTE ENTRY POSITION
MOVB #RMDC, (R2)+
MOVB #RMDA, (R2)+
MOVB #RMBA, (R2)+
MOVB #RMWC, (R2)+
MOVB #RMOF, (R2)+
MOVB #RMCS1, (R2)+
MOVB #200, (R2)                         ;TERMINATE TABLE

```

```

4708 014412 30$:
4709
4710 ;FORMAT THE DRIVE
4711 014412 004737 044006 JSR PC,PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
4712 014416 000404 BR 40$ ;GO TO 40$ IF NO ERROR
4713 014420 000240 NOP ;RETURN HERE IF ERROR
4714 014422 104000 ERROR ;ERROR # DEFINED BY PUT SUBROUTINE
4715 014424 000137 015172 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
4716 014430
4717
4718 ;SETUP GET REGISTER INDEX TABLE FOR READING STATUS
4719 014430 004737 043452 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
4720
4721 ;WAIT FOR THE FORMAT TO COMPLETE
4722 014434 004737 044346 JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
4723
4724 ;GO READ STATUS FOR FORMAT OPERATION
4725 014440 004737 043536 JSR PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
4726 014444 000404 BR 50$ ;GO TO 50$ IF NO ERROR
4727 014446 000240 NOP ;RETURN HERE IF ERROR
4728 014450 104000 ERROR ;ERROR # DEFINED BY GET SUBROUTINE
4729 014452 000137 015172 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
4730 014456
4731
4732 ;VERIFY NO ERRORS DURING FORMAT
4733 014456 004737 057036 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
4734 014462 000405 BR 60$ ;GO TO 60$ IF NO ERROR
4735 014464 000240 NOP ;RETURN HERE IF ERROR
4736 014466 104000 ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
4737 014470 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
4738 014472 000137 015172 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
4739 014476
4740
4741 ;MOVE LOOP ADDRESSES TO NEXT OPERATION
4742 014476 012737 014514 001122 MOV #70$,SLPADR
4743 014504 012737 014514 001124 MOV #70$,SLPERR
4744 014512 000410 BR 80$ ;SKIP TO WRITE OPERATION
4745
4746 ;*****
4747 ;LOOP #2 WRITE,WRITE CHECK DATA
4748
4749 014514 70$:
4750
4751 ;PREPARE DEVICE FOR WRITE OPERATION
4752
4753 014514 004737 040020 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
4754 014520 054130 WORD 054130 ;TASK DESCRIPTOR
4755 014522 000404 BR 80$ ;GO TO 80$ IF NO ERROR
4756 014524 000240 NOP ;RETURN HERE IF ERROR
4757 014526 104000 ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
4758 014530 000137 015172 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
4759 014534
4760
4761
4762 014534 80$:
4763 120$:

```

```
4764 ;WRITE DATA TO THE DRIVE
4765 014534 012737 107554 001404 MOV #BUFONE+4,RMBA0 ;CHANGE MEMORY ADDRESS
4766 014542 012737 177400 001402 MOV #(<C256,+1>),RMWCO ;CHANGE WORD COUNT
4767 014550 012737 000061 001400 MOV #WD!GO,RMCS10 ;WRITE DATA COMMAND
4768 014556 012702 001535 MOV #PUTINX,R2 ;LOAD PUT REGISTER INDEX TABLE
4769 014562 112722 000006 MOVB #RMDA,(R2)+
4770 014566 112722 000034 MOVB #RMDC,(R2)+
4771 014572 112722 000032 MOVB #RMOF,(R2)+
4772 014576 112722 000004 MOVB #RMBA,(R2)+
4773 014602 112722 000002 MOVB #RMWC,(R2)+
4774 014606 112722 000000 MOVB #RMCS1,(R2)+
4775 014612 112722 000200 MOVB #200,(R2)+ ;TERMINATE TABLE
4776
4777 014616 004737 044006 JSR PC,PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
4778 014622 000404 BR 130$ ;GO TO 130$ IF NO ERROR
4779 014624 000240 NOP ;RETURN HERE IF ERROR
4780 014626 104000 ERROR ;ERROR # DEFINED BY PUT SUBROUTINE
4781 014630 000137 015172 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
4782 014634
4783
4784 ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
4785 014634 004737 043452 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
4786
4787 ;WAIT FOR WRITE COMMAND TO COMPLETE
4788 014640 004737 044346 JSR PC,TIMOUT ;GO TO TIMEOUT SUBROUTINE
4789
4790 ;GO READ STATUS FOR WRITE COMMAND
4791 014644 004737 043536 JSR PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
4792 014650 000404 BR 140$ ;GO TO 140$ IF NO ERROR
4793 014652 000240 NOP ;RETURN HERE IF ERROR
4794 014654 104000 ERROR ;ERROR # DEFINED BY GET SUBROUTINE
4795 014656 000137 015172 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
4796 014662
4797
4798 ;CHECK FOR ERRORS DURING WRITE OPERATION
4799 014662 004737 044532 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
4800 014666 000405 BR 150$ ;GO TO 150$ IF NO ERROR
4801 014670 000240 NOP ;RETURN HERE IF ERROR
4802 014672 104000 ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
4803 014674 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
4804 014676 000137 015172 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
4805 014702
4806 014702 004737 057036 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
4807 014706 000405 BR 160$ ;GO TO 160$ IF NO ERROR
4808 014710 000240 NOP ;RETURN HERE IF ERROR
4809 014712 104000 ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
4810 014714 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
4811 014716 000137 015172 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
4812 014722
4813
4814 ;CHECK FOR SECONDARY ERRORS
4815 014722 004737 045364 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
4816 014726 000405 BR 180$ ;GO TO 180$ IF NO ERROR
4817 014730 000240 NOP ;RETURN HERE IF ERROR
4818 014732 104000 ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
4819 014734 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
```

```

4820 014736 000137 015172          JMP      350$          ;GO TO 350$ IF ERROR WAS FOUND
4821 014742          180$:
4822
4823          ;CHANGE LOOP ADDRESSES
4824 014742 012737 014760 001122    MOV      #190$, $LPADR
4825 014750 012737 014760 001124    MOV      #190$, $LPERR
4826 014756 000410          BR       200$          ;SKIP TO NEXT OPERATION
4827
4828          ;*****
4829          ;LOOP #3          WRITE CHECK DATA
4830
4831 014760          190$:
4832
4833          ;PREPARE DEVICE FOR WRITE CHECK DATA OPERATION
4834 014760 004737 040020    JSR      PC, TSTPRP    ;PREPARE DEVICE FOR TEST
4835 014764 054130          .WORD   054130    ;TASK DESCRIPTOR
4836 014766 000404          BR       200$          ;GO TO 200$ IF NO ERROR
4837 014770 000240          NOP
4838 014772 104000          ERROR   ;RETURN HERE IF ERROR
4839 014774 000137 015172    JMP      350$          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
4840 015000          200$:          ;GO TO 350$ IF ERROR WAS FOUND
4841
4842 015000          240$:
4843
4844          ;WRITE CHECK DATA DATA FROM DEVICE
4845 015000 012737 000051 001400    MOV      #WCD!GO, RMCS10 ;WRITE CHECK DATA DATA COMMAND
4846 015006 012702 001535          MOV      #PUTINX, R2    ;LOAD PUT REGISTER INDEX TABLE
4847 015012 112722 000006          MOV      #RMDA, (R2)+
4848 015016 112722 000032          MOV      #RMOF, (R2)+
4849 015022 112722 000034          MOV      #RMDC, (R2)+
4850 015026 112722 000004          MOV      #RMBA, (R2)+
4851 015032 112722 000002          MOV      #RMWC, (R2)+
4852 015036 112722 000000          MOV      #RMCS1, (R2)+
4853 015042 112712 000200          MOV      #200, (R2)
4854
4855 015046 004737 044006    JSR      PC, PUT      ;GO WRITE REGISTERS WITH PUT SUBROUTINE
4856 015052 000404          BR       250$          ;GO TO 250$ IF NO ERROR
4857 015054 000240          NOP
4858 015056 104000          ERROR   ;RETURN HERE IF ERROR
4859 015060 000137 015172    JMP      350$          ;ERROR # DEFINED BY PUT SUBROUTINE
4860 015064          250$:          ;GO TO 350$ IF ERROR WAS FOUND
4861
4862          ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
4863 015064 004737 043452    JSR      PC, GETSTS    ;GO TO GETSTS SUBROUTINE
4864
4865          ;WAIT FOR WRITE CHECK DATA OPERATION TO COMPLETE
4866 015070 004737 044346    JSR      PC, TIMEOUT   ;GO TO TIMEOUT SUBROUTINE
4867
4868          ;GO READ STATUS FOR WRITE CHECK DATA OPERATION
4869 015074 004737 043536    JSR      PC, GET      ;GO READ REGISTERS WITH GET SUBROUTINE
4870 015100 000404          BR       260$          ;GO TO 260$ IF NO ERROR
4871 015102 000240          NOP
4872 015104 104000          ERROR   ;RETURN HERE IF ERROR
4873 015106 000137 015172    JMP      350$          ;ERROR # DEFINED BY GET SUBROUTINE
4874 015112          260$:          ;GO TO 350$ IF ERROR WAS FOUND
4875

```

JOB

CZRMEBO RM03/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 100
T10 WRITE, WRITE CHECK ONES

SEQ 0100

```

4876                                     ;CHECK FOR ERRORS DURING WRITE CHECK DATA OPERATION
4877 015112 004737 044532                JSR    PC,PRIERR          ;GO CHECK FOR PRIMARY ERRORS
4878 015116 000405                        BR     270$              ;GO TO 270$ IF NO ERROR
4879 015120 000240                        NOP                                ;RETURN HERE IF ERROR
4880 015122 104000                        ERROR                   ;ERROR # DEFINED BY PRIERR SUBROUTINE
4881 015124 004736                        JSR    PC,(SP)+          ;GO BACK FOR MORE ERROR CHECKS
4882 015126 000137 015172                JMP    350$              ;GO TO 350$ IF ERROR WAS FOUND
4883 015132
4884 015132 004737 057036                270$: JSR    PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
4885 015136 000405                        BR     280$              ;GO TO 280$ IF NO ERROR
4886 015140 000240                        NOP                                ;RETURN HERE IF ERROR
4887 015142 104000                        ERROR                   ;ERROR # DEFINED BY DTASTS SUBROUTINE
4888 015144 004736                        JSR    PC,(SP)+          ;GO BACK FOR MORE ERROR CHECKS
4889 015146 000137 015172                JMP    350$              ;GO TO 350$ IF ERROR WAS FOUND
4890 015152
4891
4892
4893 015152 004737 045364                280$: JSR    PC,SECERR      ;GO CHECK FOR SECONDARY ERRORS
4894 015156 000405                        BR     300$              ;GO TO 300$ IF NO ERROR
4895 015160 000240                        NOP                                ;RETURN HERE IF ERROR
4896 015162 104000                        ERROR                   ;ERROR # DEFINED BY SECERR SUBROUTINE
4897 015164 004736                        JSR    PC,(SP)+          ;GO BACK FOR MORE ERROR CHECKS
4898 015166 000137 015172                JMP    350$              ;GO TO 350$ IF ERROR WAS FOUND
4899 015172
4900
4901 015172
4902
4903
4904
4905 015172
4906 015172 000004                        300$:
4907 015174 000240                        ;*****
4908 015176 012706 001100                ;TEST 11      WRITE, WRITE CHECK ONES W/ WCE ERROR
4909 015202 013700 001276                ;*****
4910 015206 013701 001450                ;ST11:
4911 015212 012737 000011 001226        SCOPE                   ;SCOPE CALL
4912
4913
4914
4915
4916 015220
4917
4918
4919 015220 004737 040020                ;PREPARE THE DEVICE FOR FORMAT OPERATION
4920 015224 054130                        JSR    PC,TSTPRP        ;PREPARE DEVICE FOR TEST
4921 015226 000404                        .WORD 054130 ;TASK DESCRIPTOR
4922 015230 000240                        BR     20$              ;GO TO 20$ IF NO ERROR
4923 015232 104000                        NOP                                ;RETURN HERE IF ERROR
4924 015234 000137 016412                ERROR                   ;ERROR # DEFINED BY TSTPRP SUBROUTINE
4925 015240                        JMP    350$              ;GO TO 350$ IF ERROR WAS FOUND
4926
4927
4928 015240 012737 000000 001434        ;LOAD PARAMETERS AND GENERATE DATA BUFFER
4929 015246 012737 000000 001406        MOV    #0,RMDCO         ;CYLINDER = 0
4930 015254 012737 107550 001404        MOV    #0,RMDAO         ;TRACK = 0, SECTOR = 0
4931 015262 012737 177376 001402        MOV    #BUFONE,RMBAO    ;BUS ADDRESS
                                MOV    #(<C(2+256.)+1>),RMWCO ;WORD COUNT = 1 SECTOR

```


K08

CZRMEBO RMO3/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 101
T11 WRITE, WRITE CHECK ONES W/ WCE ERROR

SEQ 0101

```

4932 015270 012737 010000 001432      MOV      #FMT16,RMOFO      ;16 BIT FORMAT
4933 015276 012737 000063 001400      MOV      #WH!GO,RMCS10    ;WRITE HEADER AND DATA COMMAND
4934
4935      ;VERIFY SECTOR(S) NOT IN BAD SECTOR FILE
4936
4937 015304 004737 040734      JSR      PC,BADSCT        ;CALL BAD SECTOR MODULE
4938 015310 000405                BR      25$              ;GO TO 25$ IF NO ERROR
4939 015312 104401 067466      TYPE    ,SCTMSG          ;TYPE BAD SECTOR MESSAGE
4940 015316 104000                ERROR   #DEFINED BY BADSCT SUBROUTINE
4941 015320 000137 016412      JMP      350$            ;GO TO 350$ IF ERROR WAS FOUND
4942 015324
4943 015324 012737 071046 001174 25$:      MOV      #ONES,$TMP0     ;STARTING ADDRESS OF PATTERN
4944 015332 012737 000001 001176      MOV      #1,$TMP1        ;RANGE OF PATTERN
4945 015340 004737 042640      JSR      PC,GENBUF        ;GO GENERATE BUFFER FOR FORMAT
4946
4947      ;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATION
4948 015344 012702 001535      MOV      #PUTINX,R2      ;R2 = BYTE ENTRY POSITION
4949 015350 112722 000034      MOVB    #RMDC,(R2)+
4950 015354 112722 000006      MOVB    #RMDA,(R2)+
4951 015360 112722 000004      MOVB    #RMBA,(R2)+
4952 015364 112722 000002      MOVB    #RMWC,(R2)+
4953 015370 112722 000032      MOVB    #RMOF,(R2)+
4954 015374 112722 000000      MOVB    #RMCS1,(R2)+
4955 015400 112712 000200      MOVB    #200,(R2)        ;TERMINATE TABLE
4956 015404
4957
4958      ;FORMAT THE DRIVE
4959 015404 004737 044006      JSR      PC,PUT          ;GO WRITE REGISTERS WITH PUT SUBROUTINE
4960 015410 000404                BR      40$              ;GO TO 40$ IF NO ERROR
4961 015412 000240                NOP
4962 015414 104000                ERROR   #DEFINED BY PUT SUBROUTINE
4963 015416 000137 016412      JMP      350$            ;GO TO 350$ IF ERROR WAS FOUND
4964 015422
4965
4966      ;SETUP GET REGISTER INDEX TABLE FOR READING STATUS
4967 015422 004737 043452      JSR      PC,GETSTS       ;GO TO GETSTS SUBROUTINE
4968
4969      ;WAIT FOR THE FORMAT TO COMPLETE
4970 015426 004737 044346      JSR      PC,TIMOUT       ;GO TO TIMOUT SUBROUTINE
4971
4972      ;GO READ STATUS FOR FORMAT OPERATION
4973 015432 004737 043536      JSR      PC,GET          ;GO READ REGISTERS WITH GET SUBROUTINE
4974 015436 000404                BR      50$              ;GO TO 50$ IF NO ERROR
4975 015440 000240                NOP
4976 015442 104000                ERROR   #DEFINED BY GET SUBROUTINE
4977 015444 000137 016412      JMP      350$            ;GO TO 350$ IF ERROR WAS FOUND
4978 015450
4979
4980      ;VERIFY NO ERRORS DURING FORMAT
4981 015450 004737 057036      JSR      PC,DTASTS       ;GO VERIFY RESULTS OF DATA TRANSFER
4982 015454 000405                BR      60$              ;GO TO 60$ IF NO ERROR
4983 015456 000240                NOP
4984 015460 104000                ERROR   #DEFINED BY DTASTS SUBROUTINE
4985 015462 004736      JSR      PC,@(SP)+       ;GO BACK FOR MORE ERROR CHECKS
4986 015464 000137 016412      JMP      350$            ;GO TO 350$ IF ERROR WAS FOUND
4987 015470
60$:

```

L08

CZRMEBO RM03/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 102
T11 WRITE, WRITE CHECK ONES W/ WCE ERROR

SEQ 0102

```

4988
4989
4990 015470 012737 015506 001122 ;MOVE LOOP ADDRESSES TO NEXT OPERATION
4991 015476 012737 015506 001124 MOV #70$, $LPADR
4992 015504 000410 BR 80$ ;SKIP TO WRITE OPERATION
4993
4994 ;*****
4995 ;LOOP #2 WRITE,WRITE CHECK DATA
4996
4997 015506 70$:
4998
4999
5000 ;PREPARE DEVICE FOR WRITE OPERATION
5001 015506 004737 040020 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
5002 015512 054130 .WORD 054130 ;TASK DESCRIPTOR
5003 015514 000404 BR 80$ ;GO TO 80$ IF NO ERROR
5004 015516 000240 NOP ;RETURN HERE IF ERROR
5005 015520 104000 ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
5006 015522 000137 016412 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
5007 015526 80$:
5008
5009
5010 015526 120$:
5011
5012 ;WRITE DATA TO THE DRIVE
5013 015526 012737 177400 001402 MOV #(<C256.+1),RMWCC ;CHANGE WORD COUNT
5014 015534 012737 107554 001404 MOV #BUFONE+4,RMBA0 ;CHANGE MEMORY ADDRESS
5015 015542 012737 000061 001400 MOV #WD!GO,RMCS10 ;WRITE DATA COMMAND
5016 015550 012702 001535 MOV #PUTINX,R2 ;LOAD PUT REGISTER INDEX TABLE
5017 015554 112722 000006 MOVB #RMDA,(R2)+
5018 015560 112722 000034 MOVB #RMDC,(R2)+
5019 015564 112722 000032 MOVB #RMOF,(R2)+
5020 015570 112722 000004 MOVB #RMBA,(R2)+
5021 015574 112722 000002 MOVB #RMWC,(R2)+
5022 015600 112722 000000 MOVB #RMCS1,(R2)+
5023 015604 112722 000200 MOVB #200,(R2)+ ;TERMINATE TABLE
5024
5025 015610 004737 044006 JSR PC,PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
5026 015614 000404 BR 130$ ;GO TO 130$ IF NO ERROR
5027 015616 000240 NOP ;RETURN HERE IF ERROR
5028 015620 104000 ERROR ;ERROR # DEFINED BY PUT SUBROUTINE
5029 015622 000137 016412 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
5030 015626 130$:
5031
5032 ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
5033 015626 004737 043452 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
5034
5035 ;WAIT FOR WRITE COMMAND TO COMPLETE
5036 015632 004737 044346 JSR PC,TIMOUT ;GO TO TIMEOUT SUBROUTINE
5037
5038 ;GO READ STATUS FOR WRITE COMMAND
5039 015636 004737 043536 JSR PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
5040 015642 000404 BR 140$ ;GO TO 140$ IF NO ERROR
5041 015644 000240 NOP ;RETURN HERE IF ERROR
5042 015646 104000 ERROR ;ERROR # DEFINED BY GET SUBROUTINE
5043 015650 000137 016412 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND

```

M08

CZRMEBO RM03/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 103
T11 WRITE, WRITE CHECK ONES W/ WCE ERROR

SEQ 0103

```

5044 015654          140$:
5045
5046          ;CHECK FOR ERRORS DURING WRITE OPERATION
5047 015654 004737 044532      JSR    PC, PRIERR      ;GO CHECK FOR PRIMARY ERRORS
5048 015660 000405          BR     150$           ;GO TO 150$ IF NO ERROR
5049 015662 000240          NOP                    ;RETURN HERE IF ERROR
5050 015664 104000          ERROR                ;ERROR # DEFINED BY PRIERR SUBROUTINE
5051 015666 004736          JSR    PC, 2(SP)+     ;GO BACK FOR MORE ERROR CHECKS
5052 015670 000137 016412      JMP    350$           ;GO TO 350$ IF ERROR WAS FOUND
5053
5054          150$:
5054 015674 004737 057036      JSR    PC, DTASTS     ;GO VERIFY RESULTS OF DATA TRANSFER
5055 015700 000405          BR     160$           ;GO TO 160$ IF NO ERROR
5056 015702 000240          NOP                    ;RETURN HERE IF ERROR
5057 015704 104000          ERROR                ;ERROR # DEFINED BY DTASTS SUBROUTINE
5058 015706 004736          JSR    PC, 2(SP)+     ;GO BACK FOR MORE ERROR CHECKS
5059 015710 000137 016412      JMP    350$           ;GO TO 350$ IF ERROR WAS FOUND
5060
5061          160$:
5062
5063          170$:
5063 015714 004737 045364      JSR    PC, SECERR     ;GO CHECK FOR SECONDARY ERRORS
5064 015720 000405          BR     180$           ;GO TO 180$ IF NO ERROR
5065 015722 000240          NOP                    ;RETURN HERE IF ERROR
5066 015724 104000          ERROR                ;ERROR # DEFINED BY SECERR SUBROUTINE
5067 015726 004736          JSR    PC, 2(SP)+     ;GO BACK FOR MORE ERROR CHECKS
5068 015730 000137 016412      JMP    350$           ;GO TO 350$ IF ERROR WAS FOUND
5069
5070          180$:
5071          ;CHANGE LOOP ADDRESSES
5072 015734 012737 015756 001122  MOV    #190$, $LPADR
5073 015742 012737 015756 001124  MOV    #190$, $LPERR
5074 015750 012737 015762 001210  MOV    #195$, $ESCAPE ;;ESCAPE TO 195$ ON ERROR
5075
5076          ;*****
5077          ;LOOP #3      WRITE CHECK DATA
5078
5079 015756          190$:
5080
5081 015756 012703 000001      MOV    #1, R3         ;R3+WCE BIT POSITION
5082 015762 040337 110552      BIC    R3, BUFTWO-2   ;CHANGE LAST WORD OF BUFFER
5083
5084          ;PREPARE DEVICE FOR WRITE CHECK DATA OPERATION
5085 015766 004737 040020      JSR    PC, TSTPRP     ;PREPARE DEVICE FOR TEST
5086 015772 054130          .WORD 054130 ;TASK DESCRIPTOR
5087 015774 000404          BR     200$           ;GO TO 200$ IF NO ERROR
5088 015776 000240          NOP                    ;RETURN HERE IF ERROR
5089 016000 104000          ERROR                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
5090 016002 000137 016412      JMP    350$           ;GO TO 350$ IF ERROR WAS FOUND
5091
5092          200$:
5093 016006          240$:
5094
5095          ;READ DATA FROM DEVICE
5096 016006 012737 000051 001400  MOV    #WCD!GO, RMCS10 ;WRITE CHECK DATA DATA COMMAND
5097 016014 012702 001535          MOV    #PUTINX, R2    ;LOAD PUT REGISTER INDEX TABLE
5098 016020 112722 000006          MOVB  #RMDA, (R2)+
5099 016024 112722 000032          MOVB  #RMOF, (R2)+

```



```

5156 016252 004737 043536 JSR PC,GET ;GO GET THE CONTENTS OF THE DATA BUFFER
5157 016256 000402 BP 290$ ;GO TO 290$ IF NO ERROR
5158 016260 000240 NOP ;RETURN HERE IF ERROR
5159 016262 104000 ERROR ;ERROR DEFINED BY GET SUBROUTINE
5160
5161 016264 290$:
5162 016264 013737 001352 001142 MOV RMDBI,$BDDAT ;RECEIVED DATA
5163 016272 013737 110552 001140 MOV BUFTWO-2,$GDDAT ;EXPECTED DATA
5164 016300 050337 001140 BIS R3,$GDDAT
5165 016304 012737 110552 001134 MOV #BUFTWO-2,$GDADR ;EXPECTED ADDRESS
5166 016312 013737 001334 001136 MOV RMBAI,$BDAADR ;RECEIVED ADDRESS
5167 016320 162737 000002 001136 SUB #2,$BDAOR ;CORRECT MEMORY ADDRESS
5168 016326 023737 001134 001136 CMP $GDAOR,$BDAOR ;ADDRESSES OK??
5169 016334 001402 BEQ 295$ ;YES!!
5170 016336 104340 ERROR 340 ;WCE AT UNEXPECTED ADDRESS
5171 016340 000424 BR 350$
5172 016342 012737 001140 001142 295$: CMP $GDDAT,$BDDAT ;DATA OK??
5173 016350 001402 BEQ 296$ ;YES!!
5174 016352 104341 ERROR 341 ;UNEXPECTED WCE DATA
5175 016354 000416 BR 350$
5176 016356 296$:
5177
5178 016356 004737 045364 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
5179 016362 000405 BR 300$ ;GO TO 300$ IF NO ERROR
5180 016364 000240 NOP ;RETURN HERE IF ERROR
5181 016366 104000 ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
5182 016370 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
5183 016372 000137 016412 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
5184 016376 300$:
5185
5186 016376 050337 110552 BIS R3,BUFTWO-2 ;RESTORE DATA PATTERN
5187 016402 006303 ASL R3 ;SHIFT TO NEXT BIT
5188 016404 001402 BEQ 350$ ;EXIT IF DONE
5189 016406 000137 015762 JMP 195$ ;REPEAT TEST FOR NEXT DATA BIT
5190 016412 350$:
5191 016412 012737 000000 001210 MOV #0,$ESCAPE ;;ESCAPE TO 0 ON ERROR
5192 016420 012737 015220 001122 MOV #10,$SLPADR ;CHANGE LOOP TO START OF TEST
5193 016426 012737 015220 001124 MOV #10,$SLPERR
5194 ;*****
5195 ;*TEST 12 WRITE, WRITE CHECK MULTIPLE SECTORS
5196 ;*****
5197 ;*ST12:
5198 016434 000004 SCOPE ;SCOPE CALL
5199 016436 000240 NOP ;START OF TEST
5200 016440 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
5201 016444 013700 001276 MOV $BASE,R0 ;R0=UNIBUS ADDRESS
5202 016450 013701 001450 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
5203 016454 012737 000012 001226 MOV #12,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
5204
5205 ;*****
5206 ;LOOP #1 FORMAT,WRITE,READ
5207
5208 016462 10$:
5209
5210 ;PREPARE THE DEVICE FOR FORMAT OPERATION
5211 016462 004737 040020 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST

```

```

5212 016466 054130 .WORD 054130 ;TASK DESCRIPTOR
5213 016470 000404 BR 20$ ;GO TO 20$ IF NO ERROR
5214 016472 000240 NOP ;RETURN HERE IF ERROR
5215 016474 104000 ERROR ;ERROR # DEFINED BY TSTPPP SUBROUTINE
5216 016476 000137 017454 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
5217 016502 20$:
5218
5219 ;LOAD PARAMETERS AND GENERATE DATA BUFFER
5220 016502 012737 000000 001434 MOV #0,RMDCO ;CYLINDER = 0
5221 016510 012737 000000 001406 MOV #0,RMDAO ;TRACK = 0, SECTOR = 0
5222 016516 012737 107550 001404 MOV #BUFONE,RMBAO ;BUS ADDRESS
5223 016524 012737 177240 001402 MOV #(<C<2*12+256>>+1),RMWCO ;WORD COUNT
5224 016532 012737 010000 001432 MOV #FMT16,RMOFO ;16 BIT FORMAT
5225 016540 012737 000063 001400 MOV #WH!GO,RMCS10 ;WRITE HEADER AND DATA COMMAND
5226
5227 ;VERIFY SECTOR(S) NOT IN BAD SECTOR FILE
5228
5229 016546 004737 040734 JSR PC,BADSCT ;CALL BAD SECTOR MODULE
5230 016552 000405 BR 25$ ;GO TO 25$ IF NO ERROR
5231 016554 104401 067466 TYPE ,SCTMSG ;TYPE BAD SECTOR MESSAGE
5232 016560 104000 ERROR ;ERROR # DEFINED BY BADSCT SUBROUTINE
5233 016562 000137 017454 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
5234 016566 25$:
5235 016566 012737 071110 001174 MOV #ZEROS,$TMP0 ;STARTING ADDRESS OF PATTERN
5236 016574 012737 000001 001176 MOV #1,$TMP1 ;RANGE OF PATTERN
5237 016602 004737 042640 JSR PC,GENBUF ;GO GENERATE BUFFER FOR FORMAT
5238
5239 ;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATION
5240 016606 012702 001535 MOV #PUTINX,R2 ;R2 = BYTE ENTRY POSITION
5241 016612 112722 000034 MOVB #RMDC,(R2)+
5242 016616 112722 000006 MOVB #RMDA,(R2)+
5243 016622 112722 000004 MOVB #RMBA,(R2)+
5244 016626 112722 000002 MOVB #RMWC,(R2)+
5245 016632 112722 000032 MOVB #RMOF,(R2)+
5246 016636 112722 000000 MOVB #RMCS1,(R2)+
5247 016642 112712 000200 MOVB #200,(R2) ;TERMINATE TABLE
5248 016646 30$:
5249
5250 ;FORMAT THE DRIVE
5251 016646 004737 044006 JSR PC,PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
5252 016652 000404 BR 40$ ;GO TO 40$ IF NO ERROR
5253 016654 000240 NOP ;RETURN HERE IF ERROR
5254 016656 104000 ERROR ;ERROR # DEFINED BY PUT SUBROUTINE
5255 016660 000137 017454 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
5256 016664 40$:
5257
5258 ;SETUP GET REGISTER INDEX TABLE FOR READING STATUS
5259 016664 004737 043452 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
5260
5261 ;WAIT FOR THE FORMAT TO COMPLETE
5262 016670 004737 044346 JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
5263
5264 ;GO READ STATUS FOR FORMAT OPERATION
5265 01667 . 004737 043536 JSR PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
5266 016700 000404 BR 50$ ;GO TO 50$ IF NO ERROR
5267 016702 000240 NOP ;RETURN HERE IF ERROR

```

```

5268 016704 104000          ERROR          ;ERROR # DEFINED BY GET SUBROUTINE
5269 016706 000137 017454  JMP          350$          ;GO TO 350$ IF ERROR WAS FOUND
5270 016712          50$:
5271          ;VERIFY NO ERRORS DURING FORMAT
5272          JSR          PC,DTASTS          ;GO VERIFY RESULTS OF DATA TRANSFER
5273 016712 004737 057036  BR          60$          ;GO TO 60$ IF NO ERROR
5274 016716 000405          ;RETURN HERE IF ERROR
5275 016720 000240          ;ERROR # DEFINED BY DTASTS SUBROUTINE
5276 016722 104000          ;GO BACK FOR MORE ERROR CHECKS
5277 016724 004736          ;GO TO 350$ IF ERROR WAS FOUND
5278 016726 000137 017454  JMP          350$
5279 016732          60$:
5280          ;MOVE LOOP ADDRESSES TO NEXT OPERATION
5281          MOV          #70$, $LPADR
5282 016732 012737 017012 001122  MOV          #70$, $LPERR
5283 016740 012737 017012 001124  ;REGENERATE DATA BUFFER
5284          MOV          <↑C<2*256.>+1>, RMWCO          ;CHANGE WORD COUNT
5285 016746 012737 177000 001402  MOV          #BUFONE, RMBAO          ;CHANGE BUS ADDRESS
5286 016754 012737 107550 001404  MOV          #WD, RMCS10          ;WRITE DATA
5287 016762 012737 000060 001400  MOV          #ONES, $TMP0          ;STARTING ADDRESS OF PATTERN
5288 016770 012737 071046 001174  MOV          #1, $TMP1          ;RANGE OF PATTERN
5289 016776 012737 000001 001176  JSR          PC, GENBUF
5290 017004 004737 042640  BR          80$          ;SKIP TO WRITE OPERATION
5291 017010 000410
5292          ;*****
5293          ;LOOP #2          WRITE, READ
5294
5295
5296 017012          70$:
5297
5298
5299          ;PREPARE DEVICE FOR WRITE OPERATION
5300 017012 004737 040020  JSR          PC, TSTPRP          ;PREPARE DEVICE FOR TEST
5301 017016 054130          .WORD          054130          ;TASK DESCRIPTOR
5302 017020 000404          BR          80$          ;GO TO 80$ IF NO ERROR
5303 017022 000240          ;RETURN HERE IF ERROR
5304 017024 104000          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
5305 017026 000137 017454  JMP          350$          ;GO TO 350$ IF ERROR WAS FOUND
5306 017032          80$:
5307
5308
5309 017032          120$:
5310
5311          ;WRITE DATA TO THE DRIVE
5312 017032 012737 000061 001400  MOV          #WD!GO, RMCS10          ;WRITE DATA COMMAND
5313 017040 012702 001535          MOV          #PUTINX, R2          ;LOAD PUT REGISTER INDEX TABLE
5314 017044 112722 000006          MOV          #RMDA, (R2)+
5315 017050 112722 000034          MOV          #RMDC, (R2)+
5316 017054 112722 000032          MOV          #RMOF, (R2)+
5317 017060 112722 000004          MOV          #RMB A, (R2)+
5318 017064 112722 000002          MOV          #RMWC, (R2)+
5319 017070 112722 000000          MOV          #RMCS1, (R2)+
5320 017074 112722 000200          MOV          #200, (R2)+          ;TERMINATE TABLE
5321
5322 017100 004737 044006          JSR          PC, PUT          ;GO WRITE REGISTERS WITH PUT SUBROUTINE
5323 017104 000404          BR          130$          ;GO TO 130$ IF NO ERROR

```

```

5324 017106 000240      NOP      ;RETURN HERE IF ERROR
5325 017110 104000      ERROR    ;ERROR # DEFINED BY PUT SUBROUTINE
5326 017112 000137 017454  JMP      350$      ;GO TO 350$ IF ERROR WAS FOUND
5327 017116      130$:
5328
5329      ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
5330 017116 004737 043452  JSR      PC,GETSTS ;GO TO GETSTS SUBROUTINE
5331
5332      ;WAIT FOR WRITE COMMAND TO COMPLETE
5333 017122 004737 044346  JSR      PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
5334
5335      ;GO READ STATUS FOR WRITE COMMAND
5336 017126 004737 043536  JSR      PC,GET    ;GO READ REGISTERS WITH GET SUBROUTINE
5337 017132 000404      BR      140$      ;GO TO 140$ IF NO ERROR
5338 017134 000240      NOP
5339 017136 104000      ERROR    ;RETURN HERE IF ERROR
5340 017140 000137 017454  JMP      350$      ;ERROR # DEFINED BY GET SUBROUTINE
5341 017144      140$:      ;GO TO 350$ IF ERROR WAS FOUND
5342
5343      ;CHECK FOR ERRORS DURING WRITE OPERATION
5344 017144 004737 044532  JSR      PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
5345 017150 000405      BR      150$      ;GO TO 150$ IF NO ERROR
5346 017152 000240      NOP
5347 017154 104000      ERROR    ;RETURN HERE IF ERROR
5348 017156 004736      JSR      PC,@(SP)+ ;ERROR # DEFINED BY PRIERR SUBROUTINE
5349 017160 000137 017454  JMP      350$      ;GO BACK FOR MORE ERROR CHECKS
5350 017164      150$:      ;GO TO 350$ IF ERROR WAS FOUND
5351 017164 004737 057036  JSR      PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
5352 017170 000405      BR      160$      ;GO TO 160$ IF NO ERROR
5353 017172 000240      NOP
5354 017174 104000      ERROR    ;RETURN HERE IF ERROR
5355 017176 004736      JSR      PC,@(SP)+ ;ERROR # DEFINED BY DTASTS SUBROUTINE
5356 017200 000137 017454  JMP      350$      ;GO BACK FOR MORE ERROR CHECKS
5357 017204      160$:      ;GO TO 350$ IF ERROR WAS FOUND
5358
5359      170$:
5360 017204 004737 045364  JSR      PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
5361 017210 000405      BR      180$      ;GO TO 180$ IF NO ERROR
5362 017212 000240      NOP
5363 017214 104000      ERROR    ;RETURN HERE IF ERROR
5364 017216 004736      JSR      PC,@(SP)+ ;ERROR # DEFINED BY SECERR SUBROUTINE
5365 017220 000137 017454  JMP      350$      ;GO BACK FOR MORE ERROR CHECKS
5366 017224      180$:      ;GO TO 350$ IF ERROR WAS FOUND
5367
5368      ;CHANGE LOOP ADDRESSES
5369 017224 012737 017242 001122  MOV      #190$,SLPADR
5370 017232 012737 017242 001124  MOV      #190$,SLPERR
5371 017240 000410      BR      200$      ;SKIP TO NEXT OPERATION
5372
5373      ;*****
5374      ;LOOP #3      READ
5375
5376 017242      190$:
5377
5378      ;PREPARE DEVICE FOR READ OPERATION
5379 017242 004737 040020  JSR      PC,TSTPRP ;PREPARE DEVICE FOR TEST

```



```

5380 017246 054130 .WORD 054130 ;TASK DESCRIPTOR
5381 017250 000404 BR 200$ ;GO TO 200$ IF NO ERROR
5382 017252 000240 NOP ;RETURN HERE IF ERROR
5383 017254 104000 ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
5384 017256 000137 017454 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
5385 017262 200$:
5386
5387 017262 240$:
5388
5389 ;READ DATA FROM DEVICE
5390 017262 012737 000051 001400 MOV #WCD!GO, RMCS10 ;READ DATA COMMAND
5391 017270 012702 001535 MOV #PUTINX, R2 ;LOAD PUT REGISTER INDEX TABLE
5392 017274 112722 000006 MOVB #RMDA, (R2)+
5393 017300 112722 000032 MOVB #RMOF, (R2)+
5394 017304 112722 000034 MOVB #RMDC, (R2)+
5395 017310 112722 000004 MOVB #RMBA, (R2)+
5396 017314 112722 000002 MOVB #RMWC, (R2)+
5397 017320 112722 000000 MOVB #RMCS1, (R2)+
5398 017324 112712 000200 MOVB #200, (R2)
5399
5400 017330 004737 044006 JSR PC, PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
5401 017334 000404 BR 250$ ;GO TO 250$ IF NO ERROR
5402 017336 000240 NOP ;RETURN HERE IF ERROR
5403 017340 104000 ERROR ;ERROR # DEFINED BY PUT SUBROUTINE
5404 017342 000137 017454 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
5405 017346 250$:
5406
5407 ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
5408 017346 004737 043452 JSR PC, GETSTS ;GO TO GETSTS SUBROUTINE
5409
5410 ;WAIT FOR READ OPERATION TO COMPLETE
5411 017352 004737 044346 JSR PC, TIMEOUT ;GO TO TIMEOUT SUBROUTINE
5412
5413 ;GO READ STATUS FOR READ OPERATION
5414 017356 004737 043536 JSR PC, GET ;GO READ REGISTERS WITH GET SUBROUTINE
5415 017362 000404 BR 260$ ;GO TO 260$ IF NO ERROR
5416 017364 000240 NOP ;RETURN HERE IF ERROR
5417 017366 104000 ERROR ;ERROR # DEFINED BY GET SUBROUTINE
5418 017370 000137 017454 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
5419 017374 260$:
5420
5421 ;CHECK FOR ERRORS DURING READ OPERATION
5422 017374 004737 044532 JSR PC, PRIERR ;GO CHECK FOR PRIMARY ERRORS
5423 017400 000405 BR 270$ ;GO TO 270$ IF NO ERROR
5424 017402 000240 NOP ;RETURN HERE IF ERROR
5425 017404 104000 ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
5426 017406 004736 JSR PC, @ (SP)+ ;GO BACK FOR MORE ERROR CHECKS
5427 017410 000137 017454 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
5428 017414 270$:
5429 017414 004737 057036 JSR PC, DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
5430 017420 000405 BR 280$ ;GO TO 280$ IF NO ERROR
5431 017422 000240 NOP ;RETURN HERE IF ERROR
5432 017424 104000 ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
5433 017426 004736 JSR PC, @ (SP)+ ;GO BACK FOR MORE ERROR CHECKS
5434 017430 000137 017454 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
5435 017434 280$:

```

```

5436
5437 017434
5438 017434 004737 045364
5439 017440 000405
5440 017442 000240
5441 017444 104000
5442 017446 004736
5443 017450 000137 017454
5444 017454
5445
5446 017454
5447
5448
5449
5450 017454
5451 017454 000004
5452 017456 000240
5453 017460 012706 001100
5454 017464 013700 001276
5455 017470 013701 001450
5456 017474 012737 000013 001226
5457
5458
5459
5460 017502
5461
5462
5463
5464 017502 004737 040020
5465 017506 054130
5466 017510 000404
5467 017512 000240
5468 017514 104000
5469 017516 000137 020776
5470 017522
5471
5472
5473 017522 012737 000000 001434
5474 017530 012737 000000 001406
5475 017536 012737 107550 001404
5476 017544 012737 177376 001402
5477 017552 012737 010000 001432
5478 017560 012737 000063 001400
5479
5480
5481
5482 017566 004737 040734
5483 017572 000405
5484 017574 104401 067466
5485 017600 104000
5486 017602 000137 020776
5487 017606
5488 017606 012737 071046 001174
5489 017614 012737 000001 001176
5490 017622 004737 042640
5491

```

```

290$:
JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
BR 300$ ;GO TO 300$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND

300$:
350$:
;*****
;TEST 13 WRITE, READ W/ IMPLIED SEEK
;*****
TST13:
MOV #0,RMDCO ;CYLINDER = 0
MOV #0,RMDAO ;TRACK = 0, SECTOR = 0
MOV #BUFONE,RMBAO ;BUS ADDRESS
MOV #(<C<2+256.>+1),RMWCO ;WORD COUNT
MOV #FMT16,RMOFO ;16 BIT FORMAT
MOV #WH!GO,RMCS10 ;WRITE HEADER AND DATA COMMAND

;*****
;LOOP #1 FORMAT,WRITE,READ

10$:
;PREPARE THE DEVICE FOR FORMAT OPERATION
JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
WORD 054130 ;TASK DESCRIPTOR
BR 20$ ;GO TO 20$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND

20$:
;LOAD PARAMETERS AND GENERATE DATA BUFFER
MOV #0,RMDCO ;CYLINDER = 0
MOV #0,RMDAO ;TRACK = 0, SECTOR = 0
MOV #BUFONE,RMBAO ;BUS ADDRESS
MOV #(<C<2+256.>+1),RMWCO ;WORD COUNT
MOV #FMT16,RMOFO ;16 BIT FORMAT
MOV #WH!GO,RMCS10 ;WRITE HEADER AND DATA COMMAND

;VERIFY SECTOR(S) NOT IN BAD SECTOR FILE
JSR PC,BADSCT ;CALL BAD SECTOR MODULE
BR 25$ ;GO TO 25$ IF NO ERROR
TYPE ,SCTMSG ;TYPE BAD SECTOR MESSAGE
ERROR ;ERROR # DEFINED BY BADSCT SUBROUTINE
JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND

25$:
MOV #ONES,$TMP0 ;STARTING ADDRESS OF PATTERN
MOV #1,$TMP1 ;RANGE OF PATTERN
JSR PC,GENBUF ;GO GENERATE BUFFER FOR FORMAT

```

```

5492 ;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATIION
5493      MOV      #PUTINX,R2          ;R2 = BYTE ENTRY POSITION
5494      MOVB     #RMDC,(R2)+
5495      MOVB     #RMDA,(R2)+
5496      MOVB     #RMBA,(R2)+
5497      MOVB     #RMWC,(R2)+
5498      MOVB     #RMOF,(R2)+
5499      MOVB     #RMCS1,(R2)+
5500      MOVB     #200,(R2)          ;TERMINATE TABLE
5501
5502
5503
5504 ;FORMAT THE DRIVE
5505      JSR      PC,PUT              ;GO WRITE REGISTERS WITH PUT SUBROUTINE
5506      BR       40$                ;GO TO 40$ IF NO ERROR
5507      NOP
5508      ERROR    ;RETURN HERE IF ERROR
5509      JMP      350$               ;ERROR # DEFINED BY PUT SUBROUTINE
5510                                     ;GO TO 350$ IF ERROR WAS FOUND
5511
5512 ;SETUP GET REGISTER INDEX TABLE FOR READING STATUS
5513      JSR      PC,GETSTS          ;GO TO GETSTS SUBROUTINE
5514
5515 ;WAIT FOR THE FORMAT TO COMPLETE
5516      JSR      PC,TIMOUT         ;GO TO TIMEOUT SUBROUTINE
5517
5518 ;GO READ STATUS FOR FORMAT OPERATION
5519      JSR      PC,GET            ;GO READ REGISTERS WITH GET SUBROUTINE
5520      BR       50$                ;GO TO 50$ IF NO ERROR
5521      NOP
5522      ERROR    ;RETURN HERE IF ERROR
5523      JMP      350$               ;ERROR # DEFINED BY GET SUBROUTINE
5524                                     ;GO TO 350$ IF ERROR WAS FOUND
5525
5526 ;VERIFY NO ERRORS DURING FORMAT
5527      JSR      PC,DTASTS        ;GO VERIFY RESULTS OF DATA TRANSFER
5528      BR       60$                ;GO TO 60$ IF NO ERROR
5529      NOP
5530      ERROR    ;RETURN HERE IF ERROR
5531      JSR      PC,@(SP)+        ;ERROR # DEFINED BY DTASTS SUBROUTINE
5532      JMP      350$               ;GO BACK FOR MORE ERROR CHECKS
5533                                     ;GO TO 350$ IF ERROR WAS FOUND
5534
5535 ;MOVE LOOP ADDRESSES TO NEXT OPERATION
5536      MOV      #70$, $LPADR
5537      MOV      #70$, $LPERR
5538      MOV      RMDCO,360$
5539      BR       #65$, $ESCAPE    ;;ESCAPE TO 65$ ON ERROR
5540                                     ;SKIP TO WRITE OPERATION
5541
5542 65$:   MOV      360$,RMDCO      ;RESTORE CYLINDER
5543      ;*****
5544      ;LOOP #2      WRITE,READ
5545
5546 70$:
5547

```

```

5548 ;PREPARE DEVICE FOR WRITE OPERATION
5549 020012 004737 040020 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
5550 020016 054130 .WORD 054130 ;TASK DESCRIPTOR
5551 020020 000404 BR 80$ ;GO TO 80$ IF NO ERROR
5552 020022 000240 NOP ;RETURN HERE IF ERROR
5553 020024 104000 ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
5554 020026 000137 020776 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
5555 020032
5556 80$:
5557 020032 013737 001434 021000 MOV RMDC0,360$ ;SAVE CYLINER
5558 020040 012737 001466 001434 MOV #822,RMDC0 ;SEEK TO LAST CYLINER
5559 020046 012737 000005 001400 MOV #SEEK!GO,RMCS10 ;WRITE SEEK COMMAND
5560 020054 004737 044006 JSR PC,PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
5561 020060 000404 BR 90$ ;GO TO 90$ IF NO ERROR
5562 020062 000240 NOP ;RETURN HERE IF ERROR
5563 020064 104000 ERROR ;ERROR # DEFINED BY PUT SUBROUTINE
5564 020066 000137 020776 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
5565 020072
5566 90$:
5567 ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
5568 020072 004737 043452 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
5569 020076 004737 044346 JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
5570 020102 004737 043536 JSR PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
5571 020106 000404 BR 100$ ;GO TO 100$ IF NO ERROR
5572 020110 000240 NOP ;RETURN HERE IF ERROR
5573 020112 104000 ERROR ;ERROR # DEFINED BY GET SUBROUTINE
5574 020114 000137 020776 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
5575 020120
5576 020120 004737 051440 JSR PC,SEKSTS ;GO VERIFY RESULTS OF SEEK OPERATION
5577 020124 000405 BR 110$ ;GO TO 110$ IF NO ERROR
5578 020126 000240 NOP ;RETURN HERE IF ERROR
5579 020130 104000 ERROR ;ERROR # DEFINED BY SEKSTS SUBROUTINE
5580 020132 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
5581 020134 000137 020776 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
5582 020140 013737 021000 001434 110$: MOV 360$,RMDC0 ;RESTORE CYLINDER
5583 020146 012737 000000 001210 MOV #0,$ESCAPE ;;ESCAPE TO 0 ON ERROR
5584
5585 020154 120$:
5586
5587 ;WRITE DATA TO THE DRIVE
5588 020154 012737 177400 001402 MOV #(<C256.+1>,RMWCO ;CHANGE WORD COUNT
5589 020162 012737 107554 001404 MOV #BUFONE+4,RMBA0 ;CHANGE BUS ADDRESS
5590 020170 012737 000061 001400 MOV #WD!GO,RMCS10 ;WRITE DATA COMMAND
5591 020176 012702 001535 MOV #PUTINX,R2 ;LOAD PUT REGISTER INDEX TABLE
5592 020202 112722 000006 MOVB #RMDA,(R2)+
5593 020206 112722 000034 MOVB #RMDC,(R2)+
5594 020212 112722 000032 MOVB #RMOF,(R2)+
5595 020216 112722 000004 MOVB #RMBA,(R2)+
5596 020222 112722 000002 MOVB #RMWC,(R2)+
5597 020226 112722 000000 MOVB #RMCS1,(R2)+
5598 020232 112722 000200 MOVB #200,(R2)+ ;TERMINATE TABLE
5599
5600 020236 004737 044006 JSR PC,PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
5601 020242 000404 BR 130$ ;GO TO 130$ IF NO ERROR
5602 020244 000240 NOP ;RETURN HERE IF ERROR
5603 020246 104000 ERROR ;ERROR # DEFINED BY PUT SUBROUTINE

```

```

5604 020250 000137 020776          JMP      350$          ;GO TO 350$ IF ERROR WAS FOUND
5605 020254          130$:
5606
5607          ;WAIT FOR WRITE COMMAND TO COMPLETE
5608 020254 004737 044346          JSR      PC,TIMOUT    ;GO TO TIMEOUT SUBROUTINE
5609
5610          ;GO READ STATUS FOR WRITE COMMAND
5611 020260 004737 043536          JSR      PC,GET      ;GO READ REGISTERS WITH GET SUBROUTINE
5612 020264 000404          BR       140$          ;GO TO 140$ IF NO ERROR
5613 020266 000240          NOP
5614 020270 104000          ERROR   ;RETURN HERE IF ERROR
5615 020272 000137 020776          JMP      350$          ;ERROR # DEFINED BY GET SUBROUTINE
5616 020276          140$:
5617
5618          ;CHECK FOR ERRORS DURING WRITE OPERATION
5619 020276 004737 044532          JSR      PC,PRIERR   ;GO CHECK FOR PRIMARY ERRORS
5620 020302 000405          BR       150$          ;GO TO 150$ IF NO ERROR
5621 020304 000240          NOP
5622 020306 104000          ERROR   ;RETURN HERE IF ERROR
5623 020310 004736          JSR      PC,@(SP)+   ;ERROR # DEFINED BY PRIERR SUBROUTINE
5624 020312 000137 020776          JMP      350$          ;GO BACK FOR MORE ERROR CHECKS
5625 020316          150$:
5626 020316 004737 057036          JSR      PC,DTASTS   ;GO VERIFY RESULTS OF DATA TRANSFER
5627 020322 000405          BR       160$          ;GO TO 160$ IF NO ERROR
5628 020324 000240          NOP
5629 020326 104000          ERROR   ;RETURN HERE IF ERROR
5630 020330 004736          JSR      PC,@(SP)+   ;ERROR # DEFINED BY DTASTS SUBROUTINE
5631 020332 000137 020776          JMP      350$          ;GO BACK FOR MORE ERROR CHECKS
5632 020336          160$:
5633
5634 020336          170$:
5635 020336 004737 045364          JSR      PC,SECERR   ;GO CHECK FOR SECONDARY ERRORS
5636 020342 000405          BR       180$          ;GO TO 180$ IF NO ERROR
5637 020344 000240          NOP
5638 020346 104000          ERROR   ;RETURN HERE IF ERROR
5639 020350 004736          JSR      PC,@(SP)+   ;ERROR # DEFINED BY SECERR SUBROUTINE
5640 020352 000137 020776          JMP      350$          ;GO BACK FOR MORE ERROR CHECKS
5641 020356          180$:
5642
5643          ;CHANGE LOOP ADDRESSES
5644 020356 012737 020416 001122          MOV      #190$,SLPADR
5645 020364 012737 020416 001124          MOV      #190$,SLPERR
5646 020372 013737 001434 021000          MOV      RMDC0,360$
5647 020400 012737 020410 001210          MOV      #185$,SESCAPE ;;ESCAPE TO 185$ ON ERROR
5648 020406 000413          BR       200$          ;SKIP TO NEXT OPERATION
5649 020410 013737 021000 001434          185$: MOV      360$,RMDC0   ;RESTORE CYLINDER
5650
5651          ;*****
5652          ;LOOP #3      READ
5653
5654 020416          190$:
5655
5656          ;PREPARE DEVICE FOR READ OPERATION
5657 020416 004737 040020          JSR      PC,TSTPRP   ;PREPARE DEVICE FOR TEST
5658 020422 054130          .WORD   054130      ;TASK DESCRIPTOR
5659 020424 000404          BR       200$          ;GO TO 200$ IF NO ERROR

```

K09

CZRMEBO RM03/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 114
T13 WRITE, READ W/ IMPLIED SEEK

SEQ 0114

```

5660 020426 000240      NOP
5661 020430 104000      ERROR
5662 020432 000137 020776 200$: JMP 350$
5663 020436
5664
5665 020436 013737 001434 021000 MOV RMDC0,360$ ;SAVE CYLINDER
5666 020444 012737 001466 001434 MOV #822,RMDC0 ;SEEK TO LAST CYLINDER
5667 020452 012737 000005 001400 MOV #SEEK!GO,RMCS10 ;WRITE SEEK COMMAND
5668 020460 004737 044006 JSR PC PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
5669 020464 000404 BR 210$ ;GO TO 210$ IF NO ERROR
5670 020466 000240 NOP ;RETURN HERE IF ERROR
5671 020470 104000 ERROR ;ERROR # DEFINED BY PUT SUBROUTINE
5672 020472 000137 020776 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
5673 020476
5674
5675 ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
5676 020476 004737 043452 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
5677 020502 004737 044346 JSR PC,TIMOUT ;GO TO TIMEOUT SUBROUTINE
5678 020506 004737 043536 JSR PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
5679 020512 000404 BR 220$ ;GO TO 220$ IF NO ERROR
5680 020514 000240 NOP ;RETURN HERE IF ERROR
5681 020516 104000 ERROR ;ERROR # DEFINED BY GET SUBROUTINE
5682 020520 000137 020776 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
5683 020524
5684 020524 004737 051440 JSR PC,SEKSTS ;GO VERIFY RESULTS OF SEEK OPERATION
5685 020530 000405 BR 230$ ;GO TO 230$ IF NO ERROR
5686 020532 000240 NOP ;RETURN HERE IF ERROR
5687 020534 104000 ERROR ;ERROR # DEFINED BY SEKSTS SUBROUTINE
5688 020536 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
5689 020540 000137 020776 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
5690 020544 013737 021000 001434 230$: MOV 360$,RMDC0 ;RESTORE CYLINDER
5691 020552 012737 000000 001210 MOV #0,$ESCAPE ;;ESCAPE TO 0 ON ERROR
5692
5693 020560 240$:
5694
5695 ;READ DATA FROM DEVICE
5696 020560 012737 000071 001400 MOV #RD!GO,RMCS10 ;READ DATA COMMAND
5697 020566 012737 110560 001404 MOV #BUFTWO+4,RMBAO ;LOAD STARTING BUFFER ADDRESS
5698 020574 012702 001535 MOV #PUTINX,R2 ;LOAD PUT REGISTER INDEX TABLE
5699 020600 112722 000006 MOVB #RMDA,(R2)+
5700 020604 112722 000032 MOVB #RMOF,(R2)+
5701 020610 112722 000034 MOVB #RMDC,(R2)+
5702 020614 112722 000004 MOVB #RMBA,(R2)+
5703 020620 112722 000002 MOVB #RMWC,(R2)+
5704 020624 112722 000000 MOVB #RMCS1,(R2)+
5705 020630 112712 000200 MOVB #200,(R2)
5706
5707 020634 004737 044006 JSR PC,PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
5708 020640 000404 BR 250$ ;GO TO 250$ IF NO ERROR
5709 020642 000240 NOP ;RETURN HERE IF ERROR
5710 020644 104000 ERROR ;ERROR # DEFINED BY PUT SUBROUTINE
5711 020646 000137 020776 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
5712 020652
5713
5714
5715 ;WAIT FOR READ OPERATION TO COMPLETE

```

```

5716 020652 004737 044346      JSR      PC,TIMOUT      ;GO TO TIMOUT SUBROUTINE
5717
5718      ;GO READ STATUS FOR READ OPERATION
5719 020656 004737 043536      JSR      PC,GET        ;GO READ REGISTERS WITH GET SUBROUTINE
5720 020662 000404      BR       260$          ;GO TO 260$ IF NO ERROR
5721 020664 000240      NOP
5722 020666 104000      ERROR   ;RETURN HERE IF ERROR
5723 020670 000137 020776      ERROR   ;ERROR # DEFINED BY GET SUBROUTINE
5724 020674      JMP      350$          ;GO TO 350$ IF ERROR WAS FOUND
5725
5726      260$:
5727      ;CHECK FOR ERRORS DURING READ OPERATION
5728 020674 004737 044532      JSR      PC,PRIERR     ;GO CHECK FOR PRIMARY ERRORS
5729 020700 000405      BR       270$          ;GO TO 270$ IF NO ERROR
5730 020702 000240      NOP
5731 020704 104000      ERROR   ;RETURN HERE IF ERROR
5732 020706 004736      JSR      PC,(SP)+      ;ERROR # DEFINED BY PRIERR SUBROUTINE
5733 020710 000137 020776      JMP      350$          ;GO BACK FOR MORE ERROR CHECKS
5734 020714 004737 057036      ;GO TO 350$ IF ERROR WAS FOUND
5735 020720 000405      JSR      PC,DTASTS     ;GO VERIFY RESULTS OF DATA TRANSFER
5736 020722 000240      BR       280$          ;GO TO 280$ IF NO ERROR
5737 020724 104000      NOP
5738 020726 004736      ERROR   ;RETURN HERE IF ERROR
5739 020730 000137 020776      ERROR   ;ERROR # DEFINED BY DTASTS SUBROUTINE
5740 020734      JMP      350$          ;GO BACK FOR MORE ERROR CHECKS
5741
5742      280$:
5743      290$:
5744 020734 004737 045364      JSR      PC,SECERR     ;GO CHECK FOR SECONDARY ERRORS
5745 020740 000405      BR       300$          ;GO TO 300$ IF NO ERROR
5746 020742 000240      NOP
5747 020744 104000      ERROR   ;RETURN HERE IF ERROR
5748 020746 004736      JSR      PC,(SP)+      ;ERROR # DEFINED BY SECERR SUBROUTINE
5749 020750 000137 020776      JMP      350$          ;GO BACK FOR MORE ERROR CHECKS
5750 020754 004737 043104      ;GO TO 350$ IF ERROR WAS FOUND
5751 020760 107554      JSR      PC,CMPBUF     ;GO COMPARE WRITE, READ DATA BUFFERS
5752 020762 110560      .WORD   BUFONE+4      ;STARTING ADDRESS OF WRITE BUFFER
5753 020764 000404      .WORD   BUFTWO+4      ;STARTING ADDRESS OF READ BUFFER
5754 020766 000240      BR       310$          ;GO TO 310$ IF NO ERROR
5755 020770 104000      NOP
5756 020772 000137 020776      ERROR   ;RETURN HERE IF ERROR
5757 020776      JMP      350$          ;ERROR # DEFINED BY CMPBUF SUBROUTINE
5758
5759      310$:
5760      350$:
5761 021000 000000      BR       370$
5762 021002
5763
5764      360$:
5765      370$:
5766      ;*****
5767      ;*TEST 14      WRITE, WRITE CHECK W/ HEAD SWITCHING
5768      ;*****
5769      †ST14:
5770      SCOPE      ;SCOPE CALL
5771      NOP        ;START OF TEST
5772      MOV        #STACK,SP ;INITIALIZE STACK POINTER
5773      MOV        $BASE,R0  ;R0=UNIBUS ADDRESS
5774      MOV        TSTQUE,R1 ;(R1) = DEVICE BEING TESTED

```

```

5772 021022 012737 000014 001226      MOV      #14,$TESTN      ;;SET TEST NUMBER IN APT MAIL BOX
5773
5774      ;;*****
5775      ;;LOOP #1          FORMAT,WRITE,READ
5776
5777 021030      10$:
5778
5779      ;PREPARE THE DEVICE FOR FORMAT OPERATION
5780 021030 004737 040020      JSR      PC,TSTPRP      ;PREPARE DEVICE FOR TEST
5781 021034 054130      .WORD   054130      ;TASK DESCRIPTOR
5782 021036 000404      BR       20$          ;GO TO 20$ IF NO ERROR
5783 021040 000240      NOP
5784 021042 104000      ERROR   ;RETURN HERE IF ERROR
5785 021044 000137 022022      ERROR   ;ERROR # DEFINED BY TSTPRP SUBROUTINE
5786 021050      JMP      350$        ;GO TO 350$ IF ERROR WAS FOUND
5787
5788      20$:
5789      ;LOAD PARAMETERS AND GENERATE DATA BUFFER
5790 021050 012737 000000 001434      MOV      #0,RMDCO      ;CYLINDER = 0
5791 021056 012737 000037 001406      MOV      #037,RMDAO     ;TRACK = 0, SECTOR = 31
5792 021064 012737 107550 001404      MOV      #BUFONE,RMBA0  ;BUS ADDRESS
5793 021072 012737 176774 001402      MOV      #(<C<2*<256.+2>>+1),RMWCO ;WORD COUNT
5794 021100 012737 010000 001432      MOV      #FMT16,RMOFO   ;16 BIT FORMAT
5795 021106 012737 000063 001400      MOV      #WH!GO,RMCS10  ;WRITE HEADER AND DATA COMMAND
5796
5797      ;VERIFY SECTOR(S) NOT IN BAD SECTOR FILE
5798 021114 004737 040734      JSR      PC,BADSCT     ;CALL BAD SECTOR MODULE
5799 021120 000405      BR       25$          ;GO TO 25$ IF NO ERROR
5800 021122 104401 067466      TYPE    ,SCTMSG       ;TYPE BAD SECTOR MESSAGE
5801 021126 104000      ERROR   ;ERROR # DEFINED BY BADSCT SUBROUTINE
5802 021130 000137 022022      JMP      350$        ;GO TO 350$ IF ERROR WAS FOUND
5803 021134
5804 021134 012737 071046 001174      25$:      MOV      #ONES,$TMP0    ;STARTING ADDRESS OF PATTERN
5805 021142 012737 000001 001176      MOV      #1,$TMP1     ;RANGE OF PATTERN
5806 021150 004737 042640      JSR      PC,GENBUF     ;GO GENERATE BUFFER FOR FORMAT
5807
5808      ;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATION
5809 021154 012702 001535      MOV      #PUTINX,R2   ;R2 = BYTE ENTRY POSITION
5810 021160 112722 000034      MOVB    #RMDC,(R2)+
5811 021164 112722 000006      MOVB    #RMDA,(R2)+
5812 021170 112722 000004      MOVB    #RMBA,(R2)+
5813 021174 112722 000002      MOVB    #RMWC,(R2)+
5814 021200 112722 000032      MOVB    #RMOF,(R2)+
5815 021204 112722 000000      MOVB    #RMCS1,(R2)+
5816 021210 112712 000200      MOVB    #200,(R2)    ;TERMINATE TABLE
5817 021214
5818
5819      30$:
5820      ;FORMAT THE DRIVE
5821 021214 004737 044006      JSR      PC,PUT      ;GO WRITE REGISTERS WITH PUT SUBROUTINE
5822 021220 000404      BR       40$          ;GO TO 40$ IF NO ERROR
5823 021222 000240      NOP
5824 021224 104000      ERROR   ;RETURN HERE IF ERROR
5825 021226 000137 022022      ERROR   ;ERROR # DEFINED BY PUT SUBROUTINE
5826 021232      JMP      350$        ;GO TO 350$ IF ERROR WAS FOUND
5827
      40$:
      ;SETUP GET REGISTER INDEX TABLE FOR READING STATUS

```



```

5828 021232 004737 043452          JSR    PC,GETSTS          ;GO TO GETSTS SUBROUTINE
5829
5830          ;WAIT FOR THE FORMAT TO COMPLETE
5831 021236 004737 044346          JSR    PC,TIMOUT          ;GO TO TIMOUT SUBROUTINE
5832
5833          ;GO READ STATUS FOR FORMAT OPERATION
5834 021242 004737 043536          JSR    PC,GET          ;GO READ REGISTERS WITH GET SUBROUTINE
5835 021246 000404                    BR     50$                ;GO TO 50$ IF NO ERROR
5836 021250 000240                    NOP                                ;RETURN HERE IF ERROR
5837 021252 104000                    ERROR          ;ERROR # DEFINED BY GET SUBROUTINE
5838 021254 000137 022022          JMP    350$              ;GO TO 350$ IF ERROR WAS FOUND
5839 021260
5840
5841          ;VERIFY NO ERRORS DURING FORMAT
5842 021260 004737 057036          JSR    PC,DTASTS        ;GO VERIFY RESULTS OF DATA TRANSFER
5843 021264 000405                    BR     60$                ;GO TO 60$ IF NO ERROR
5844 021266 000240                    NOP                                ;RETURN HERE IF ERROR
5845 021270 104000                    ERROR          ;ERROR # DEFINED BY DTASTS SUBROUTINE
5846 021272 004736                    JSR    PC,(SP)+          ;GO BACK FOR MORE ERROR CHECKS
5847 021274 000137 022022          JMP    350$              ;GO TO 350$ IF ERROR WAS FOUND
5848 021300
5849
5850          ;MOVE LOOP ADDRESSES TO NEXT OPERATION
5851 021300 012737 021360 001122      MOV    #70$,$LPADR
5852 021306 012737 021360 001124      MOV    #70$,$LPERR
5853          ;REGENERATE BUFFER
5854 021314 012737 177000 001402      MOV    #(<C(2*256.)+1>),RMWCO ;CHANGE WORD COUNT
5855 021322 012737 107550 001404      MOV    #BUFONE,RMBAO      ;CHANGE BUS ADDRESS
5856 021330 012737 071110 001174      MOV    #ZEROS,$TMPO       ;STARTING ADDRESS
5857 021336 012737 000001 001176      MOV    #1,$TMP1           ;RANGE
5858 021344 012737 000060 001400      MOV    #WD,RMCS10         ;WRITE DATA
5859 021352 004737 042640                    JSR    PC,GENBUF          ;GENERATE BUFFER
5860 021356 000410                    BR     80$                ;SKIP TO WRITE OPERATION
5861
5862          ;*****
5863          ;LOOP #2          WRITE,READ
5864
5865 021360
5866
5867
5868          ;PREPARE DEVICE FOR WRITE OPERATION
5869 021360 004737 040020          JSR    PC,TSTPRP        ;PREPARE DEVICE FOR TEST
5870 021364 054130                    .WORD 054130 ;TASK DESCRIPTOR
5871 021366 000404                    BR     80$                ;GO TO 80$ IF NO ERROR
5872 021370 000240                    NOP                                ;RETURN HERE IF ERROR
5873 021372 104000                    ERROR          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
5874 021374 000137 022022          JMP    350$              ;GO TO 350$ IF ERROR WAS FOUND
5875 021400
5876
5877
5878 021400
5879
5880          ;WRITE DATA TO THE DRIVE
5881 021400 012737 000061 001400      MOV    #WD,GO,RMCS10     ;WRITE DATA COMMAND
5882 021406 012702 001535                    MOV    #PUTINX,R2        ;LOAD PUT REGISTER INDEX TABLE
5883 021412 112722 000006                    MOV    #RMDA,(R2)+

```

5884	021416	112722	000034		MOV B	#RMD C, (R2)+	
5885	021422	112722	000032		MOV B	#RMO F, (R2)+	
5886	021426	112722	000004		MOV B	#RMB A, (R2)+	
5887	021432	112722	000002		MOV B	#RMW C, (R2)+	
5888	021436	112722	000000		MOV B	#RMC S1, (R2)+	
5889	021442	112722	000200		MOV B	#200, (R2)+	; TERMINATE TABLE
5890							
5891	021446	004737	044006		JSR	PC, PUT	; GO WRITE REGISTERS WITH PUT SUBROUTINE
5892	021452	000404			BR	130\$; GO TO 130\$ IF NO ERROR
5893	021454	000240			NOP		; RETURN HERE IF ERROR
5894	021456	104000			ERROR		; ERROR # DEFINED BY PUT SUBROUTINE
5895	021460	000137	022022		JMP	350\$; GO TO 350\$ IF ERROR WAS FOUND
5896	021464			130\$:			
5897							
5898							; SETUP REGISTER INPUT BUFFER FOR READING STATUS
5899	021464	004737	043452		JSR	PC, GETSTS	; GO TO GETSTS SUBROUTINE
5900							
5901							; WAIT FOR WRITE COMMAND TO COMPLETE
5902	021470	004737	044346		JSR	PC, TIMEOUT	; GO TO TIMEOUT SUBROUTINE
5903							
5904							; GO READ STATUS FOR WRITE COMMAND
5905	021474	004737	043536		JSR	PC, GET	; GO READ REGISTERS WITH GET SUBROUTINE
5906	021500	000404			BR	140\$; GO TO 140\$ IF NO ERROR
5907	021502	000240			NOP		; RETURN HERE IF ERROR
5908	021504	104000			ERROR		; ERROR # DEFINED BY GET SUBROUTINE
5909	021506	000137	022022		JMP	350\$; GO TO 350\$ IF ERROR WAS FOUND
5910	021512			140\$:			
5911							
5912							; CHECK FOR ERRORS DURING WRITE OPERATION
5913	021512	004737	044532		JSR	PC, PRIERR	; GO CHECK FOR PRIMARY ERRORS
5914	021516	000405			BR	150\$; GO TO 150\$ IF NO ERROR
5915	021520	000240			NOP		; RETURN HERE IF ERROR
5916	021522	104000			ERROR		; ERROR # DEFINED BY PRIERR SUBROUTINE
5917	021524	004736			JSR	PC, 2(SP)+	; GO BACK FOR MORE ERROR CHECKS
5918	021526	000137	022022		JMP	350\$; GO TO 350\$ IF ERROR WAS FOUND
5919	021532			150\$:			
5920	021532	004737	057036		JSR	PC, DTASTS	; GO VERIFY RESULTS OF DATA TRANSFER
5921	021536	000405			BR	160\$; GO TO 160\$ IF NO ERROR
5922	021540	000240			NOP		; RETURN HERE IF ERROR
5923	021542	104000			ERROR		; ERROR # DEFINED BY DTASTS SUBROUTINE
5924	021544	004736			JSR	PC, 2(SP)+	; GO BACK FOR MORE ERROR CHECKS
5925	021546	000137	022022		JMP	350\$; GO TO 350\$ IF ERROR WAS FOUND
5926	021552			160\$:			
5927							
5928	021552			170\$:			
5929	021552	004737	045364		JSR	PC, SECERR	; GO CHECK FOR SECONDARY ERRORS
5930	021556	000405			BR	180\$; GO TO 180\$ IF NO ERROR
5931	021560	000240			NOP		; RETURN HERE IF ERROR
5932	021562	104000			ERROR		; ERROR # DEFINED BY SECERR SUBROUTINE
5933	021564	004736			JSR	PC, 2(SP)+	; GO BACK FOR MORE ERROR CHECKS
5934	021566	000137	022022		JMP	350\$; GO TO 350\$ IF ERROR WAS FOUND
5935	021572			180\$:			
5936							
5937							; CHANGE LOOP ADDRESSES
5938	021572	012737	021610	001122	MOV	#190\$, \$LPADR	
5939	021600	012737	021610	001124	MOV	#190\$, \$LPERR	

```

5940 021606 000410 BR 200$ ;SKIP TO NEXT OPERATION
5941
5942 ;*****
5943 ;LOOP #3 READ
5944
5945 021610 190$:
5946
5947 ;PREPARE DEVICE FOR READ OPERATION
5948 021610 004737 040020 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
5949 021614 054130 .WORD 054130 ;TASK DESCRIPTOR
5950 021616 000404 BR 200$ ;GO TO 200$ IF NO ERROR
5951 021620 000240 NOP ;RETURN HERE IF ERROR
5952 021622 104000 ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
5953 021624 000137 022022 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
5954 021630 200$:
5955
5956 021630 240$:
5957
5958 ;READ DATA FROM DEVICE
5959 021630 012737 000051 001400 MOV #WCD!GO,RMCS10 ;READ DATA COMMAND
5960 021636 012702 001535 MOV #PUTINX,R2 ;LOAD PUT REGISTER INDEX TABLE
5961 021642 112722 000006 MOVB #RMDA,(R2)+
5962 021646 112722 000032 MOVB #RMOF,(R2)+
5963 021652 112722 000034 MOVB #RMDC,(R2)+
5964 021656 112722 000004 MOVB #RMBB,(R2)+
5965 021662 112722 000002 MOVB #RMWC,(R2)+
5966 021666 112722 000000 MOVB #RMCS1,(R2)+
5967 021672 112712 000200 MOVB #200,(R2)
5968
5969 021676 004737 044006 JSR PC,PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
5970 021702 000404 BR 250$ ;GO TO 250$ IF NO ERROR
5971 021704 000240 NOP ;RETURN HERE IF ERROR
5972 021706 104000 ERROR ;ERROR # DEFINED BY PUT SUBROUTINE
5973 021710 000137 022022 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
5974 021714 250$:
5975
5976 ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
5977 021714 004737 043452 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
5978
5979 ;WAIT FOR READ OPERATION TO COMPLETE
5980 021720 004737 044346 JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
5981
5982 ;GO READ STATUS FOR READ OPERATION
5983 021724 004737 043536 JSR PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
5984 021730 000404 BR 260$ ;GO TO 260$ IF NO ERROR
5985 021732 000240 NOP ;RETURN HERE IF ERROR
5986 021734 104000 ERROR ;ERROR # DEFINED BY GET SUBROUTINE
5987 021736 000137 022022 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
5988 021742 260$:
5989
5990 ;CHECK FOR ERRORS DURING READ OPERATION
5991 021742 004737 044532 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
5992 021746 000405 BR 270$ ;GO TO 270$ IF NO ERROR
5993 021750 000240 NOP ;RETURN HERE IF ERROR
5994 021752 104000 ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
5995 021754 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS

```

```

5996 021756 000137 022022          JMP      350$          ;GO TO 350$ IF ERROR WAS FOUND
5997 021762          270$:
5998 021762 004737 057036          JSR      PC,DTASTS    ;GO VERIFY RESULTS OF DATA TRANSFER
5999 021766 000405          BR       280$          ;GO TO 280$ IF NO ERROR
6000 021770 000240          NOP
6001 021772 104000          ERROR   ;RETURN HERE IF ERROR
6002 021774 004736          JSR      PC,(SP)+     ;ERROR # DEFINED BY DTASTS SUBROUTINE
6003 021776 000137 022022          JMP      350$          ;GO BACK FOR MORE ERROR CHECKS
6004 022002          280$:                ;GO TO 350$ IF ERROR WAS FOUND
6005
6006 022002          290$:
6007 022002 004737 045364          JSR      PC,SECERR    ;GO CHECK FOR SECONDARY ERRORS
6008 022006 000405          BR       300$          ;GO TO 300$ IF NO ERROR
6009 022010 000240          NOP
6010 022012 104000          ERROR   ;RETURN HERE IF ERROR
6011 022014 004736          JSR      PC,(SP)+     ;ERROR # DEFINED BY SECERR SUBROUTINE
6012 022016 000137 022022          JMP      350$          ;GO BACK FOR MORE ERROR CHECKS
6013 022022          300$:                ;GO TO 350$ IF ERROR WAS FOUND
6014
6015 022022          350$:
6016 *****
6017 ;*TEST 15 WRITE, WRITE CHECK W/ MID-TRANSFER SEEK
6018 *****
6019 ;*T15:
6020          SCOPE          ;SCOPE CALL
6021          NOP           ;START OF TEST
6022          MOV          #STACK,SP ;INITIALIZE STACK POINTER
6023          MOV          $BASE,R0  ;R0=UNIBUS ADDRESS
6024          MOV          !STQUE,R1  ;(R1) = DEVICE BEING TESTED
6025          MOV          #15,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
6026
6027 *****
6028 ;LOOP #1          FORMAT,WRITE,READ
6029
6030 022050          10$:
6031
6032 ;PREPARE THE DEVICE FOR FORMAT OPERATION
6033 022050 004737 040020          JSR      PC,TSTPRP    ;PREPARE DEVICE FOR TEST
6034 022054 054130          .WORD   054130 ;TASK DESCRIPTOR
6035 022056 000404          BR       20$          ;GO TO 20$ IF NO ERROR
6036 022060 000240          NOP
6037 022062 104000          ERROR   ;RETURN HERE IF ERROR
6038 022064 000137 023042          JMP      350$          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
6039 022070          20$:                ;GO TO 350$ IF ERROR WAS FOUND
6040
6041 ;LOAD PARAMETERS AND GENERATE DATA BUFFER
6042 022070 012737 000000 001434          MOV      #0,RMDC0     ;CYLINDER = 0
6043 022076 012737 002037 001406          MOV      #2037,RMDA0  ;TRACK = 4, SECTOR = 31
6044 022104 012737 107550 001404          MOV      #BUFONE,RMBA0 ;BUS ADDRESS
6045 022112 012737 176774 001402          MOV      #<?C<2*<256.+2>>+1>,RMWC0 ;WORD COUNT
6046 022120 012737 010000 001432          MOV      #FMT16,RMFO0 ;16 BIT FORMAT
6047 022126 012737 000063 001400          MOV      #WH!GO,RMCS10 ;WRITE HEADER AND DATA COMMAND
6048
6049 ;VERIFY SECTOR(S) NOT IN BAD SECTOR FILE
6050
6051 022134 004737 040734          JSR      PC,BADSCT    ;CALL BAD SECTOR MODULE

```

E10

CZRMEBO RM03/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 121
T15 WRITE, WRITE CHECK W/ MID-TRANSFER SEEK

SEQ 0121

6052	022140	000405			BR	25\$;GO TO 25\$ IF NO ERROR
6053	022142	104401	067466		TYPE	,SCTMSG		;TYPE BAD SECTOR MESSAGE
6054	022146	104000			ERROR			;ERROR # DEFINED BY BADSCT SUBROUTINE
6055	022150	000137	023042		JMP	350\$;GO TO 350\$ IF ERROR WAS FOUND
6056	022154							
6057	022154	012737	071110	001174	25\$:	MOV	#ZEROS,\$TMP0	;STARTING ADDRESS OF PATTERN
6058	022162	012737	000001	001176		MOV	#1,\$TMP1	;RANGE OF PATTERN
6059	022170	004737	042640			JSR	PC,GENBUF	;GO GENERATE BUFFER FOR FORMAT
6060								
6061								
6062	022174	012702	001535					
6063	022200	112722	000034			MOV	#PUTINX,R2	;R2 = BYTE ENTRY POSITION
6064	022204	112722	000006			MOVB	#RMDC,(R2)+	
6065	022210	112722	000004			MOVB	#RMDA,(R2)+	
6066	022214	112722	000002			MOVB	#RMBA,(R2)+	
6067	022220	112722	000032			MOVB	#RMWC,(R2)+	
6068	022224	112722	000000			MOVB	#RMOF,(R2)+	
6069	022230	112712	000200			MOVB	#RMCS1,(R2)+	
6070	022234				30\$:			;TERMINATE TABLE
6071								
6072								
6073	022234	004737	044006					
6074	022240	000404				JSR	PC,PUT	;GO WRITE REGISTERS WITH PUT SUBROUTINE
6075	022242	000240				BR	40\$;GO TO 40\$ IF NO ERROR
6076	022244	104000				NOP		;RETURN HERE IF ERROR
6077	022246	000137	023042			ERROR		;ERROR # DEFINED BY PUT SUBROUTINE
6078	022252					JMP	350\$;GO TO 350\$ IF ERROR WAS FOUND
6079					40\$:			
6080								
6081	022252	004737	043452					
6082								
6083								
6084	022256	004737	044346					
6085								
6086								
6087	022262	004737	043536					
6088	022266	000404						
6089	022270	000240						
6090	022272	104000						
6091	022274	000137	023042					
6092	022300							
6093								
6094								
6095	022300	004737	057036					
6096	022304	000405						
6097	022306	000240						
6098	022310	104000						
6099	022312	004736						
6100	022314	000137	023042					
6101	022320							
6102								
6103								
6104	022320	012737	022400	001122				
6105	022326	012737	022400	001124				
6106								
6107	022334	012737	000060	001400				

F10

CZRMEBO RM03/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 122
T15 WRITE, WRITE CHECK W/ MID-TRANSFER SEEK

SEQ 0122

```

6108 022342 012737 177000 001402      MOV      #(<C<2*256.>+1),RMWCO ;CHANGE WORD COUNT
6109 022350 012737 107550 001404      MOV      #BUFONE,RMBA0 ;CHANGE BUS ADDRSS
6110 022356 012737 071046 001174      MOV      #ONES,$TMP0 ;PATTERN ADDRESS
6111 022364 012737 000001 001176      MOV      #1,$TMP1 ;PATTERN RANGE
6112 022372 004737 042640      JSR      PC,GENBUF
6113 022376 000410      BR       80$ ;SKIP TO WRITE OPERATION
6114
6115 ;*****
6116 ;LOOP #2 WRITE,READ
6117
6118 022400      70$:
6119
6120
6121 ;PREPARE DEVICE FOR WRITE OPERATION
6122 022400 004737 040020      JSR      PC,TSTPRP ;PREPARE DEVICE FOR TEST
6123 022404 054130      .WORD   054130 ;TASK DESCRIPTOR
6124 022406 000404      BR       80$ ;GO TO 80$ IF NO ERROR
6125 022410 000240      NOP
6126 022412 104000      ERROR   ;RETURN HERE IF ERROR
6127 022414 000137 023042      JMP      350$ ;ERROR # DEFINED BY TSTPRP SUBROUTINE
6128 022420      80$: ;GO TO 350$ IF ERROR WAS FOUND
6129
6130
6131 022420      120$:
6132
6133 ;WRITE DATA TO THE DRIVE
6134 022420 012737 000061 001400      MOV      #WD!GO,RMCS10 ;WRITE DATA COMMAND
6135 022426 012702 001535      MOV      #PUTINX,R2 ;LOAD PUT REGISTER INDEX TABLE
6136 022432 112722 000006      MOVB    #RMDA,(R2)+
6137 022436 112722 000034      MOVB    #RMDC,(R2)+
6138 022442 112722 000032      MOVB    #RMOF,(R2)+
6139 022446 112722 000004      MOVB    #RMBA,(R2)+
6140 022452 112722 000002      MOVB    #RMWC,(R2)+
6141 022456 112722 000000      MOVB    #RMCS1,(R2)+
6142 022462 112722 000200      MOVB    #200,(R2)+ ;TERMINATE TABLE
6143
6144 022466 004737 044006      JSR      PC,PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
6145 022472 000404      BR       130$ ;GO TO 130$ IF NO ERROR
6146 022474 000240      NOP
6147 022476 104000      ERROR   ;RETURN HERE IF ERROR
6148 022500 000137 023042      JMP      350$ ;ERROR # DEFINED BY PUT SUBROUTINE
6149 022504      130$: ;GO TO 350$ IF ERROR WAS FOUND
6150
6151 ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
6152 022504 004737 043452      JSR      PC,GETSTS ;GO TO GETSTS SUBROUTINE
6153
6154 ;WAIT FOR WRITE COMMAND TO COMPLETE
6155 022510 004737 044346      JSR      PC,TIMOUT ;GO TO TIMEOUT SUBROUTINE
6156
6157 ;GO READ STATUS FOR WRITE COMMAND
6158 022514 004737 043536      JSR      PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
6159 022520 000404      BR       140$ ;GO TO 140$ IF NO ERROR
6160 022522 000240      NOP
6161 022524 104000      ERROR   ;RETURN HERE IF ERROR
6162 022526 000137 023042      JMP      350$ ;ERROR # DEFINED BY GET SUBROUTINE
6163 022532      140$: ;GO TO 350$ IF ERROR WAS FOUND

```

G10

CZRMEBO RM03/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 123
T15 WRITE, WRITE CHECK W/ MID-TRANSFER SEEK

SEQ 0123

```

6164
6165
6166 022532 004737 044532
6167 022536 000405
6168 022540 000240
6169 022542 104000
6170 022544 004736
6171 022546 000137 023042
6172 022552
6173 022552 004737 057036
6174 022556 000405
6175 022560 000240
6176 022562 104000
6177 022564 004736
6178 022566 000137 023042
6179 022572
6180
6181 022572
6182 022572 004737 045364
6183 022576 000405
6184 022600 000240
6185 022602 104000
6186 022604 004736
6187 022606 000137 023042
6188 022612
6189
6190
6191 022612 012737 022630 001122
6192 022620 012737 022630 001124
6193 022626 000410
6194
6195
6196
6197
6198 022630
6199
6200
6201 022630 004737 040020
6202 022634 054130
6203 022636 000404
6204 022640 000240
6205 022642 104000
6206 022644 000137 023042
6207 022650
6208
6209 022650
6210
6211
6212 022650 012737 000051 001400
6213 022656 012702 001535
6214 022662 112722 000006
6215 022666 112722 000032
6216 022672 112722 000034
6217 022676 112722 000004
6218 022702 112722 000002
6219 022706 112722 000000

;CHECK FOR ERRORS DURING WRITE OPERATION
JSR PC_PRIERR ;GO CHECK FOR PRIMARY ERRORS
BR 150$ ;GO TO 150$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
JSR PC_2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND

150$:
JSR PC_DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
BR 160$ ;GO TO 160$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
JSR PC_2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND

160$:
170$:
JSR PC_SECERR ;GO CHECK FOR SECONDARY ERRORS
BR 180$ ;GO TO 180$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
JSR PC_2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND

180$:
;CHANGE LOOP ADDRESSES
MOV #190$, $LPADR
MOV #190$, $LPERR
BR 200$ ;SKIP TO NEXT OPERATION

;*****
;LOOP #3 READ

190$:
;PREPARE DEVICE FOR READ OPERATION
JSR PC_TSTPRP ;PREPARE DEVICE FOR TEST
WORD 054130 ;TASK DESCRIPTOR
BR 200$ ;GO TO 200$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND

200$:
240$:
;READ DATA FROM DEVICE
MOV #WCD!GO, RMCS10 ;READ DATA COMMAND
MOV #PUTINX, R2 ;LOAD PUT REGISTER INDEX TABLE
MOVB #RMDA, (R2)+
MOVB #RMOF, (R2)+
MOVB #RMDC, (R2)+
MOVB #RMBA, (R2)+
MOVB #RMWC, (R2)+
MOVB #RMCS1, (R2)+

```

```

6220 022712 112712 000200      MOVB      #200,(R2)
6221
6222 022716 004737 044006      JSR      PC,PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
6223 022722 000404      BR      250$ ;GO TO 250$ IF NO ERROR
6224 022724 000240      NOP     ;RETURN HERE IF ERROR
6225 022726 104000      ERROR  ;ERROR # DEFINED BY PUT SUBROUTINE
6226 022730 000137 023042      JMP     350$ ;GO TO 350$ IF ERROR WAS FOUND
6227 022734
250$:
6228
6229 ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
6230 022734 004737 043452      JSR     PC,GETSTS ;GO TO GETSTS SUBROUTINE
6231
6232 ;WAIT FOR READ OPERATION TO COMPLETE
6233 022740 004737 044346      JSR     PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
6234
6235 ;GO READ STATUS FOR READ OPERATION
6236 022744 004737 043536      JSR     PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
6237 022750 000404      BR      260$ ;GO TO 260$ IF NO ERROR
6238 022752 000240      NOP     ;RETURN HERE IF ERROR
6239 022754 104000      ERROR  ;ERROR # DEFINED BY GET SUBROUTINE
6240 022756 000137 023042      JMP     350$ ;GO TO 350$ IF ERROR WAS FOUND
6241 022762
260$:
6242
6243 ;CHECK FOR ERRORS DURING READ OPERATION
6244 022762 004737 044532      JSR     PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
6245 022766 000405      BR      270$ ;GO TO 270$ IF NO ERROR
6246 022770 000240      NOP     ;RETURN HERE IF ERROR
6247 022772 104000      ERROR  ;ERROR # DEFINED BY PRIERR SUBROUTINE
6248 022774 004736      JSR     PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
6249 022776 000137 023042      JMP     350$ ;GO TO 350$ IF ERROR WAS FOUND
6250 023002
270$:
6251 023002 004737 057036      JSR     PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
6252 023006 000405      BR      280$ ;GO TO 280$ IF NO ERROR
6253 023010 000240      NOP     ;RETURN HERE IF ERROR
6254 023012 104000      ERROR  ;ERROR # DEFINED BY DTASTS SUBROUTINE
6255 023014 004736      JSR     PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
6256 023016 000137 023042      JMP     350$ ;GO TO 350$ IF ERROR WAS FOUND
6257 023022
280$:
6258
6259 ;GO CHECK FOR SECONDARY ERRORS
6260 023022 004737 045364      JSR     PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
6261 023026 000405      BR      300$ ;GO TO 300$ IF NO ERROR
6262 023030 000240      NOP     ;RETURN HERE IF ERROR
6263 023032 104000      ERROR  ;ERROR # DEFINED BY SECERR SUBROUTINE
6264 023034 004736      JSR     PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
6265 023036 000137 023042      JMP     350$ ;GO TO 350$ IF ERROR WAS FOUND
6266 023042
300$:
6267
6268 023042
350$:
6269 ;*****
6270 ;*TEST 16 WRITE, READ W/ HCE ERROR
6271 ;*****
6272 023042
↑ST16:
6273 023042 000004      SCOPE  ;SCOPE CALL
6274 023044 000240      NOP    ;START OF TEST
6275 023046 012706 001100      MOV    #STACK,SP ;INITIALIZE STACK POINTER

```



```

6276 023052 013700 001276      MOV      $BASE,R0      ;R0=UNIBUS ADDRESS
6277 023056 013701 001450      MOV      TSTQUE,R1     ;(R1) = DEVICE BEING TESTED
6278 023062 012737 000016 001226  MOV      #16,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX
6279
6280 ;:*****
6281 ;:LOOP #1          FORMAT,WRITE,READ
6282
6283 023070 012704 000000      MOV      #0,R4         ;R4=HEADER WORD
6284 ;:*****
6285 ;:NOTE: BSE NOT TESTED
6286 023074 012703 020000 10$:  MOV      #BIT13,R3    ;R3=HCE BIT
6287 023100 15$:
6288
6289 ;PREPARE THE DEVICE FOR FORMAT OPERATION
6290 023100 004737 040020      JSR      PC,TSTPRP     ;PREPARE DEVICE FOR TEST
6291 023104 054130      .WORD   054130 ;TASK DESCRIPTOR
6292 023106 000404      BR      20$           ;GO TO 20$ IF NO ERROR
6293 023110 000240      NOP
6294 023112 104000      ERROR   ;RETURN HERE IF ERROR
6295 023114 000137 024724      JMP      370$         ;ERROR # DEFINED BY TSTPRP SUBROUTINE
6296 023120 20$:          ;GO TO 370$ IF ERROR WAS FOUND
6297
6298 ;LOAD PARAMETERS AND GENERATE DATA BUFFER
6299 023120 012737 000000 001434  MOV      #0,RMDCO      ;CYLINDER = 0
6300 023126 012737 000000 001406  MOV      #0,RMDAO      ;TRACK = 0, SECTOR = 0
6301 023134 012737 107550 001404  MOV      #BUFONE,RMBAO ;BUS ADDRESS
6302 023142 012737 177376 001402  MOV      #(<C<2+256.>+1),RMWCO ;WORD COUNT
6303 023150 012737 010000 001432  MOV      #FMT16,RMOFO  ;16 BIT FORMAT
6304 023156 012737 000063 001400  MOV      #WH!GO,RMCS10 ;WRITE HEADER AND DATA COMMAND
6305
6306 ;VERIFY SECTOR(S) NOT IN BAD SECTOR FILE
6307
6308 023164 004737 040734      JSR      PC,BADSCT    ;CALL BAD SECTOR MODULE
6309 023170 000405 25$:          BR      25$           ;GO TO 25$ IF NO ERROR
6310 023172 104401 067466      TYPE    ,SCTMSG      ;TYPE BAD SECTOR MESSAGE
6311 023176 104000      ERROR   ;ERROR # DEFINED BY BADSCT SUBROUTINE
6312 023200 000137 024536      JMP      350$         ;GO TO 350$ IF ERROR WAS FOUND
6313 25$:
6314 023204 012737 071110 001174  MOV      #ZEROS,$TMPD  ;STARTING ADDRESS OF PATTERN
6315 023212 012737 000001 001176  MOV      #1,$TMP1     ;RANGE OF PATTERN
6316 023220 004737 042640      JSR      PC,GENBUF    ;GO GENERATE BUFFER FOR FORMAT
6317
6318 ;CHANGE HEADER WORD TO FORCE HCE DURING WRITE & READ
6319 023224 030364 107550      BIT     R3,BUFONE(R4) ;SET OR RESET FOR HCE??
6320 023230 001403 26$:          BEQ     26$           ;
6321 023232 040364 107550      BIC     R3,BUFONE(R4) ;RESET BIT FOR HCE
6322 023236 000402 27$:          BR      27$           ;
6323 023240 050364 107550 26$:          BIS     R3,BUFONE(R4) ;SET FOR HCE
6324 023244 27$:
6325 ;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATION
6326 023244 012702 001535      MOV      #PUTINX,R2   ;R2 = BYTE ENTRY POSITION
6327 023250 112722 000034      MOVB   #RMDC,(R2)+
6328 023254 112722 000006      MOVB   #RMDA,(R2)+
6329 023260 112722 000004      MOVB   #RMBA,(R2)+
6330 023264 112722 000002      MOVB   #RMWC,(R2)+
6331 023270 112722 000032      MOVB   #RMOF,(R2)+

```

6332 023274 112722 000000
6333 023300 112712 000200
6334 023304
6335
6336
6337 023304 004737 044006
6338 023310 000404
6339 023312 000240
6340 023314 104000
6341 023316 000137 024536
6342 023322
6343
6344
6345 023322 004737 043452
6346
6347
6348 023326 004737 044346
6349
6350
6351 023332 004737 043536
6352 023336 000404
6353 023340 000240
6354 023342 104000
6355 023344 000137 024536
6356 023350
6357
6358
6359 023350 004737 057036
6360 023354 000405
6361 023356 000240
6362 023360 104000
6363 023362 004736
6364 023364 000137 024536
6365 023370
6366
6367
6368 023370 012737 023406 001122
6369 023376 012737 023406 001124
6370 023404 000410
6371
6372
6373
6374
6375 023406
6376
6377
6378
6379 023406 004737 040020
6380 023412 054130
6381 023414 000404
6382 023416 000240
6383 023420 104000
6384 023422 000137 024536
6385 023426
6386
6387

```

MOVW #RMC51,(R2)+
MOVW #200,(R2) ;TERMINATE TABLE
30$:
;FORMAT THE DRIVE
JSR PC,PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
BR 40$ ;GO TO 40$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY PUT SUBROUTINE
JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
40$:
;SETUP GET REGISTER INDEX TABLE FOR READING STATUS
JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
;WAIT FOR THE FORMAT TO COMPLETE
JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
;GO READ STATUS FOR FORMAT OPERATION
JSR PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
BR 50$ ;GO TO 50$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY GET SUBROUTINE
JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
50$:
;VERIFY NO ERRORS DURING FORMAT
JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
BR 60$ ;GO TO 60$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
60$:
;MOVE LOOP ADDRESSES TO NEXT OPERATION
MOV #70$,SLPADR
MOV #70$,SLPERR
BR 80$ ;SKIP TO WRITE OPERATION
;*****
;LOOP #2 WRITE,READ
70$:
;PREPARE DEVICE FOR WRITE OPERATION
JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
WORD 054130 ;TASK DESCRIPTOR
BR 80$ ;GO TO 80$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
80$:

```

K10

CZRMEBO RMO3/2 FCTNL TST 3
 CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 127
 T16 WRITE, READ W/ HCE ERROR

SEQ 0127

```

6388 023426          120$:
6389
6390          ;WRITE DATA TO THE DRIVE
6391 023426 012737 107554 001404  MOV      #BUFONE+4,RMBA0          ;CHANGE BUS ADDRESS
6392 023434 012737 177400 001402  MOV      #(<C256.+1>,RMWCO        ;CHANGE WORD COUNT
6393 023442 012737 000061 001400  MOV      #WD!GO,RMCS10          ;WRITE DATA COMMAND
6394 023450 012702 001535          MOV      #PUTINX,R2            ;LOAD PUT REGISTER INDEX TABLE
6395 023454 112722 000006          MOVVB   #RMDA,(R2)+
6396 023460 112722 000034          MOVVB   #RMDC,(R2)+
6397 023464 112722 000032          MOVVB   #RMOF,(R2)+
6398 023470 112722 000004          MOVVB   #RMBA,(R2)+
6399 023474 112722 000002          MOVVB   #RMWC,(R2)+
6400 023500 112722 000000          MOVVB   #RMCS1,(R2)+
6401 023504 112722 000200          MOVVB   #200,(R2)+          ;TERMINATE TABLE
6402
6403 023510 004737 044006          JSR     PC,PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
6404 023514 000404          BR     130$          ;GO TO 130$ IF NO ERROR
6405 023516 000240          NOP
6406 023520 104000          ERROR  ;RETURN HERE IF ERROR
6407 023522 000137 024536          JMP     350$        ;ERROR # DEFINED BY PUT SUBROUTINE
6408 023526          ;GO TO 350$ IF ERROR WAS FOUND
6409
6410          ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
6411 023526 004737 043452          JSR     PC,GETSTS    ;GO TO GETSTS SUBROUTINE
6412
6413          ;WAIT FOR WRITE COMMAND TO COMPLETE
6414 023532 004737 044346          JSR     PC,TIMOUT    ;GO TO TIMEOUT SUBROUTINE
6415
6416          ;GO READ STATUS FOR WRITE COMMAND
6417 023536 004737 043536          JSR     PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
6418 023542 000404          BR     140$          ;GO TO 140$ IF NO ERROR
6419 023544 000240          NOP
6420 023546 104000          ERROR  ;RETURN HERE IF ERROR
6421 023550 000137 024536          JMP     350$        ;ERROR # DEFINED BY GET SUBROUTINE
6422 023554          ;GO TO 350$ IF ERROR WAS FOUND
6423
6424          ;CHECK FOR ERRORS DURING WRITE OPERATION
6425 023554 004737 044532          JSR     PC,PRIERR    ;GO CHECK FOR PRIMARY ERRORS
6426 023560 000405          BR     145$          ;GO TO 145$ IF NO ERROR
6427 023562 000240          NOP
6428 023564 104000          ERROR  ;RETURN HERE IF ERROR
6429 023566 004736          JSR     PC,(SP)+    ;ERROR # DEFINED BY PRIERR SUBROUTINE
6430 023570 000137 024536          JMP     350$        ;GO BACK FOR MORE ERROR CHECKS
6431 023574          ;GO TO 350$ IF ERROR WAS FOUND
6432
6433          ;DECODE THE TYPE OF ERROR PRODUCED BY THE BIT POSITION OF R3
6434 023574 005704          TST     R4
6435 023576 001006          BNE    146$          ;IS THIS THE FIRST HEADER WORD ?
6436 023600 032703 010000          BIT     #FMT16,R3    ;NO !!
6437 023604 001034          BNE    155$          ;IS THIS A FORMAT ERROR ??
6438 023606 032703 140000          BIT     #MSE!USE,R3  ;YES !!
6439 023612 001056          BNE    165$          ;IS THIS A BAD SECTOR ERROR ??
6440 023614          ;YES !!
6441
6442          ;VERIFY THAT A HEADER COMPARE ERROR WAS DETECTED
6443 023614 032737 000200 001344          BIT     #HCE,RMER11  ;WAS HCE DETECTED ??
  
```

```

6444 023622 001077          BNE      175$          ;YES !!
6445          ;HCE WAS NOT DETECTED - SEE IF THERE WAS ANOTHER ERROR
6446 023624 004737 057036  JSR      PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
6447 023630 000405          BR       150$          ;GO TO 150$ IF NO ERROR
6448 023632 000240          NOP              ;RETURN HERE IF ERROR
6449 023634 104000          ERROR         ;ERROR # DEFINED BY DTASTS SUBROUTINE
6450 023636 004736          JSR      PC,(SP)+      ;GO BACK FOR MORE ERROR CHECKS
6451 023640 000137 024536  JMP      350$          ;GO TO 350$ IF ERROR WAS FOUND
6452 023644
6453 023644 013737 001344 001142 150$:  MOV      RMER1I,$BDDAT ;RECEIVED STATUS
6454 023652 012737 000200 001140  MOV      #HCE,$GDDAT  ;EXPECTED STATUS
6455 023660 010437 001174          MOV      R4,$TMPD     ;R4 = HEADER WORD NUMBER
6456 023664 010337 001176          MOV      R3,$TMP1     ;R3 = BIT POSIITON
6457 023670 104344          ERROR         344    ;HCE NOT DETECTED
6458 023672 000137 024536  JMP      350$
6459 023676
6460
6461          ;VERIFY THAT A FORMAT ERROR WAS DETECTED
6462 023676 032737 000020 001344  BIT      #FER,RMER1I  ;WAS FER DETECTED ??
6463 023704 001046          BNE      175$          ;YES !!
6464          ;FER WAS NOT DETECTED - SEE IF THERE WAS ANOTHER ERROR
6465 023706 004737 057036  JSR      PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
6466 023712 000405          BR       160$          ;GO TO 160$ IF NC ERROR
6467 023714 000240          NOP              ;RETURN HERE IF ERROR
6468 023716 104000          ERROR         ;ERROR # DEFINED BY DTASTS SUBROUTINE
6469 023720 004736          JSR      PC,(SP)+      ;GO BACK FOR MORE ERROR CHECKS
6470 023722 000137 024536  JMP      350$          ;GO TO 350$ IF ERROR WAS FOUND
6471 023726
6472 023726 012737 000020 001140 160$:  MOV      #FER,$GDDAT  ;EXPECTED STATUS
6473 023734 013737 001344 001142  MOV      RMER1I,$BDDAT ;RECEIVED STATUS
6474 023742 104343          ERROR         343    ;FER NOT DETECTED
6475 023744 000137 024536  JMP      350$
6476 023750
6477
6478          ;VERIFY THAT A BAD SECTOR ERROR WAS DETECTED
6479 023750 032737 100000 001372  BIT      #BSE,RMER2I  ;WAS BSE DETECTED ??
6480 023756 001021          BNE      175$          ;YES !!
6481          ;BSE WAS NOT DETECTED - SEE IF THERE WAS ANOTHER ERROR
6482 023760 004737 057036  JSR      PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
6483 023764 000405          BR       170$          ;GO TO 170$ IF NO ERROR
6484 023766 000240          NOP              ;RETURN HERE IF ERROR
6485 023770 104000          ERROR         ;ERROR # DEFINED BY DTASTS SUBROUTINE
6486 023772 004736          JSR      PC,(SP)+      ;GO BACK FOR MORE ERROR CHECKS
6487 023774 000137 024536  JMP      350$          ;GO TO 350$ IF ERROR WAS FOUND
6488 024000
6489 024000 013737 001372 001142 170$:  MOV      RMER2I,$BDDAT ;RECEIVED STATUS
6490 024006 012737 100000 001140  MOV      #BSE,$GDDAT  ;EXPECTED STAIUS
6491 024014 104345          ERROR         345    ;BSE NOTDETECTED
6492 024016 000137 024536  JMP      350$
6493 024022
6494 024022 004737 045364 175$:  JSR      PC,SECERR     ;GO CHECK FOR SECONDARY ERRORS
6495 024026 000405          BR       180$          ;GO TO 180$ IF NO ERROR
6496 024030 000240          NOP              ;RETURN HERE IF ERROR
6497 024032 104000          ERROR         ;ERROR # DEFINED BY SECERR SUBROUTINE
6498 024034 004736          JSR      PC,(SP)+      ;GO BACK FOR MORE ERROR CHECKS
6499 024036 000137 024536  JMP      350$          ;GO TO 350$ IF ERROR WAS FOUND

```

M10

CZRMEBO RM03/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 129
T16 WRITE, READ W/ HCE ERROR

SEG 0129

```

6500 024042          180$:
6501
6502                ;CHANGE LOOP ADDRESSES
6503 024042 012737 024056 001122      MOV    #190$, $LPADR
6504 024050 012737 024056 001124      MOV    #190$, $LPERR
6505
6506                ;*****
6507                ;LOOP #3      READ
6508
6509 024056          190$:
6510
6511                ;PREPARE DEVICE FOR READ OPERATION
6512 024056 004737 040020      JSR    PC, TSTPRP      ;PREPARE DEVICE FOR TEST
6513 024062 054130              .WORD   054130      ;TASK DESCRIPTOR
6514 024064 000404              BR     200$           ;GO TO 200$ IF NO ERROR
6515 024066 000240              NOP                    ;RETURN HERE IF ERROR
6516 024070 104000              ERROR          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
6517 024072 000137 024536      JMP    350$           ;GO TO 350$ IF ERROR WAS FOUND
6518 024076
6519
6520                200$:
6521                240$:
6522                ;READ DATA FROM DEVICE
6523 024076 012737 110560 001404      MOV    #BUFTWO+4, RMBAO      ;CHANGE BUS ADDRESS
6524 024104 012737 177400 001402      MOV    #(<C256.+1>), RMCWCO  ;CHANGE WORD COUNT
6525 024112 012737 000071 001400      MOV    #RD!GO, RMC$10      ;READ DATA COMMAND
6526 024120 012702 001535              MOV    #PUTINX, R2         ;LOAD PUT REGISTER INDEX TABLE
6527 024124 112722 000006              MOVB   #RMDA, (R2)+
6528 024130 112722 000032              MOVB   #RMOF, (R2)+
6529 024134 112722 000034              MOVB   #RMDC, (R2)+
6530 024140 112722 000004              MOVB   #RMB, (R2)+
6531 024144 112722 000002              MOVB   #RMC, (R2)+
6532 024150 112722 000000              MOVB   #RMC$1, (R2)+
6533 024154 112712 000200              MOVB   #200, (R2)
6534
6535 024160 004737 044006      JSR    PC, PUT      ;GO WRITE REGISTERS WITH PUT SUBROUTINE
6536 024164 000404              BR     250$           ;GO TO 250$ IF NO ERROR
6537 024166 000240              NOP                    ;RETURN HERE IF ERROR
6538 024170 104000              ERROR          ;ERROR # DEFINED BY PUT SUBROUTINE
6539 024172 000137 024536      JMP    350$           ;GO TO 350$ IF ERROR WAS FOUND
6540 024176
6541                250$:
6542                ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
6543 024176 004737 043452      JSR    PC, GETSTS      ;GO TO GETSTS SUBROUTINE
6544
6545                ;WAIT FOR READ OPERATION TO COMPLETE
6546 024202 004737 044346      JSR    PC, TIMEOUT     ;GO TO TIMEOUT SUBROUTINE
6547
6548                ;GO READ STATUS FOR READ OPERATION
6549 024206 004737 043536      JSR    PC, GET      ;GO READ REGISTERS WITH GET SUBROUTINE
6550 024212 000404              BR     260$           ;GO TO 260$ IF NO ERROR
6551 024214 000240              NOP                    ;RETURN HERE IF ERROR
6552 024216 104000              ERROR          ;ERROR # DEFINED BY GET SUBROUTINE
6553 024220 000137 024536      JMP    350$           ;GO TO 350$ IF ERROR WAS FOUND
6554 024224
6555                260$:

```

```

6556 ;CHECK FOR ERRORS DURING READ OPERATION
6557 024224 004737 044532 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
6558 024230 000405 BR 265$ ;GO TO 265$ IF NO ERROR
6559 024232 000240 NOP ;RETURN HERE IF ERROR
6560 024234 104000 ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
6561 024236 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
6562 024240 000137 024536 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
6563 024244
6564
6565 024244 005704 ;DECODE THE TYPE OF ERROR PRODUCED BY THE BIT POSITION OF R3
6566 024246 001006 YST R4 ;IS THIS THE FIRST HEADER WORD ?
6567 024250 032703 010000 BNE 266$ ;NO !!
6568 024254 001034 BIT #FMT16,R3 ;IS THIS A FORMAT ERROR ??
6569 024256 032703 140000 BNE 275$ ;YES !!
6570 024262 001056 BIT #MSE!USE,R3 ;IS THIS A BAD SECTOR ERROR ??
6571 024264 BNE 285$ ;YES !!
6572
6573
6574 024264 032737 000200 001344 ;VERIFY THAT A HEADER COMPARE ERROR WAS DETECTED
6575 024272 001077 BIT #HCE,RMER1I ;WAS HCE DETECTED ??
6576 BNE 295$ ;YES !!
6577 024274 004737 057036 ;HCE WAS NOT DETECTED - SEE IF THERE WAS ANOTHER ERROR
6578 024300 000405 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
6579 024302 000240 BR 270$ ;GO TO 270$ IF NO ERROR
6580 024304 104000 NOP ;RETURN HERE IF ERROR
6581 024306 004736 ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
6582 024310 000137 024536 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
6583 024314 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
6584 024314 013737 001344 001142 270$: MOV RMER1I,$BDDAT ;RECEIVED STATUS
6585 024322 012737 000200 001140 MOV #HCE,$GDDAT ;EXPECTED STATUS
6586 024330 010437 001174 MOV R4,$TMP0 ;R4 = HEADER WORD NUMBER
6587 024334 010337 001176 MOV R3,$TMP1 ;R3 = BIT POSITION
6588 024340 104344 ERROR 344 ;HCE NOT DETECTED
6589 024342 000137 024536 JMP 350$
6590 024346
6591
6592
6593 024346 032737 000020 001344 ;VERIFY THAT A FORMAT ERROR WAS DETECTED
6594 024354 001046 BIT #FER,RMER1I ;WAS FER DETECTED ??
6595 BNE 295$ ;YES !!
6596 024356 004737 057036 ;FER WAS NOT DETECTED - SEE IF THERE WAS ANOTHER ERROR
6597 024362 000405 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
6598 024364 000240 BR 280$ ;GO TO 280$ IF NO ERROR
6599 024366 104000 NOP ;RETURN HERE IF ERROR
6600 024370 004736 ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
6601 024372 000137 024536 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
6602 024376 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
6603 024376 012737 000020 001140 280$: MOV #FER,$GDDAT ;EXPECTED STATUS
6604 024404 013737 001344 001142 MOV RMER1I,$BDDAT ;RECEIVED STATUS
6605 024412 104343 ERROR 343 ;FER NOT DETECTED
6606 024414 000137 024536 JMP 350$
6607 024420
6608
6609
6610 024420 032737 100000 001372 ;VERIFY THAT A BAD SECTOR ERROR WAS DETECTED
6611 024426 001021 BIT #BSE,RMER2I ;WAS BSE DETECTED ??
BNE 295$ ;YES !!

```

```

;BSE WAS NOT DETECTED - SEE IF THERE WAS ANOTHER ERROR
6612          ;BSE WAS NOT DETECTED - SEE IF THERE WAS ANOTHER ERROR
6613 024430 004737 057036          JSR    PC,DTASTS          ;GO VERIFY RESULTS OF DATA TRANSFER
6614 024434 000405                   BR     290$              ;GO TO 290$ IF NO ERROR
6615 024436 000240                   NOP                                ;RETURN HERE IF ERROR
6616 024440 104000                   ERROR # DEFINED BY DTASTS SUBROUTINE
6617 024442 004736                   JSR    PC,(SP)+          ;GO BACK FOR MORE ERROR CHECKS
6618 024444 000137 024536          JMP    350$              ;GO TO 350$ IF ERROR WAS FOUND
6619 024450
6620 024450 013737 001372 001142 290$: MOV    RMERZI,$BDDAT          ;RECEIVED STATUS
6621 024456 012737 100000 001140  MOV    #BSE,$GDDAT          ;EXPECTED STATUS
6622 024464 104345                   ERROR 345                 ;BSE NOTDETECTED
6623 024466 000137 000350          JMP    350
6624 024472
6625 024472 004737 045364          JSR    PC,SECERR          ;GO CHECK FOR SECONDARY ERRORS
6626 024476 000405                   BR     300$              ;GO TO 300$ IF NO ERROR
6627 024500 000240                   NOP                                ;RETURN HERE IF ERROR
6628 024502 104000                   ERROR # DEFINED BY SECERR SUBROUTINE
6629 024504 004736                   JSR    PC,(SP)+          ;GO BACK FOR MORE ERROR CHECKS
6630 024506 000137 024536          JMP    350$              ;GO TO 350$ IF ERROR WAS FOUND
6631 024512
6632
6633 024512 006203                   ASR    R3                  ;SHIFT TO NEXT BIT
6634 024514 001006                   BNE    310$              ;REPEAT IF NOT DONE
6635 024516 005704                   TST    R4                  ;SECOND HEADER WORD DONE??
6636 024520 001007                   BNE    355$              ;YES!!
6637 024522 012704 000002          MOV    #2,R4              ;SETUP FOR SECOND HEADER WORD
6638
6639
6640 024526 012703 020000          ;*****
6641 024532 000137 023100          ;NOTE: BSE NOT TESTED
6642 024532 000137 023100          MOV    #BIT13,R3
6643 024536
6644 024536 000472          JMP    15$
6645 024540
6646 024540 013737 001476 001434 310$: MOV    ASNDC,RMDCO          ;EXIT IF ERROR
6647 024546 013737 001500 001406  MOV    ASNDA,RMDAO          ;
6648 024554 012737 107550 001404  MOV    #BUFONE,RMBAO        ;BUFFER ADDRESS,REFORMAT THE SECTOR
6649 024562 012737 177776 001402  MOV    #-2,RMWC0           ;ONLY TWO HEAD WORDS
6650 024570 012737 010000 001432  MOV    #FM16,RMOFO         ;ALWAYS IN 16 BITS MODE
6651 024576 012737 000062 001400  MOV    #WH,RMCS10          ;WRITE HEAD AND DATA COMMAND
6652 024604 004737 042640          JSR    PC,GENBUF          ;SET UP THE BUFFER
6653
6654 024610 004737 040020          JSR    PC,TSTPRP          ;PREPARE DEVICE FOR TEST
6655 024614 054130 ;TASK DESCRIPTOR
6656 024616 000404                   BR     360$              ;GO TO 360$ IF NO ERROR
6657 024620 000240                   NOP                                ;RETURN HERE IF ERROR
6658 024622 104000                   ERROR # DEFINED BY TSTPRP SUBROUTINE
6659 024624 000137 024630          JMP    360$              ;GO TO 360$ IF ERROR WAS FOUND
6660 024630
6661 024630 012737 000063 001400 360$: MOV    #WH!GO,RMCS10        ;FORMAT THE SECTOR
6662 024636 012702 001535          MOV    #PUTINX,R2         ;SET UP THE REGS
6663 024642 112722 000006          MOVB  #RMDA,(R2)+
6664 024646 112722 000032          MOVB  #RMOF,(R2)+
6665 024652 112722 000034          MOVB  #RMDC,(R2)+
6666 024656 112722 000004          MOVB  #RMBA,(R2)+
6667 024662 112722 000002          MOVB  #RMWC,(R2)+

```

```

6668 024666 112722 000000      MOVB    #RMCS1,(R2)+      ;
6669 024672 112722 000200      MOVB    #200,(R2)+      ;
6670 024676 004737 044006      JSR     PC,PUT          ;GO WRITE REGISTERS VIA PUT SUB
6671 024702 000404              BR      365$           ;GO TO 365$ IF NO ERROR
6672 024704 000240              NOP                    ;RETURN HERE IF ERROR
6673 024706 104000              ERROR                 ;ERROR DEFINED BY PUT SUB
6674 024710 000137 024724      JMP     370$           ;GO TO 370$ IF ERROR WAS FOUND
6675 024714 000240      365$:  NOP                    ;WAIT FOR FINISH
6676 024716 004737 044346      JSR     PC,TIMOUT
6677 024722 000240              NOP
6678 024724 012737 023074 001122 370$:  MOV     #10$,$LPADR      ;CHANGE LOOP TO START OF TEST
6679 024732 012737 023074 001124      MOV     #10$,$LPERR
;*****
; *TEST 17 WRITE, READ W/ HCI
;*****
TST17:
6683 024740              SCOPE                 ;SCOPE CALL
6684 024740 000004              NOP                    ;START OF TEST
6685 024742 000240              MOV     #STACK,SP      ;INITIALIZE STACK POINTER
6686 024744 012706 001100      MOV     $BASE,R0       ;R0=UNIBUS ADDRESS
6687 024750 013700 001276      MOV     TSTQUE,R1      ;(R1) = DEVICE BEING TESTED
6688 024754 013701 001450      MOV     #17,$TESTN     ;; SET TEST NUMBER IN APT MAIL BOX
6689 024760 012737 000017 001226
;*****
;LOOP #1      FORMAT,WRITE,READ
6694 024766 012704 000000 10$:  MOV     #0,R4           ;HEADER WORD
6695 024772 012703 000001      MOV     #1,R3           ;HCE BIT
6696
6697 024776      15$:
6698 ;PREPARE THE DEVICE FOR FORMAT OPERATION
6699 024776 004737 040020      JSR     PC,TSTPRP      ;PREPARE DEVICE FOR TEST
6700 025002 054130      .WORD  054130 ;TASK DESCRIPTOR
6701 025004 000404              BR      20$           ;GO TO 20$ IF NO ERROR
6702 025006 000240              NOP                    ;RETURN HERE IF ERROR
6703 025010 104000              ERROR                 ;ERROR # DEFINED BY TSTPRP SUBROUTINE
6704 025012 000137 026400      JMP     370$           ;GO TO 370$ IF ERROR WAS FOUND
6705 025016      20$:
6706
6707 ;LOAD PARAMETERS AND GENERATE DATA BUFFER
6708 025016 012737 000000 001434      MOV     #0,RMDCO       ;CYLINDER = 0
6709 025024 012737 000000 001406      MOV     #0,RMDAO       ;TRACK = 0, SECTOR = 0
6710 025032 012737 107550 001404      MOV     #BUFONE,RMBAO  ;BUS ADDRESS
6711 025040 012737 177376 001402      MOV     #(<PC(2+256.)+1>),RMWCO ;WORD COUNT
6712 025046 012737 010000 001432      MOV     #FMT16,RMOFO   ;16 BIT FORMAT
6713 025054 012737 000063 001400      MOV     #WH!GO,RMCS10  ;WRITE HEADER AND DATA COMMAND
6714
6715 ;VERIFY SECTOR(S) NOT IN BAD SECTOR FILE
6716
6717 025062 004737 040734      JSR     PC,BADSCT      ;CALL BAD SECTOR MODULE
6718 025066 000405      BR      25$           ;GO TO 25$ IF NO ERROR
6719 025070 104401 067466      TYPE   ,SCTMSG        ;TYPE BAD SECTOR MESSAGE
6720 025074 104000              ERROR                 ;ERROR # DEFINED BY BADSCT SUBROUTINE
6721 025076 000137 026214      JMP     350$           ;GO TO 350$ IF ERROR WAS FOUND
6722 025102
6723 025102 012737 071046 001174 25$:  MOV     #ONES,$TMPO    ;STARTING ADDRESS OF PATTERN

```



```

6724 025110 012737 000001 001176      MOV      #1,STMP1      ;RANGE OF PATTERN
6725 025116 004737 042640      JSR      PC,GENBUF    ;GO GENERATE BUFFER FOR FORMAT
6726
6727      ;CHANGE HEADER WORD TO FORCE ERROR DURING WRITE
6728 025122 030364 107550      BIT      R3,BUFONE(R4) ;SET OR RESET BIT??
6729 025126 001403      BEQ      27$
6730 025130 040364 107550      BIC      R3,BUFONE(R4) ;RESET BIT
6731 025134 000402      BR       28$
6732 025136 050364 107550      27$:    BIS      R3,BUFONE(R4) ;SET BIT
6733 025142
6734      ;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATIION
6735 025142 012702 001535      MOV      #PUTINX,R2   ;R2 = BYTE ENTRY POSITION
6736 025146 112722 000034      MOVB     #RMDC,(R2)+
6737 025152 112722 000006      MOVB     #RMDA,(R2)+
6738 025156 112722 000004      MOVB     #RMBA,(R2)+
6739 025162 112722 000002      MOVB     #RMWC,(R2)+
6740 025166 112722 000032      MOVB     #RMOF,(R2)+
6741 025172 112722 000000      MOVB     #RMCS1,(R2)+
6742 025176 112712 000200      MOVB     #200,(R2)    ;TERMINATE TABLE
6743 025202
6744
6745      ;FORMAT THE DRIVE
6746 025202 004737 044006      JSR      PC,PUT      ;GO WRITE REGISTERS WITH PUT SUBROUTINE
6747 025206 000404      BR       40$        ;GO TO 40$ IF NO ERROR
6748 025210 000240      NOP
6749 025212 104000      ERROR   ;RETURN HERE IF ERROR
6750 025214 000137 026214      JMP      350$       ;ERROR # DEFINED BY PUT SUBROUTINE
6751 025220
6752
6753      ;SETUP GET REGISTER INDEX TABLE FOR READING STATUS
6754 025220 004737 043452      JSR      PC,GETSTS   ;GO TO GETSTS SUBROUTINE
6755
6756      ;WAIT FOR THE FORMAT TO COMPLETE
6757 025224 004737 044346      JSR      PC,TIMOUT   ;GO TO TIMOUT SUBROUTINE
6758
6759      ;GO READ STATUS FOR FORMAT OPERATION
6760 025230 004737 043536      JSR      PC,GET      ;GO READ REGISTERS WITH GET SUBROUTINE
6761 025234 000404      BR       50$        ;GO TO 50$ IF NO ERROR
6762 025236 000240      NOP
6763 025240 104000      ERROR   ;RETURN HERE IF ERROR
6764 025242 000137 026214      JMP      350$       ;ERROR # DEFINED BY GET SUBROUTINE
6765 025246
6766
6767      ;VERIFY NO ERRORS DURING FORMAT
6768 025246 004737 057036      JSR      PC,DTASTS   ;GO VERIFY RESULTS OF DATA TRANSFER
6769 025252 000405      BR       60$        ;GO TO 60$ IF NO ERROR
6770 025254 000240      NOP
6771 025256 104000      ERROR   ;RETURN HERE IF ERROR
6772 025260 004736      JSR      PC,(SP)+    ;ERROR # DEFINED BY DTASTS SUBROUTINE
6773 025262 000137 026214      JMP      350$       ;GO BACK FOR MORE ERROR CHECKS
6774 025266
6775
6776      ;MOVE LOOP ADDRESSES TO NEXT OPERATION
6777 025266 012737 025304 001122      MOV      #70$,SLPADR
6778 025274 012737 025304 001124      MOV      #70$,SLPERR
6779 025302 000410      BR       80$        ;SKIP TO WRITE OPERATION

```

```

6780
6781
6782
6783
6784 025304
6785
6786
6787
6788 025304 004737 040020
6789 025310 054130
6790 025312 000404
6791 025314 000240
6792 025316 104000
6793 025320 000137 026214
6794 025324
6795
6796
6797 025324
6798
6799
6800 025324 012737 012000 001432
6801 025332 012737 000061 001400
6802 025340 012737 107554 001404
6803 025346 012702 001535
6804 025352 112722 000006
6805 025356 112722 000034
6806 025362 112722 000032
6807 025366 112722 000004
6808 025372 112722 000002
6809 025376 112722 000000
6810 025402 112722 000200
6811
6812 025406 004737 044006
6813 025412 000404
6814 025414 000240
6815 025416 104000
6816 025420 000137 026214
6817 025424
6818
6819
6820 025424 004737 043452
6821
6822
6823 025430 004737 044346
6824
6825
6826 025434 004737 043536
6827 025440 000404
6828 025442 000240
6829 025444 104000
6830 025446 000137 026214
6831 025452
6832
6833
6834 025452 004737 044532
6835 025456 000405

```

```

*****
;LOOP #2 WRITE,READ
70$:
;PREPARE DEVICE FOR WRITE OPERATION
JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
WORD 054130 ;TASK DESCRIPTOR
BR 80$ ;GO TO 80$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR 104000 ;ERROR # DEFINED BY TSTPRP SUBROUTINE
JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
80$:
120$:
;WRITE DATA TO THE DRIVE
MOV #HCI!FMT16,RMOFO ;INHIBIT HEADER COMPARE
MOV #WD!GO,RMCS10 ;WRITE DATA COMMAND
MOV #BUFONE+4,RMBA0 ;LOAD STARTING BUFFER ADDRESS
MOV #PUTINX,R2 ;LOAD PUT REGISTER INDEX TABLE
MOVB #RMDA,(R2)+
MOVB #RMDC,(R2)+
MOVB #RMOF,(R2)+
MOVB #RMBA,(R2)+
MOVB #RMWC,(R2)+
MOVB #RMCS1,(R2)+
MOVB #200,(R2)+ ;TERMINATE TABLE
JSR PC,PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
BR 130$ ;GO TO 130$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR 104000 ;ERROR # DEFINED BY PUT SUBROUTINE
JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
130$:
;SETUP REGISTER INPUT BUFFER FOR READING STATUS
JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
;WAIT FOR WRITE COMMAND TO COMPLETE
JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
;GO READ STATUS FOR WRITE COMMAND
JSR PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
BR 140$ ;GO TO 140$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR 104000 ;ERROR # DEFINED BY GET SUBROUTINE
JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
140$:
;CHECK FOR ERRORS DURING WRITE OPERATION
JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
BR 150$ ;GO TO 150$ IF NO ERROR

```

```

6836 025460 000240      NOP      ;RETURN HERE IF ERROR
6837 025462 104000      ERROR    ;ERROR # DEFINED BY PRIERR SUBROUTINE
6838 025464 004736      JSR      PC,2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
6839 025466 000137 026214      JMP      350$      ;GO TO 350$ IF ERROR WAS FOUND
6840 025472      150$:
6841 025472 032737 000220 001344      BIT      #HCE!FER,RMER1I ;ANY ERROR??
6842 025500 001407      BEQ      151$
6843 025502 013737 001344 001142      MOV      RMER1I,$BDDAT ;RECEIVED STATUS
6844 025510 042737 177557 001142      BIC      #↑C<HCE!FER>,$BDDAT ;CLEAR DONT CARES
6845 025516 000412      BR       152$
6846 025520 032737 100000 001372 151$:      BIT      #BSE,RMER2I ;ANY BAD SECTOR ERROR ??
6847 025526 001414      BEQ      155$      ;NO !!
6848 025530 013737 001372 001142      MOV      RMER2I,$BDDAT ;RECEIVED STATUS
6849 025536 042737 077777 001142      BIC      #↑BSE,$BDDAT ;CLEAR DONT CARES
6850 025544 012737 000000 001140 152$:      MOV      #0,$GDDAT ;EXPECTED STATUS
6851 025552 104346      ERROR    346
6852 025554 000137 026214      JMP      350$      ;HCE W/HCI SET
6853 025560      155$:
6854 025560 004737 057036      JSR      PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
6855 025564 000405      BR       160$      ;GO TO 160$ IF NO ERROR
6856 025566 000240      NOP
6857 025570 104000      ERROR    ;RETURN HERE IF ERROR
6858 025572 004736      JSR      PC,2(SP)+ ;ERROR # DEFINED BY DTASTS SUBROUTINE
6859 025574 000137 026214      JMP      350$      ;GO BACK FOR MORE ERROR CHECKS
6860 025600      160$:      ;GO TO 350$ IF ERROR WAS FOUND
6861
6862      170$:
6863 025600 004737 045364      JSR      PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
6864 025604 000405      BR       180$      ;GO TO 180$ IF NO ERROR
6865 025606 000240      NOP
6866 025610 104000      ERROR    ;RETURN HERE IF ERROR
6867 025612 004736      JSR      PC,2(SP)+ ;ERROR # DEFINED BY SECERR SUBROUTINE
6868 025614 000137 026214      JMP      350$      ;GO BACK FOR MORE ERROR CHECKS
6869 025620      180$:      ;GO TO 350$ IF ERROR WAS FOUND
6870
6871      ;CHANGE LOOP ADDRESSES
6872 025620 012737 025634 001122      MOV      #190$,$LPADR
6873 025626 012737 025634 001124      MOV      #190$,$LPERR
6874
6875      ;*****
6876      ;LOOP #3      READ
6877
6878 025634      190$:
6879
6880      ;PREPARE DEVICE FOR READ OPERATION
6881 025634 004737 040020      JSR      PC,TSTPRP ;PREPARE DEVICE FOR TEST
6882 025640 054130      .WORD   054130 ;TASK DESCRIPTOR
6883 025642 000404      BR       200$
6884 025644 000240      NOP      ;GO TO 200$ IF NO ERROR
6885 025646 104000      ERROR    ;RETURN HERE IF ERROR
6886 025650 000137 026214      JMP      350$      ;ERROR # DEFINED BY TSTPRP SUBROUTINE
6887 025654      200$:      ;GO TO 350$ IF ERROR WAS FOUND
6888
6889 025654      240$:
6890
6891      ;READ DATA FROM DEVICE

```

```

6892 025654 012737 000071 001400 MOV #RD!GO,RMCS10 ;READ DATA COMMAND
6893 025662 012737 110560 001404 MOV #BUFTW0+4,RMBA0 ;LOAD STARTING BUFFER ADDRESS
6894 025670 012702 001535 MOV #PUTINX,R2 ;LOAD PUT REGISTER INDEX TABLE
6895 025674 112722 000006 MOVB #RMDA,(R2)+
6896 025700 112722 000032 MOVB #RM0F,(R2)+
6897 025704 112722 000034 MOVB #RMDC,(R2)+
6898 025710 112722 000004 MOVB #RMBA,(R2)+
6899 025714 112722 000002 MOVB #RMWC,(R2)+
6900 025720 112722 000000 MOVB #RMCS1,(R2)+
6901 025724 112712 000200 MOVB #200,(R2)
6902
6903 025730 004737 044006 JSR PC,PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
6904 025734 000404 BR 250$ ;GO TO 250$ IF NO ERROR
6905 025736 000240 NOP ;RETURN HERE IF ERROR
6906 025740 104000 ERROR ;ERROR # DEFINED BY PUT SUBROUTINE
6907 025742 000137 026214 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
6908 025746
6909
6910 ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
6911 025746 004737 043452 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
6912
6913 ;WAIT FOR READ OPERATION TO COMPLETE
6914 025752 004737 044346 JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
6915
6916 ;GO READ STATUS FOR READ OPERATION
6917 025756 004737 043536 JSR PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
6918 025762 000404 BR 260$ ;GO TO 260$ IF NO ERROR
6919 025764 000240 NOP ;RETURN HERE IF ERROR
6920 025766 104000 ERROR ;ERROR # DEFINED BY GET SUBROUTINE
6921 025770 000137 026214 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
6922 025774
6923
6924 ;CHECK FOR ERRORS DURING READ OPERATION
6925 025774 004737 044532 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
6926 026000 000405 BR 270$ ;GO TO 270$ IF NO ERROR
6927 026002 000240 NOP ;RETURN HERE IF ERROR
6928 026004 104000 ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
6929 026006 004736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
6930 026010 000137 026214 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
6931 026014
6932 026014 032737 000220 001344 270$: BIT #HCE!FER,RMER1I ;ANY ERROR??
6933 026022 001407 BEQ 271$ ;NO!!
6934 026024 013737 001344 001142 MOV RMER1I,$BDDAT ;RECEIVED STATUS
6935 026032 042737 177557 001142 BIC #!C<HCE!FER>,$BDDAT ;CLEAR DONT CARES
6936 026040 000412 BR 272$
6937 026042 032737 100000 001372 271$: BIT #BSE,RMER2I ;ANY BAD SECTOR ERROR ??
6938 026050 001414 BEQ 275$ ;NO !!
6939 026052 013737 001372 001142 MOV RMER2I,$BDDAT ;RECEIVED STATUS
6940 026060 042737 077777 001142 BIC #!CBSE,$BDDAT ;CLEAR DONT CARES
6941 026066 012737 000000 001140 272$: MOV #0,$GDDAT ;EXPECTED STATUS
6942 026074 104346 ERROR 346 ;HCE W/HCI SET
6943 026076 000137 026214 JMP 350$
6944 026102
6945 026102 004737 057036 275$: JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
6946 026106 000405 BR 280$ ;GO TO 280$ IF NO ERROR
6947 026110 000240 NOP ;RETURN HERE IF ERROR

```

```

6948 026112 104000          ERROR
6949 026114 004736          JSR      PC,(SP)+
6950 026116 000137 026214    JMP      350$
6951 026122                280$:
6952
6953 026122                290$:
6954 026122 004737 045364    JSR      PC,SECERR
6955 026126 000405          BR       300$
6956 026130 000240          NOP
6957 026132 104000          ERROR
6958 026134 004736          JSR      PC,(SP)+
6959 026136 000137 026214    JMP      350$
6960 026142                300$:
6961 026142 004737 043104    JSR      PC,CMPBUF
6962 026146 107554          .WORD   BUFOFF+4
6963 026150 110560          .WORD   BUFTWO+4
6964 026152 000404          BR       310$
6965 026154 000240          NOP
6966 026156 104000          ERROR
6967 026160 000137 026214    JMP      350$
6968 026164                310$:
6969
6970 026164 006303          ASL     R3
6971 026166 001006          BNE     320$
6972 026170 005704          TST     R4
6973 026172 001011          BNE     355$
6974 026174 012703 000001    MOV     #1,R3
6975 026200 012704 000002    MOV     #2,R4
6976 026204                320$:
6977 026204 000137 024776    JMP     15$
6978 026210 000240          NOP
6979 026212 000240          NOP
6980 026214                350$:
6981 026214 000471          BR      370$
6982 026216                355$:
6983 026216 013737 001476 001434    MOV     ASNDC,RMDCO
6984 026224 013737 001500 001406    MOV     ASNDR,RMDAO
6985 026232 012737 107550 001404    MOV     #BUFOFF,RMBAO
6986 026240 012737 177776 001402    MOV     #-2,RMDCO
6987 026246 012737 010000 001432    MOV     #FMT16,RMFOFO
6988 026254 012737 000062 001400    MOV     #WH,RMCS10
6989 026262 004737 042640          JSR     PC,GENBUF
6990 026266 004737 040020          JSR     PC,TSTPRP
6991 026272 054130          .WORD   054130 ;TASK DESCRIPTOR
6992 026274 000404          BR      360$
6993 026276 000240          NOP
6994 026300 104000          ERROR
6995 026302 000137 026306    JMP     360$
6996 026306                360$:
6997 026306 012737 000063 001400    MOV     #WH!GO,RMCS10
6998 026314 012702 001535          MOV     #PUTINX,R2
6999 026320 112722 000006          MOVB   #RMDA,(R2)+
7000 026324 112722 000032          MOVB   #RMOF,(R2)+
7001 026330 112722 000034          MOVB   #RMDC,(R2)+
7002 026334 112722 000004          MOVB   #RMBA,(R2)+
7003 026340 112722 000002          MOVB   #RMWC,(R2)+

```

```

;ERROR # DEFINED BY DTASTS SUBROUTINE
;GO BACK FOR MORE ERROR CHECKS
;GO TO 350$ IF ERROR WAS FOUND

;GO CHECK FOR SECONDARY ERRORS
;GO TO 300$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY SECERR SUBROUTINE
;GO BACK FOR MORE ERROR CHECKS
;GO TO 350$ IF ERROR WAS FOUND

;GO COMPARE WRITE, READ DATA BUFFERS
;STARTING ADDRESS OF WRITE BUFFER
;STARTING ADDRESS OF READ BUFFER
;GO TO 310$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY CMPBUF SUBROUTINE
;GO TO 350$ IF ERROR WAS FOUND

;SHIFT HCE BIT
;CONTINUE IF NOT DONE
;SECOND HEADER DONE??
;YES!!
;START WITH BIT 0
;DO SECOND HEADER WORD

;EXIT IF ERROR

;BUFFER ADDRESS
;WORD COUNT
;IN 16 BIT MODE
;FORMAT COMMAND
;SET UP BUFFER
;PREPARE DEVICE FOR TEST
;TASK DESCRIPTOR
;GO TO 360$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 360$ IF ERROR WAS FOUND

;TABLE ADDRESS

```

```

7004 026344 112722 000000      MOVB    #RMCS1,(R2)+      ;
7005 026350 112722 000200      MOVB    #200,(R2)+      ;
7006 026354 004737 044006      JSR     PC,PUT          ;GO WRITE REGISTERS VIA PUT SUB
7007 026360 000404              BR      365$           ;GO TO 365$ IF NO ERROR
7008 026362 000240              NOP                    ;RETURN HERE IF ERROR
7009 026364 104000              ERROR                ;ERROR DEFINED BY PUT SUB
7010 026366 000137 026400      JMP     370$           ;GO TO 370$ IF ERROR WAS FOUND
7011 026372 000240      365$:  NOP
7012 026374 004737 044346      JSR     PC,TIMOUT      ;
7013 026400 012737 024766 001122 370$:  MOV     #10$,$LPADR    ;CHANGE LOOP TO START OF TEST
7014 026406 012737 024766 001124      MOV     #10$,$LPERR
7015                                     ;*****
7016                                     ;*TEST 20 WRITE, READ W/ IVC ERROR
7017                                     ;*****
7018                                     †TST20:
7019 026414 000004              SCOPE                ;SCOPE CALL
7020 026416 000240              NOP                    ;START OF TEST
7021 026420 012706 001100      MOV     #STACK,SP     ;INITIALIZE STACK POINTER
7022 026424 013700 001276      MOV     $BASE,R0      ;R0=UNIBUS ADDRESS
7023 026430 013701 001450      MOV     TSTQUE,R1     ;(R1) = DEVICE BEING TESTED
7024 026434 012737 000020 001226      MOV     #20,$TESTN    ;SET TEST NUMBER IN APT MAIL BOX
7025
7026                                     ;LOAD PARAMETERS AND GENERATE DATA BUFFER
7027 026442 012737 000000 001434      MOV     #0,RMDCO      ;CYLINDER = *
7028 026450 012737 000000 001406      MOV     #0,RMDAO      ;TRACK = *, SECTOR = *
7029 026456 012737 107550 001404      MOV     #BUFONE,RMBAO ;BUS ADDRESS
7030 026464 012737 177522 001402      MOV     #(<C256+1>),RMWCO ;WORD COUNT
7031 026472 012737 010000 001432      MOV     #FMT16,RMOFO  ;16 BIT FORMAT
7032 026500 012737 000060 001400      MOV     #WD,RMCS10    ;WRITE HEADER AND DATA COMMAND
7033
7034                                     ;SETUP GET REGISTER INDEX TABLE FOR READING STATUS
7035 026506 004737 043452      JSR     PC,GETSTS     ;GO TO GETSTS SUBROUTINE
7036
7037
7038                                     ;MOVE LOOP ADDRESSES TO NEXT OPERATION
7039 026512 012737 026530 001122      MOV     #70$,$LPADR
7040 026520 012737 026530 001124      MOV     #70$,$LPERR
7041 026526 000410              BR      80$           ;SKIP TO WRITE OPERATION
7042
7043                                     ;*****
7044                                     ;LOOP #2 WRITE,READ
7045
7046 026530      70$:
7047
7048
7049                                     ;PREPARE DEVICE FOR WRITE OPERATION
7050 026530 004737 040020      JSR     PC,TSTPRP     ;PREPARE DEVICE FOR TEST
7051 026534 054130              .WORD   054130        ;TASK DESCRIPTOR
7052 026536 000404              BR      80$           ;GO TO 80$ IF NO ERROR
7053 026540 000240              NOP                    ;RETURN HERE IF ERROR
7054 026542 104000              ERROR                ;ERROR # DEFINED BY TSTPRP SUBROUTINE
7055 026544 000137 027506      JMP     350$         ;GO TO 350$ IF ERROR WAS FOUND
7056 026550      80$:
7057
7058                                     ;RESET VOLUME VALID
7059 026550 112737 000024 001535      MOVB    #RMMR1,PUTINX ;SETUP PUT INDEX TABLE

```

```

7060 026556 112737 000200 001536      MOVB      #200,PUTINX+1      ;WRITE TERMINATOR BYTE
7061 026564 012737 000001 001424      MOV      #DMD,RMMR10      ;RMMR1=DMD
7062 026572 004737 044006      JSR      PC,PUT          ;GO WRITE RMMR1 VIA SUB
7063 026576 000404      BR       90$             ;GO TO 90$ IF NO ERROR
7064 026600 000240      NOP                      ;RETURN HERE IF ERROR
7065 026602 104000      ERROR   ;ERROR NUMBER DEFINED BY PUT SUB
7066 026604 000137 027506      JMP      350$           ;GO TO 350$ IF ERROR WAS FOUND
7067 026610
7068 026610 112737 000024 001535      MOVB      #RMMR1,PUTINX   ;SETUP PUT INDEX TABLE
7069 026616 112737 000200 001536      MOVB      #200,PUTINX+1   ;WRITE TERMINATOR BYTE
7070 026624 012737 000000 001424      MOV      #0,RMMR10       ;RMMR1=0
7071 026632 004737 044006      JSR      PC,PUT          ;GO WRITE RMMR1 VIA SUB
7072 026636 000404      BR       100$           ;GO TO 100$ IF NO ERROR
7073 026640 000240      NOP                      ;RETURN HERE IF ERROR
7074 026642 104000      ERROR   ;ERROR NUMBER DEFINED BY PUT SUB
7075 026644 000137 027506      JMP      350$           ;GO TO 350$ IF ERROR WAS FOUND
7076 026650
7077
7078 026650
7079
7080
7081 026650 012737 000061 001400      ;WRITE DATA TO THE DRIVE
7082 026656 012737 107554 001404      MOV      #WD!GO,RMCS10   ;WRITE DATA COMMAND
7083 026664 012702 001535      MOV      #BUFONE+4,RMBA0 ;LOAD STARTING BUFFER ADDRESS
7084 026670 112722 000006      MOV      #PUTINX,R2      ;LOAD PUT REGISTER INDEX TABLE
7085 026674 112722 000034      MOVB     #RMDA,(R2)+
7086 026700 112722 000032      MOVB     #RMDC,(R2)+
7087 026704 112722 000004      MOVB     #RMOF,(R2)+
7088 026710 112722 000002      MOVB     #RMBA,(R2)+
7089 026714 112722 000000      MOVB     #RMWC,(R2)+
7090 026720 112722 000200      MOVB     #RMCS1,(R2)+
7091
7092 026724 004737 044006      MOVB     #200,(R2)+      ;TERMINATE TABLE
7093 026730 000404      JSR      PC,PUT          ;GO WRITE REGISTERS WITH PUT SUBROUTINE
7094 026732 000240      BR       130$           ;GO TO 130$ IF NO ERROR
7095 026734 104000      NOP                      ;RETURN HERE IF ERROR
7096 026736 000137 027506      ERROR   ;ERROR # DEFINED BY PUT SUBROUTINE
7097 026742      JMP      350$           ;GO TO 350$ IF ERROR WAS FOUND
7098
7099
7100 026742 004737 043452      ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
7101      JSR      PC,GETSTS     ;GO TO GETSTS SUBROUTINE
7102
7103 026746 004737 044346      ;WAIT FOR WRITE COMMAND TO COMPLETE
7104      JSR      PC,TIMOUT   ;GO TO TIMEOUT SUBROUTINE
7105
7106 026752 004737 043536      ;GO READ STATUS FOR WRITE COMMAND
7107      JSR      PC,GET      ;GO READ REGISTERS WITH GET SUBROUTINE
7108 026756 000404      BR       140$           ;GO TO 140$ IF NO ERROR
7109 026760 000240      NOP                      ;RETURN HERE IF ERROR
7110 026762 104000      ERROR   ;ERROR # DEFINED BY GET SUBROUTINE
7111 026764 000137 027506      JMP      350$           ;GO TO 350$ IF ERROR WAS FOUND
7112
7113
7114 026770 004737 044532      ;CHECK FOR ERRORS DURING WRITE OPERATION
7115 026774 000405      JSR      PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
7116      BR       150$           ;GO TO 150$ IF NO ERROR

```

```

7116 026776 000240      NOP
7117 027000 104000      ERROR
7118 027002 004736      JSR    PC,2(SP)+
7119 027004 000137 027506  JMP    350$
7120 027010      150$:
7121 027010 032737 010000 001372  BIT    #IVC,RMER2I
7122 027016 001024      BNE    170$
7123 027020 004737 057036  JSR    PC,DTASTS
7124 027024 000405      BR     160$
7125 027026 000240      NOP
7126 027030 104000      ERROR
7127 027032 004736      JSR    PC,2(SP)+
7128 027034 000137 027506  JMP    350$
7129 027040      160$:
7130
7131 027040 013737 001372 001142  MOV    RMER2I,$BDDAT
7132 027046 013737 001372 001140  MOV    RMER2I,$GDDAT
7133 027054 052737 010000 001140  BIS    #IVC,$GDDAT
7134 027062 104342      ERROR
7135 027064 000137 027506  JMP    350$
7136 027070      170$:
7137 027070 004737 045364  JSR    PC,SECERR
7138 027074 000405      BR     180$
7139 027076 000240      NOP
7140 027100 104000      ERROR
7141 027102 004736      JSR    PC,2(SP)+
7142 027104 000137 027506  JMP    350$
7143 027110      180$:
7144
7145 ;CHANGE LOOP ADDRESSES
7146 027110 012737 027126 001122  MOV    #190$,SLPADR
7147 027116 012737 027126 001124  MOV    #190$,SLPERR
7148 027124 000410      BR     200$
7149
7150 ;*****
7151 ;LOOP #3      READ
7152
7153 027126      190$:
7154
7155 ;PREPARE DEVICE FOR READ OPERATION
7156 027126 004737 040020  JSR    PC,TSTPRP
7157 027132 054130  .WORD 054130 ;TASK DESCRIPTOR
7158 027134 000404      BR     200$
7159 027136 000240      NOP
7160 027140 104000      ERROR
7161 027142 000137 027506  JMP    350$
7162 027146      200$:
7163
7164 ;RESET VOLUME VALID
7165 027146 112737 000024 001535  MOVB  #RMMR1,PUTINX
7166 027154 112737 000200 001536  MOVB  #200,PUTINX+1
7167 027162 012737 000001 001424  MOV   #DMD,RMMR10
7168 027170 004737 044006  JSR   PC,PUT
7169 027174 000404      BR     210$
7170 027176 000240      NOP
7171 027200 104000      ERROR

```

```

;RETURN HERE IF ERROR
;ERROR # DEFINED BY PRIERR SUBROUTINE
;GO BACK FOR MORE ERROR CHECKS
;GO TO 350$ IF ERROR WAS FOUND

;WAS "IVC" DETECTED??
;YES!!
;GO VERIFY RESULTS OF DATA TRANSFER
;GO TO 160$ IF NO ERROR

;RETURN HERE IF ERROR
;ERROR # DEFINED BY DTASTS SUBROUTINE
;GO BACK FOR MORE ERROR CHECKS
;GO TO 350$ IF ERROR WAS FOUND

;RECEIVED STATUS
;EXPECTED STATUS
;IVC NOT DETECTED

;GO CHECK FOR SECONDARY ERRORS
;GO TO 180$ IF NO ERROR

;RETURN HERE IF ERROR
;ERROR # DEFINED BY SECERR SUBROUTINE
;GO BACK FOR MORE ERROR CHECKS
;GO TO 350$ IF ERROR WAS FOUND

;SKIP TO NEXT OPERATION

;*****
;LOOP #3      READ

;PREPARE DEVICE FOR TEST
;GO TO 200$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR # DEFINED BY TSTPRP SUBROUTINE
;GO TO 350$ IF ERROR WAS FOUND

;SETUP PUT INDEX TABLE
;WRITE TERMINATOR BYTE
;RMMR1=OMD
;GO WRITE RMMR1 VIA SUB
;GO TO 210$ IF NO ERROR
;RETURN HERE IF ERROR
;ERROR NUMBER DEFINED BY PUT SUB

```



```

7172 027202 000137 027506          JMP      350$          ;GO TO 350$ IF ERROR WAS FOUND
7173 027206          210$:
7174 027206 112737 000024 001535  MOVB    #RMMR1,PUTINX ;SETUP PUT INDEX TABLE
7175 027214 112737 000200 001536  MOVB    #200,PUTINX+1 ;WRITE TERMINATOR BYTE
7176 027222 012737 000000 001424  MOV     #0,RMMR10     ;RMMR1=0
7177 027230 004737 044006          JSR     PC,PUT        ;GO WRITE RMMR1 VIA SUB
7178 027234 000404          BR      220$         ;GO TO 220$ IF NO ERROR
7179 027236 000240          NOP                     ;RETURN HERE IF ERROR
7180 027240 104000          ERROR  ;ERROR NUMBER DEFINED BY PUT SUB
7181 027242 000137 027506          JMP     350$         ;GO TO 350$ IF ERROR WAS FOUND
7182 027246          220$:
7183 027246          240$:
7184
7185          ;READ DATA FROM DEVICE
7186 027246 012737 000071 001400  MOV     #RD!GO,RMCS10 ;READ DATA COMMAND
7187 027254 012737 110560 001404  MOV     #BUFTWO+4,RMBA0 ;LOAD STARTING BUFFER ADDRESS
7188 027262 012702 001535          MOV     #PUTINX,R2    ;LOAD PUT REGISTER INDEX TABLE
7189 027266 112722 000006          MOVB   #RMDA,(R2)+
7190 027272 112722 000032          MOVB   #RMOF,(R2)+
7191 027276 112722 000034          MOVB   #RMDC,(R2)+
7192 027302 112722 000004          MOVB   #RMBA,(R2)+
7193 027306 112722 000002          MOVB   #RMWC,(R2)+
7194 027312 112722 000000          MOVB   #RMCS1,(R2)+
7195 027316 112712 000200          MOVB   #200,(R2)
7196
7197 027322 004737 044006          JSR     PC,PUT        ;GO WRITE REGISTERS WITH PUT SUBROUTINE
7198 027326 000404          BR      250$         ;GO TO 250$ IF NO ERROR
7199 027330 000240          NOP                     ;RETURN HERE IF ERROR
7200 027332 104000          ERROR  ;ERROR # DEFINED BY PUT SUBROUTINE
7201 027334 000137 027506          JMP     350$         ;GO TO 350$ IF ERROR WAS FOUND
7202 027340          250$:
7203
7204          ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
7205 027340 004737 043452          JSR     PC,GETSTS    ;GO TO GETSTS SUBROUTINE
7206
7207          ;WAIT FOR READ OPERATION TO COMPLETE
7208 027344 004737 044346          JSR     PC,TIMOUT    ;GO TO TIMOUT SUBROUTINE
7209
7210          ;GO READ STATUS FOR READ OPERATION
7211 027350 004737 043536          JSR     PC,GET        ;GO READ REGISTERS WITH GET SUBROUTINE
7212 027354 000404          BR      260$         ;GO TO 260$ IF NO ERROR
7213 027356 000240          NOP                     ;RETURN HERE IF ERROR
7214 027360 104000          ERROR  ;ERROR # DEFINED BY GET SUBROUTINE
7215 027362 000137 027506          JMP     350$         ;GO TO 350$ IF ERROR WAS FOUND
7216 027366          260$:
7217
7218          ;CHECK FOR ERRORS DURING READ OPERATION
7219 027366 004737 044532          JSR     P,PRIERR     ;GO CHECK FOR PRIMARY ERRORS
7220 027372 000405          BR      270$         ;GO TO 270$ IF NO ERROR
7221 027374 000240          NOP                     ;RETURN HERE IF ERROR
7222 027376 104000          ERROR  ;ERROR # DEFINED BY PRIERR SUBROUTINE
7223 027400 004736          JSR     PC,ISP'+     ;GO BACK FOR MORE ERROR CHECKS
7224 027402 000137 027506          JMP     350$         ;GO TO 350$ IF ERROR WAS FOUND
7225 027406          270$:
7226 027406 032737 010000 001372  BIT     #IVC,RMER2I   ;WAS "IVC" DETECTED?"
7227 027414 001024          BNE    290$         ;YES!!

```

M11

CZRMEBO RM03/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 142
T20 WRITE, READ W/ IVC ERROR

SEQ 0142

```

7228 027416 004737 057036 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
7229 027422 000405 BR 280$ ;GO TO 280$ IF NO ERROR
7230 027424 000240 NOP ;RETURN HERE IF ERROR
7231 027426 104000 ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
7232 027430 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
7233 027432 000137 027506 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
7234 027436 013737 001372 001140 280$: MOV RMER2I,$GDDAT ;EXPECTED STATUS
7235 027436 052737 010000 001140 BIS #IVC,$GDDAT
7236 027444 013737 001372 001142 MOV RMER2I,$BDDAT ;RECEIVED STATUS
7237 027452 104342 ERROR 342 ;IVC NOT DETECTED
7238 027460 000137 027506 JMP 350$
7239 027462 000137 027506
7240
7241 027466 290$: JSR PC,SECERP ;GO CHECK FOR SECONDARY ERRORS
7242 027466 004737 045364 BR 300$ ;GO TO 300$ IF NO ERROR
7243 027472 000405 NOP ;RETURN HERE IF ERROR
7244 027474 000240 ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
7245 027476 104000 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
7246 027500 004736 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
7247 027502 000137 027506
7248 027506
7249
7250 027506 350$: MOV #70$,$LPADR ;CHANGE LOOP TO START OF TEST
7251 027506 012737 026530 001122 MOV #70$,$LPERR
7252 027514 012737 026530 001124
7253 ;*****
7254 ;*TEST 21 WRITE, READ W/ ABORT
7255 ;*****
7256 ;TST21:
7257 027522 000004 SCOPE ;SCOPE CALL
7258 027524 000240 NOP ;START OF TEST
7259 027526 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
7260 027532 013700 001276 MOV $BASE,RO ;RO=UNIBUS ADDRESS
7261 027536 013701 001450 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
7262 027542 012737 000021 001226 MOV #21,$TSTN ;;SET TEST NUMBER IN APT MAIL BOX
7263
7264
7265 ;LOAD PARAMETERS AND GENERATE DATA BUFFER
7266 027550 012737 000000 001434 MOV #0,RMDCO ;CYLINDER = 0
7267 027556 012737 000000 001406 MOV #0,RMDAO ;TRACK = 0, SECTOR = 0
7268 027564 012737 107550 001404 MOV #BUFONE,RMBAO ;BUS ADDRESS
7269 027572 012737 177400 001402 MOV #(<C256.+1>,RMWCO ;WORD COUNT
7270 027600 012737 010000 001432 MOV #FMT16,RMOF0 ;16 BIT FORMAT
7271 027606 012737 000063 001400 MOV #WH!GO,RMCS10 ;WRITE HEADER AND DATA COMMAND
7272
7273 ;SETUP GET REGISTER INDEX TABLE FOR READING STATUS
7274 027614 004737 043452 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
7275
7276 ;MOVE LOOP ADDRESSES TO NEXT OPERATION
7277 027620 012737 027636 001122 MOV #70$,$LPADR
7278 027626 012737 027636 001124 MOV #70$,$LPERR
7279 027634 000410 BR 80$ ;SKIP TO WRITE OPERATION
7280
7281 ;*****
7282 ;LOOP #2 WRITE,READ
7283

```

```

7284 027636
7285
7286
7287
7288 027636 004737 040020
7289 027642 054130
7290 027644 000404
7291 027646 000240
7292 027650 104000
7293 027652 000137 030470
7294 027656
7295
7296
7297 027656 112737 000014 001535
7298 027664 112737 000200 001536
7299 027672 012737 040000 001414
7300 027700 004737 044006
7301 027704 000404
7302 027706 000240
7303 027710 104000
7304 027712 000137 030470
7305 027716
7306
7307 027716
7308
7309
7310 027716 012737 000061 001400
7311 027724 012702 001535
7312 027730 112722 000006
7313 027734 112722 000034
7314 027740 112722 000032
7315 027744 112722 000004
7316 027750 112722 000002
7317 027754 112722 000000
7318 027760 112722 000200
7319
7320
7321
7322 027764 004737 043452
7323 027770 004737 044006
7324 027774 000404
7325 027776 000240
7326 030000 104000
7327 030002 000137 030470
7328 030006
7329 030006 004737 043536
7330 030012 000404
7331 030014 000240
7332 030016 104000
7333 030020 000137 030470
7334 030024 032737 020000 001342
7335 030032 001412
7336 030034 013737 001342 001140
7337 030042 042737 020000 001140
7338 030050 013737 001342 001142
7339 030056 104347

70$:
;PREPARE DEVICE FOR WRITE OPERATION
JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
;WORD 054130 ;TASK DESCRIPTOR
BR 80$ ;GO TO 80$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND

80$:
;SET UNSAFE ERROR
MOVB #RMER1,PUTINX ;SETUP PUT INDEX TABLE
MOVB #200,PUTINX+1 ;WRITE TERMINATOR BYTE
MOV #UNS,RMER10 ;RMER1 OUTPUT BUFFER = UNS
JSR PC,PUT ;WRITE RMER1 VIA PUT SUB
BR 90$ ;GO TO 90$ IF NO ERROR
NOP ;RETURN HERE TO REPORT ERROR
ERROR ;ERROR NUMBER DEFINED BY PUT SUB
JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND

90$:
120$:
;WRITE DATA TO THE DRIVE
MOV #WD!GO,RMCS10 ;WRITE DATA COMMAND
MOV #PUTINX,R2 ;LOAD PUT REGISTER INDEX TABLE
MOVB #RMDA,(R2)+
MOVB #RMDC,(R2)+
MOVB #RMOF,(R2)+
MOVB #RMBA,(R2)+
MOVB #RMWC,(R2)+
MOVB #RMCS1,(R2)+
MOVB #200,(R2)+ ;TERMINATE TABLE

;SETUP REGISTER INPUT BUFFER FOR READING STATUS
JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
JSR PC,PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
BR 130$ ;GO TO 130$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY PUT SUBROUTINE
JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND

130$:
JSR PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
BR 135$ ;GO TO 135$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY GET SUBROUTINE
JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND

135$:
BIT #PIP,RMDSI ;WAS COMMAND EXECUTED?
BEQ 136$
MOV RMDSI,$GDDAT ;EXPECTED STATUS
BIC #PIP,$GDDAT
MOV RMDSI,$BDDAT ;RECEIVED STATUS
ERROR 347 ;NO ABORT

```

```

7340 030060          136$:
7341
7342                ;WAIT FOR WRITE COMMAND TO COMPLETE
7343 030060 004737 044346      JSR      PC,TIMOUT      ;GO TO TIMEOUT SUBROUTINE
7344
7345                ;GO READ STATUS FOR WRITE COMMAND
7346 030064 004737 043536      JSR      PC,GET      ;GO READ REGISTERS WITH GET SUBROUTINE
7347 030070 000404              BR      140$      ;GO TO 140$ IF NO ERROR
7348 030072 000240              NOP                      ;RETURN HERE IF ERROR
7349 030074 104000              ERROR          ;ERROR # DEFINED BY GET SUBROUTINE
7350 030076 000137 030470      JMP      350$      ;GO TO 350$ IF ERROR WAS FOUND
7351 030102
7352
7353                ;CHECK FOR ERRORS DURING WRITE OPERATION
7354 030102 004737 044532      JSR      PC,PRIERR    ;GO CHECK FOR PRIMARY ERRORS
7355 030106 000405              BR      150$      ;GO TO 150$ IF NO ERROR
7356 030110 000240              NOP                      ;RETURN HERE IF ERROR
7357 030112 104000              ERROR          ;ERROR # DEFINED BY PRIERR SUBROUTINE
7358 030114 074736              JSR      PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
7359 030116 000137 030470      JMP      350$      ;GO TO 350$ IF ERROR WAS FOUND
7360 030122
7361
7362                ;GO CHECK FOR SECONDARY ERRORS
7363 030122 004737 045364      JSR      PC,SECERR    ;GO CHECK FOR SECONDARY ERRORS
7364 030126 000405              BR      180$      ;GO TO 180$ IF NO ERROR
7365 030130 000240              NOP                      ;RETURN HERE IF ERROR
7366 030132 104000              ERROR          ;ERROR # DEFINED BY SECERR SUBROUTINE
7367 030134 004736              JSR      PC,@(SP)+    ;GO BACK FOR MORE ERROR CHECKS
7368 030136 000137 030470      JMP      350$      ;GO TO 350$ IF ERROR WAS FOUND
7369 030142
7370
7371                ;CHANGE LOOP ADDRESSES
7372 030142 012737 030160 001122  MOV      #190$,SLPADR
7373 030150 012737 030160 001124  MOV      #190$,SLPERR
7374 030156 000410              BR      200$      ;SKIP TO NEXT OPERATION
7375
7376                ;*****
7377                ;LOOP #3      READ
7378 030160
7379
7380                ;PREPARE DEVICE FOR READ OPERATION
7381 030160 004737 040020      JSR      PC,TSTPRP    ;PREPARE DEVICE FOR TEST
7382 030164 054130              .WORD    054130      ;TASK DESCRIPTOR
7383 030166 000404              BR      200$      ;GO TO 200$ IF NO ERROR
7384 030170 000240              NOP                      ;RETURN HERE IF ERROR
7385 030172 104000              ERROR          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
7386 030174 000137 030470      JMP      350$      ;GO TO 350$ IF ERROR WAS FOUND
7387 030200
7388
7389                ;SET UNSAFE ERROR
7390 030200 112737 000014 001535  MOVB     #RMER1,PUTINX ;SETUP PUT INDEX TABLE
7391 030206 112737 000200 001536  MOVB     #200,PUTINX+1 ;WRITE TERMINATOR BYTE
7392 030214 012737 040000 001414  MOV      #UNS,RMER10   ;RMER1 OUTPUT BUFFER = UNS
7393 030222 004737 044006      JSR      PC,PUT      ;WRITE RMER1 VIA PUT SUB
7394 030226 000404              BR      210$      ;GO TO 210$ IF NO ERROR
7395 030230 000240              NOP                      ;RETURN HERE TO REPORT ERROR
7395 030232 104000              ERROR          ;ERROR NUMBER DEFINED BY PUT SUB

```

```

7396 030234 000137 030470          JMP      350$      ;GO TO 350$ IF ERROR WAS FOUND
7397 030240          210$:
7398
7399 030240          240$:
7400
7401          ;READ DATA FROM DEVICE
7402 030240 012737 000071 001400      MOV      #RD,GO, RMCS10      ;READ DATA COMMAND
7403 030246 012702 001535      MOV      #PUT,INX, R2      ;LOAD PUT REGISTER INDEX TABLE
7404 030252 112722 000006      MOV      #RMDA, (R2)+
7405 030256 112722 000032      MOV      #RMOF, (R2)+
7406 030262 112722 000034      MOV      #RMDC, (R2)+
7407 030266 112722 000004      MOV      #RMBA, (R2)+
7408 030272 112722 000002      MOV      #RMWC, (R2)+
7409 030276 112722 000000      MOV      #RMCS1, (R2)+
7410 030302 112712 000200      MOV      #200, (R2)
7411
7412          ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
7413 030306 004737 043452      JSR      PC,GETSTS      ;GO TO GETSTS SUBROUTINE
7414
7415 030312 004737 044006      JSR      PC,PUT      ;GO WRITE REGISTERS WITH PUT SUBROUTINE
7416 030316 000404          BR      250$      ;GO TO 250$ IF NO ERROR
7417 030320 000240          NOP          ;RETURN HERE IF ERROR
7418 030322 104000          ERROR      ;ERROR # DEFINED BY PUT SUBROUTINE
7419 030324 000137 030470      JMP      350$      ;GO TO 350$ IF ERROR WAS FOUND
7420 030330          250$:
7421
7422          ;SEE IF DEVICE STARTED COMMAND
7423 030330 004737 043536      JSR      PC,GET      ;GO READ REGISTERS WITH GET SUBROUTINE
7424 030334 000404          BR      255$      ;GO TO 255$ IF NO ERROR
7425 030336 000240          NOP          ;RETURN HERE IF ERROR
7426 030340 104000          ERROR      ;ERROR # DEFINED BY GET SUBROUTINE
7427 030342 000137 030470      JMP      350$      ;GO TO 350$ IF ERROR WAS FOUND
7428 030346 032737 020000 001342 255$:      BIT      #PIP, RMDSI      ;WAS COMMAND EXECUTED??
7429 030354 001414          BEQ      256$      ;NO!
7430 030356 013737 00 342 001140      MOV      RMDSI, $GDDAT      ;EXPECTED STATUS
7431 030364 042737 020000 001140      BIC      #PIP, $GDDAT
7432 030372 013737 001342 001142      MOV      RMDSI, $BDDAT      ;RECEIVED STATUS
7433 030400 104347          ERROR      347      ;NO ABORT
7434 030402 000137 030470      JMP      350$
7435 030406          256$:
7436
7437          ;WAIT FOR READ OPERATION TO COMPLETE
7438 030406 004737 044346      JSR      PC,TIMOUT      ;GO TO TIMOUT SUBROUTINE
7439
7440          ;GO READ STATUS FOR READ OPERATION
7441 030412 004737 043536      JSR      PC,GET      ;GO READ REGISTERS WITH GET SUBROUTINE
7442 030416 000404          BR      260$      ;GO TO 260$ IF NO ERROR
7443 030420 000240          NOP          ;RETURN HERE IF ERROR
7444 030422 104000          ERROR      ;ERROR # DEFINED BY GET SUBROUTINE
7445 030424 000137 030470      JMP      350$      ;GO TO 350$ IF ERROR WAS FOUND
7446 030430          260$:
7447
7448          ;CHECK FOR ERRORS DURING READ OPERATION
7449 030430 004737 044532      JSR      PC,PRIERR      ;GO CHECK FOR PRIMARY ERRORS
7450 030434 000405          BR      270$      ;GO TO 270$ IF NO ERROR
7451 030436 000240          NOP          ;RETURN HERE IF ERROR
    
```

```

7452 030440 104000          ERROR          ;ERROR # DEFINED BY PRIERR SUBROUTINE
7453 030442 004736          JSR          PC,2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
7454 030444 000137 030470    JMP          350$      ;GO TO 350$ IF ERROR WAS FOUND
7455 030450          270$:
7456 030450 004737 045364    JSR          PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
7457 030454 000405          BR          300$      ;GO TO 300$ IF NO ERROR
7458 030456 000240          NOP          ;RETURN HERE IF ERROR
7459 030460 104000          ERROR          ;ERROR # DEFINED BY SECERR SUBROUTINE
7460 030462 004736          JSR          PC,2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
7461 030464 000137 030470    JMP          350$      ;GO TO 350$ IF ERROR WAS FOUND
7462 030470          300$:
7463
7464 030470          350$:
7465 ;*****
7466 ;*TEST 22 WRITE, READ W/ BAI
7467 ;*****
7468 030470          †ST22:
7469 030470 000004          SCOPE          ;SCOPE CALL
7470 030472 000240          NOP          ;START OF TEST
7471 030474 012706 001100    MOV          #STACK,SP ;INITIALIZE STACK POINTER
7472 030500 013700 001276    MOV          $BASE,R0   ;R0=UNIBUS ADDRESS
7473 030504 013701 001450    MOV          TSTQUE,R1  ;(R1) = DEVICE BEING TESTED
7474 030510 012737 000022 001226 MOV          #22,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
7475
7476 ;*****
7477 ;LOOP #1 FORMAT,WRITE,READ
7478
7479 030516          10$:
7480
7481 ;PREPARE THE DEVICE FOR FORMAT OPERATION
7482 030516 004737 040020    JSR          PC,TSTPRP ;PREPARE DEVICE FOR TEST
7483 030522 054130          .WORD       054130 ;TASK DESCRIPTOR
7484 030524 000404          BR          20$      ;GO TO 20$ IF NO ERROR
7485 030526 000240          NOP          ;RETURN HERE IF ERROR
7486 030530 104000          ERROR          ;ERROR # DEFINED BY TSTPRP SUBROUTINE
7487 030532 000137 031546    JMP          350$      ;GO TO 350$ IF ERROR WAS FOUND
7488 030536          20$:
7489
7490 ;LOAD PARAMETERS AND GENERATE DATA BUFFER
7491 030536 012737 000000 001434    MOV          #0,RMDCO   ;CYLINDER = 0
7492 030544 012737 000000 001406    MOV          #0,RMDAO   ;TRACK = 0, SECTOR = 0
7493 030552 012737 107550 001404    MOV          #BUFONE,RMBAO ;BUS ADDRESS
7494 030560 012737 177376 001402    MOV          #(<C<2+256.>+1),RMWCO ;WORD COUNT
7495 030566 012737 010000 001432    MOV          #FMT16,RMFO ;16 BIT FORMAT
7496 030574 012737 000063 001400    MOV          #WH!GO,RMCSIO ;WRITE HEADER AND DATA COMMAND
7497
7498 ;VERIFY SECTOR(S) NOT IN BAD SECTOR FILE
7499
7500 030602 004737 040734          JSR          PC,BADSCT ;CALL BAD SECTOR MODULE
7501 030606 000405          BR          25$      ;GO TO 25$ IF NO ERROR
7502 030610 104401 067466          TYPE          ,SCTMSG ;TYPE BAD SECTOR MESSAGE
7503 030614 104000          ERROR          ;ERROR # DEFINED BY BADSCT SUBROUTINE
7504 030616 000137 031546          JMP          350$      ;GO TO 350$ IF ERROR WAS FOUND
7505 030622          25$:
7506 030622 012737 071046 001174    MOV          #ONES,$TMPD ;STARTING ADDRESS OF PATTERN
7507 030630 012737 000001 001176    MOV          #1,$TMP1   ;RANGE OF PATTERN

```

```

7508 030636 004737 042640 JSR PC,GENBUF ;GO GENERATE BUFFER FOR FORMAT
7509
7510 ;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATIION
7511 030642 012702 001535 MOV #PUTINX,R2 ;R2 = BYTE ENTRY POSITION
7512 030646 112722 000034 MOVB #RMDC,(R2)+
7513 030652 112722 000006 MOVB #RMDA,(R2)+
7514 030656 112722 000004 MOVB #RMBA,(R2)+
7515 030662 112722 000002 MOVB #RMWC,(R2)+
7516 030666 112722 000032 MOVB #RMOF,(R2)+
7517 030672 112722 000000 MOVB #RMCS1,(R2)+
7518 030676 112712 000200 MOVB #200,(R2) ;TERMINATE TABLE
7519 030702
7520
7521 ;FORMAT THE DRIVE
7522 030702 004737 044006 JSR PC,PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
7523 030706 000404 BR 40$ ;GO TO 40$ IF NO ERROR
7524 030710 000240 NOP ;RETURN HERE IF ERROR
7525 030712 104000 ERROR ;ERROR # DEFINED BY PUT SUBROUTINE
7526 030714 000137 031546 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
7527 030720
7528
7529 ;SETUP GET REGISTER INDEX TABLE FOR READING STATUS
7530 030720 004737 043452 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
7531
7532 ;WAIT FOR THE FORMAT TO COMPLETE
7533 030724 004737 044346 JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
7534
7535 ;GO READ STATUS FOR FORMAT OPERATION
7536 030730 004737 043536 JSR PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
7537 030734 000404 BR 50$ ;GO TO 50$ IF NO ERROR
7538 030736 000240 NOP ;RETURN HERE IF ERROR
7539 030740 104000 ERROR ;ERROR # DEFINED BY GET SUBROUTINE
7540 030742 000137 031546 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
7541 030746
7542
7543 ;VERIFY NO ERRORS DURING FORMAT
7544 030746 004737 057036 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
7545 030752 000405 BR 60$ ;GO TO 60$ IF NO ERROR
7546 030754 000240 NOP ;RETURN HERE IF ERROR
7547 030756 104000 ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
7548 030760 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
7549 030762 000137 031546 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
7550 030766
7551
7552 ;MOVE LOOP ADDRESSES TO NEXT OPERATION
7553 030766 012737 031004 001122 MOV #70$,SLPADR
7554 030774 012737 031004 001124 MOV #70$,SLPERR
7555 031002 000410 BR 80$ ;SKIP TO WRITE OPERATION
7556
7557 ;:*****
7558 ;LOOP #2 WRITE,READ
7559
7560 031004
7561
7562
7563 ;PREPARE DEVICE FOR WRITE OPERATION

```

```

7564 031004 004737 040020      JSR    PC,TSTPRP      ;PREPARE DEVICE FOR TEST
7565 031010 054130              .WORD  054130 ;TASK DESCRIPTOR
7566 031012 000404              BR     80$           ;GO TO 80$ IF NO ERROR
7567 031014 000240              NOP                    ;RETURN HERE IF ERROR
7568 031016 104000              ERROR  # DEFINED BY TSTPRP SUBROUTINE
7569 031020 000137 031546      JMP    350$         ;GO TO 350$ IF ERROR WAS FOUND
7570 031024
80$:
7571
7572
7573 031024
120$:
7574
7575 ;WRITE DATA TO THE DRIVE
7576 031024 111102      MOVB   (R1),R2      ;GENERATE CS2
7577 031026 042702 177770      BIC   #↑C<U0!U1!U2>,R2
7578 031032 042702 000010      BIC   #BAI,R2
7579 031036 010237 001410      MOV   R2,RMCS20    ;INHIBIT ADDRESS INCREMENT
7580 031042 012737 177400      MOV   #↑C256.+1>,RMWCO ;CHANGE WORD COUNT
7581 031050 012737 107554      MOV   #BUFONE+4,RMBA0 ;CHANGE ADDRESS
7582 031056 012737 000061      MOV   #WD!GO,RMCS10 ;WRITE DATA COMMAND
7583 031064 012702 001535      MOV   #PUTINX,R2   ;LOAD PUT REGISTER INDEX TABLE
7584 031070 112722 000010      MOVB  #RMCS2,(R2)+
7585 031074 112722 000006      MOVB  #RMDA,(R2)+
7586 031100 112722 000034      MOVB  #RMDC,(R2)+
7587 031104 112722 000032      MOVB  #RMOF,(R2)+
7588 031110 112722 000004      MOVB  #RMBR,(R2)+
7589 031114 112722 000002      MOVB  #RMWC,(R2)+
7590 031120 112722 000000      MOVB  #RMCS1,(R2)+
7591 031124 112722 000200      MOVB  #200,(R2)+ ;TERMINATE TABLE
7592
7593 031130 004737 044006      JSR    PC,PUT      ;GO WRITE REGISTERS WITH PUT SUBROUTINE
7594 031134 000404              BR     130$         ;GO TO 130$ IF NO ERROR
7595 031136 000240              NOP                    ;RETURN HERE IF ERROR
7596 031140 104000              ERROR  # DEFINED BY PUT SUBROUTINE
7597 031142 000137 031546      JMP    350$         ;GO TO 350$ IF ERROR WAS FOUND
7598 031146
130$:
7599
7600 ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
7601 031146 004737 043452      JSR    PC,GETSTS   ;GO TO GETSTS SUBROUTINE
7602
7603 ;WAIT FOR WRITE COMMAND TO COMPLETE
7604 031152 004737 044346      JSR    PC,TIMOUT   ;GO TO TIMOUT SUBROUTINE
7605
7606 ;GO READ STATUS FOR WRITE COMMAND
7607 031156 004737 043536      JSR    PC,GET      ;GO READ REGISTERS WITH GET SUBROUTINE
7608 031162 000404              BR     140$         ;GO TO 140$ IF NO ERROR
7609 031164 000240              NOP                    ;RETURN HERE IF ERROR
7610 031166 104000              ERROR  # DEFINED BY GET SUBROUTINE
7611 031170 000137 031546      JMP    350$         ;GO TO 350$ IF ERROR WAS FOUND
7612 031174
140$:
7613
7614 ;CHECK FOR ERRORS DURING WRITE OPERATION
7615 031174 004737 044532      JSR    PC,PRIERR   ;GO CHECK FOR PRIMARY ERRORS
7616 031200 000405              BR     150$         ;GO TO 150$ IF NO ERROR
7617 031202 000240              NOP                    ;RETURN HERE IF ERROR
7618 031204 104000              ERROR  # DEFINED BY PRIERR SUBROUTINE
7619 031206 004736      JSR    PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS

```



```

7620 031210 000137 031546          JMP      350$          ;GO TO 350$ IF ERROR WAS FOUND
7621 031214          150$:          JSR      PC,DTASTS    ;GO VERIFY RESULTS OF DATA TRANSFER
7622 031214 004737 057036          BR       160$          ;GO TO 160$ IF NO ERROR
7623 031220 000405          NOP      ;RETURN HERE IF ERROR
7624 031222 000240          ERROR   ;ERROR # DEFINED BY DTASTS SUBROUTINE
7625 031224 104000          JSR      PC,(SP)+    ;GO BACK FOR MORE ERROR CHECKS
7626 031226 004736          JMP      350$          ;GO TO 350$ IF ERROR WAS FOUND
7627 031230 000137 031546          160$:          JSR      PC,SECERR   ;GO CHECK FOR SECONDARY ERRORS
7628 031234          170$:          BR       180$          ;GO TO 180$ IF NO ERROR
7629          JSR      PC,SECERR   ;RETURN HERE IF ERROR
7630 031234 004737 045364          NOP      ;ERROR # DEFINED BY SECERR SUBROUTINE
7631 031240 000405          JSR      PC,(SP)+    ;GO BACK FOR MORE ERROR CHECKS
7632 031240 000240          JMP      350$          ;GO TO 350$ IF ERROR WAS FOUND
7633 031242 000240          180$:          JSR      PC,SECERR   ;GO CHECK FOR SECONDARY ERRORS
7634 031244 104000          BR       180$          ;GO TO 180$ IF NO ERROR
7635 031246 004736          JSR      PC,(SP)+    ;RETURN HERE IF ERROR
7636 031250 000137 031546          NOP      ;ERROR # DEFINED BY SECERR SUBROUTINE
7637 031254          JMP      350$          ;GO BACK FOR MORE ERROR CHECKS
7638          ;GO TO 350$ IF ERROR WAS FOUND
7639          ;CHANGE LOOP ADDRESSES
7640 031254 012737 031272 001122      MOV      #190$,SLPADR
7641 031262 012737 031272 001124      MOV      #190$,SLPERR
7642 031270 000410          BR       200$          ;SKIP TO NEXT OPERATION
7643          ;*****
7644          ;LOOP #3          READ
7645          190$:
7646          ;PREPARE DEVICE FOR READ OPERATION
7647 031272          JSR      PC,TSTPRP   ;PREPARE DEVICE FOR TEST
7648          .WORD 054130 ;TASK DESCRIPTOR
7649          BR       200$          ;GO TO 200$ IF NO ERROR
7650 031272 004737 040020          JSR      PC,TSTPRP   ;PREPARE DEVICE FOR TEST
7651 031276 054130          .WORD 054130 ;TASK DESCRIPTOR
7652 031300 000404          BR       200$          ;GO TO 200$ IF NO ERROR
7653 031302 000240          NOP      ;RETURN HERE IF ERROR
7654 031304 104000          ERROR   ;ERROR # DEFINED BY TSTPRP SUBROUTINE
7655 031306 000137 031546          JSR      PC,(SP)+    ;GO BACK FOR MORE ERROR CHECKS
7656 031312          JMP      350$          ;GO TO 350$ IF ERROR WAS FOUND
7657          200$:
7658 031312          240$:
7659          ;READ DATA FROM DEVICE
7660          MOV      (R1),R2          ;RESET BAI
7661 031312 111102          MOV      #1C(U0!U1!U2),R2
7662 031314 042702 177770          BIC     #1C(U0!U1!U2),R2
7663 031320 010237 001410          MOV     R2,RMCS20
7664 031324 012737 110560 001404      MOV     #BUFTWO+4,RMBA0 ;CHANGE BUFFER
7665 031332 012737 000071 001400      MOV     #RD!GO,RMCS10 ;READ DATA COMMAND
7666 031340 012702 001535          MOV     #PUTINX,R2 ;LOAD PUT REGISTER INDEX TABLE
7667 031344 112722 000006          MOV     #RMDA,(R2)+
7668 031350 112722 000032          MOV     #RMOF,(R2)+
7669 031354 112722 000034          MOV     #RMDC,(R2)+
7670 031360 112722 000004          MOV     #RMBA,(R2)+
7671 031364 112722 000002          MOV     #RMWC,(R2)+
7672 031370 112722 000000          MOV     #RMCS1,(R2)+
7673 031374 112712 000200          MOV     #200,(R2)
7674          JSR      PC,PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
7675 031 00 004737 044006

```

```

7676 031404 000404 BR 250$ ;GO TO 250$ IF NO ERROR
7677 031406 000240 NOP ;RETURN HERE IF ERROR
7678 031410 104000 ERROR ;ERROR # DEFINED BY PUT SUBROUTINE
7679 031412 000137 031546 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
7680 031416 250$:
7681 ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
7682 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
7683 031416 004737 043452
7684 ;WAIT FOR READ OPERATION TO COMPLETE
7685 JSR PC,TIMOUT ;GO TO TIMEOUT SUBROUTINE
7686 031422 004737 044346
7687 ;GO READ STATUS FOR READ OPERATION
7688 JSR PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
7689 031426 004737 043536 BR 260$ ;GO TO 260$ IF NO ERROR
7690 031432 000404 NOP ;RETURN HERE IF ERROR
7691 031434 000240 ERROR ;ERROR # DEFINED BY GET SUBROUTINE
7692 031436 104000 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
7693 031440 000137 031546 260$:
7694 031444 ;CHECK FOR ERRORS DURING READ OPERATION
7695 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
7696 031444 004737 044532 BR 270$ ;GO TO 270$ IF NO ERROR
7697 031450 000405 NOP ;RETURN HERE IF ERROR
7698 031452 000240 ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
7699 031454 104000 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
7700 031456 004736 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
7701 031460 000137 031546 270$:
7702 031464 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
7703 031464 004737 057036 BR 280$ ;GO TO 280$ IF NO ERROR
7704 031470 000405 NOP ;RETURN HERE IF ERROR
7705 031472 000240 ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
7706 031474 104000 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
7707 031476 004736 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
7708 031476 004736 280$:
7709 031500 000137 031546 290$:
7710 031504 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
7711 031504 004737 045364 BR 300$ ;GO TO 300$ IF NO ERROR
7712 031510 000405 NOP ;RETURN HERE IF ERROR
7713 031512 000240 ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
7714 031514 104000 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
7715 031516 004736 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
7716 031520 000137 031546 300$:
7717 031524 JSR PC,CMPBUF ;GO COMPARE WRITE, READ DATA BUFFERS
7718 031524 004737 043104 .WORD BUFOFF+4 ;STARTING ADDRESS OF WRITE BUFFER
7719 031530 107554 .WORD BUFTWO+4 ;STARTING ADDRESS OF READ BUFFER
7720 031532 110560 BR 310$ ;GO TO 310$ IF NO ERROR
7721 031534 000404 NOP ;RETURN HERE IF ERROR
7722 031536 000240 ERROR ;ERROR # DEFINED BY CMPBUF SUBROUTINE
7723 031540 104000 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
7724 031542 000137 031546 310$:
7725 031546 350$:
7726 ;*****
7727 ;*TEST 23 WRITE, READ EACH CURRENT LEVEL
7728
7729
7730
7731

```

```

7732
7733 031546
7734 031546 000004
7735 031550 000240
7736 031552 012706 001100
7737 031556 013700 001276
7738 031562 013701 001450
7739 031566 012737 000023 001226
7740
7741
7742
7743
7744 031574
7745
7746
7747 031574 004737 040020
7748 031600 054130
7749 031602 000404
7750 031604 000240
7751 031606 104000
7752 031610 000137 032620
7753 031614
7754
7755
7756 031614 012737 000000 001434
7757 031622 012737 000000 001406
7758 031630 012737 107550 001404
7759 031636 012737 177376 001402
7760 031644 012737 010000 001432
7761 031652 012737 000063 001400
7762
7763
7764
7765 031660 004737 040734
7766 031664 000405
7767 031666 104401 067466
7768 031672 104000
7769 031674 000137 032620
7770 031700
7771 031700 012737 071110 001174
7772 031706 012737 000001 001176
7773 031714 004737 042640
7774
7775
7776 031720 012702 001535
7777 031724 112722 000034
7778 031730 112722 000006
7779 031734 112722 000004
7780 031740 112722 000002
7781 031744 112722 000032
7782 031750 112722 000000
7783 031754 112712 000200
7784 031760
7785
7786
7787 031760 004737 044006

```

```

*****
TST23:
SCOPE ;SCOPE CALL
NOP ;START OF TEST
MOV #STACK, SP ;INITIALIZE STACK POINTER
MOV $BASE, R0 ;R0=UNIBUS ADDRESS
MOV TSTQUE, R1 ;(R1) = DEVICE BEING TESTED
MOV #23, $TESTN ;;SET TEST NUMBER IN APT MAIL BOX
*****
;LOOP #1 FORMAT, WRITE, READ
10$:
;PREPARE THE DEVICE FOR FORMAT OPERATION
JSR PC, TSTPRP ;PREPARE DEVICE FOR TEST
;WORD 054130 ;TASK DESCRIPTOR
BR 20$ ;GO TO 20$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
20$:
;LOAD PARAMETERS AND GENERATE DATA BUFFER
MOV #0, RMDCO ;CYLINDER = 0
25$: MOV #0, RMDAO ;TRACK = 0, SECTOR = 0
MOV #BUFONE, RMBAO ;BUS ADDRESS
MOV #(<C(2+256.>+1), RMWCO ;WORD COUNT
MOV #FMT16, RMOFO ;16 BIT FORMAT
MOV #WH!GO, RMCS10 ;WRITE HEADER AND DATA COMMAND
;VERIFY SECTOR(S) NOT IN BAD SECTOR FILE
JSR PC, BADSCT ;CALL BAD SECTOR MODULE
BR 26$ ;GO TO 26$ IF NO ERROR
TYPE ,SCTMSG ;TYPE BAD SECTOR MESSAGE
ERROR ;ERROR # DEFINED BY BADSCT SUBROUTINE
JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
26$: MOV #ZEROS, $TMPD ;STARTING ADDRESS OF PATTERN
MOV #1, $TMP1 ;RANGE OF PATTERN
JSR PC, GENBUF ;GO GENERATE BUFFER FOR FORMAT
;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATION
MOV #PUTINX, R2 ;R2 = BYTE ENTRY POSITION
MOVB #RMDC, (R2)+
MOVB #RMDA, (R2)+
MOVB #RMB, (R2)+
MOVB #RMWC, (R2)+
MOVB #RMOF, (R2)+
MOVB #RMCS1, (R2)+
MOVB #200, (R2) ;TERMINATE TABLE
30$:
;FORMAT THE DRIVE
JSR PC, PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE

```

```

7788 031764 000404          BR      40$          ;GO TO 40$ IF NO ERROR
7789 031766 000240          NOP                    ;RETURN HERE IF ERROR
7790 031770 104000          ERROR                   ;ERROR # DEFINED BY PUT SUBROUTINE
7791 031772 000137 032620  JMP      350$          ;GO TO 350$ IF ERROR WAS FOUND
7792 031776
7793
7794          ;SETUP GET REGISTER INDEX TABLE FOR READING STATUS
7795 031776 004737 043452  JSR      PC,GETSTS      ;GO TO GETSTS SUBROUTINE
7796
7797          ;WAIT FOR THE FORMAT TO COMPLETE
7798 032002 004737 044346  JSR      PC,TIMOUT      ;GO TO TIMOUT SUBROUTINE
7799
7800          ;GO READ STATUS FOR FORMAT OPERATION
7801 032006 004737 043536  JSR      PC,GET        ;GO READ REGISTERS WITH GET SUBROUTINE
7802 032012 000404          BR      50$          ;GO TO 50$ IF NO ERROR
7803 032014 000240          NOP                    ;RETURN HERE IF ERROR
7804 032016 104000          ERROR                   ;ERROR # DEFINED BY GET SUBROUTINE
7805 032020 000137 032620  JMP      350$          ;GO TO 350$ IF ERROR WAS FOUND
7806 032024
7807
7808          ;VERIFY NO ERRORS DURING FORMAT
7809 032024 004737 057036  JSR      PC,DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
7810 032030 000405          BR      60$          ;GO TO 60$ IF NO ERROR
7811 032032 000240          NOP                    ;RETURN HERE IF ERROR
7812 032034 104000          ERROR                   ;ERROR # DEFINED BY DTASTS SUBROUTINE
7813 032036 004736          JSR      PC,(SP)+      ;GO BACK FOR MORE ERROR CHECKS
7814 032040 000137 032620  JMP      350$          ;GO TO 350$ IF ERROR WAS FOUND
7815 032044
7816
7817          ;MOVE LOOP ADDRESSES TO NEXT OPERATION
7818 032044 012737 032062 001122  MOV      #70$,SLPADR
7819 032052 012737 032062 001124  MOV      #70$,SLPERR
7820 032060 000410          BR      80$          ;SKIP TO WRITE OPERATION
7821
7822          ;*****
7823          ;LOOP #2      WRITE,READ
7824
7825 032062
7826
7827
7828          ;PREPARE DEVICE FOR WRITE OPERATION
7829 032062 004737 040020  JSR      PC,TSTPRP      ;PREPARE DEVICE FOR TEST
7830 032066 054130          .WORD    054130 ;TASK DESCRIPTOR
7831 032070 000404          BR      80$          ;GO TO 80$ IF NO ERROR
7832 032072 000240          NOP                    ;RETURN HERE IF ERROR
7833 032074 104000          ERROR                   ;ERROR # DEFINED BY TSTPRP SUBROUTINE
7834 032076 000137 032620  JMP      350$          ;GO TO 350$ IF ERROR WAS FOUND
7835 032102
7836
7837
7838 032102
7839
7840          ;WRITE DATA TO THE DRIVE
7841 032102 012737 177400 001402  MOV      #(<C256.+1),RMWCO ;CHANGE WORD COUNT
7842 032110 012737 107554 001404  MOV      #BUFONE+4,RMBAO ;CHANGE ADDRESS
7843 032116 012737 000061 001400  MOV      #WD!GO,RMC$IO ;WRITE DATA COMMAND

```

```

7844 032124 012702 001535      MOV      #PUTINX,R2          ;LOAD PUT REGISTER INDEX TABLE
7845 032130 112722 000006      MOVB    #RMDA,(R2)+
7846 032134 112722 000034      MOVB    #RMDC,(R2)+
7847 032140 112722 000032      MOVB    #RMOF,(R2)+
7848 032144 112722 000004      MOVB    #RMBR,(R2)+
7849 032150 112722 000002      MOVB    #RMWC,(R2)+
7850 032154 112722 000000      MOVB    #RMCS1,(R2)+
7851 032160 112722 000200      MOVB    #200,(R2)+          ;TERMINATE TABLE
7852
7853 032164 004737 044006      JSR     PC,PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
7854 032170 000404          BR      130$ ;GO TO 130$ IF NO ERROR
7855 032172 000240          NOP
7856 032174 104000          ERROR  ;RETURN HERE IF ERROR
7857 032176 000137 032620      JMP     350$ ;ERROR # DEFINED BY PUT SUBROUTINE
7858 032202          ;GO TO 350$ IF ERROR WAS FOUND
7859
7860          ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
7861 032202 004737 043452      JSR     PC,GETSTS ;GO TO GETSTS SUBROUTINE
7862
7863          ;WAIT FOR WRITE COMMAND TO COMPLETE
7864 032206 004737 044346      JSR     PC,TIMOUT ;GO TO TIMEOUT SUBROUTINE
7865
7866          ;GO READ STATUS FOR WRITE COMMAND
7867 032212 004737 043536      JSR     PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
7868 032216 000404          BR      140$ ;GO TO 140$ IF NO ERROR
7869 032220 000240          NOP
7870 032222 104000          ERROR  ;RETURN HERE IF ERROR
7871 032224 000137 032620      JMP     350$ ;ERROR # DEFINED BY GET SUBROUTINE
7872 032230          ;GO TO 350$ IF ERROR WAS FOUND
7873
7874          ;CHECK FOR ERRORS DURING WRITE OPERATION
7875 032230 004737 044532      JSR     PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
7876 032234 000405          BR      150$ ;GO TO 150$ IF NO ERROR
7877 032236 000240          NOP
7878 032240 104000          ERROR  ;RETURN HERE IF ERROR
7879 032242 004736          JSR     PC,@(SP)+ ;ERROR # DEFINED BY PRIERR SUBROUTINE
7880 032244 000137 032620      JMP     350$ ;GO BACK FOR MORE ERROR CHECKS
7881 032250          ;GO TO 350$ IF ERROR WAS FOUND
7882 032250 004737 057036      JSR     PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
7883 032254 000405          BR      160$ ;GO TO 160$ IF NO ERROR
7884 032256 000240          NOP
7885 032260 104000          ERROR  ;RETURN HERE IF ERROR
7886 032262 004736          JSR     PC,@(SP)+ ;ERROR # DEFINED BY DTASTS SUBROUTINE
7887 032264 000137 032620      JMP     350$ ;GO BACK FOR MORE ERROR CHECKS
7888 032270          ;GO TO 350$ IF ERROR WAS FOUND
7889
7890          ;GO CHECK FOR SECONDARY ERRORS
7891 032270 004737 045364      JSR     PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
7892 032274 000405          BR      180$ ;GO TO 180$ IF NO ERROR
7893 032276 000240          NOP
7894 032300 104000          ERROR  ;RETURN HERE IF ERROR
7895 032302 004736          JSR     PC,@(SP)+ ;ERROR # DEFINED BY SECERR SUBROUTINE
7896 032304 000137 032620      JMP     350$ ;GO BACK FOR MORE ERROR CHECKS
7897 032310          ;GO TO 350$ IF ERROR WAS FOUND
7898
7899          ;CHANGE LOOP ADDRESSES

```

```

7900 032310 012737 032326 001122      MOV      #190$,SLPADR
7901 032316 012737 032326 001124      MOV      #190$,SLPERR
7902 032324 000410                BR        200$      ;SKIP TO NEXT OPERATION
7903
7904      ;*****
7905      ;LOOP #3      READ
7906
7907 032326      190$:
7908
7909      ;PREPARE DEVICE FOR READ OPERATION
7910 032326 004737 040020      JSR      PC,TSTPRP ;PREPARE DEVICE FOR TEST
7911 032332 054130                .WORD    054130 ;TASK DESCRIPTOR
7912 032334 000404                BR        200$      ;GO TO 200$ IF NO ERROR
7913 032336 000240                NOP
7914 032340 104000                ERROR    ;RETURN HERE IF ERROR
7915 032342 000137 032620      JMP      350$      ;ERROR # DEFINED BY TSTPRP SUBROUTINE
7916 032346                200$:      ;GO TO 350$ IF ERROR WAS FOUND
7917
7918 032346                240$:
7919
7920      ;READ DATA FROM DEVICE
7921 032346 012737 110560 001404      MOV      #BUFTWO+4,RMBAD ;CHANGE ADDRESS
7922 032354 012737 177400 001402      MOV      #(<C256.+1>),RMWCO
7923 032362 012737 000071 001400      MOV      #RD!GO,RMCS10 ;READ DATA COMMAND
7924 032370 012702 001535                MOV      #PUTINX,R2 ;LOAD PUT REGISTER INDEX TABLE
7925 032374 112722 000006      MOVVB   #RMDA,(R2)+
7926 032400 112722 000032      MOVVB   #RMOF,(R2)+
7927 032404 112722 000034      MOVVB   #RMDC,(R2)+
7928 032410 112722 000004      MOVVB   #RMDA,(R2)+
7929 032414 112722 000002      MOVVB   #RMWC,(R2)+
7930 032420 112722 000000      MOVVB   #RMCS1,(R2)+
7931 032424 112712 000200      MOVVB   #200,(R2)
7932
7933 032430 004737 044006      JSR      PC,PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
7934 032434 000404                BR        250$      ;GO TO 250$ IF NO ERROR
7935 032436 000240                NOP
7936 032440 104000                ERROR    ;RETURN HERE IF ERROR
7937 032442 000137 032620      JMP      350$      ;ERROR # DEFINED BY PUT SUBROUTINE
7938 032446                250$:      ;GO TO 350$ IF ERROR WAS FOUND
7939
7940      ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
7941 032446 004737 043452      JSR      PC,GETSTS ;GO TO GETSTS SUBROUTINE
7942
7943      ;WAIT FOR READ OPERATION TO COMPLETE
7944 032452 004737 044346      JSR      PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
7945
7946      ;GO READ STATUS FOR READ OPERATION
7947 032456 004737 043536      JSR      PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
7948 032462 000404                BR        260$      ;GO TO 260$ IF NO ERROR
7949 032464 000240                NOP
7950 032466 104000                ERROR    ;RETURN HERE IF ERROR
7951 032470 000137 032620      JMP      350$      ;ERROR # DEFINED BY GET SUBROUTINE
7952 032474                260$:      ;GO TO 350$ IF ERROR WAS FOUND
7953
7954      ;CHECK FOR ERRORS DURING READ OPERATION
7955 032474 004737 044532      JSR      PC,PIERR ;GO CHECK FOR PRIMARY ERRORS

```

M12

CZRMEBO RMO3/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 155
T23 WRITE, READ EACH CURRENT LEVEL

SEQ 0155

```

7956 032500 000405 BR 270$ ;GO TO 270$ IF NO ERROR
7957 032502 000240 NOP ;RETURN HERE IF ERROR
7958 032504 104000 ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
7959 032506 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
7960 032510 000137 032620 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
7961 032514 000240 270$:
7962 032514 004737 057036 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
7963 032520 000405 BR 280$ ;GO TO 280$ IF NO ERROR
7964 032522 000240 NOP ;RETURN HERE IF ERROR
7965 032524 104000 ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
7966 032526 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
7967 032530 000137 032620 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
7968 032534 280$:
7969 290$:
7970 032534 290$:
7971 032534 004737 045364 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
7972 032540 000405 BR 300$ ;GO TO 300$ IF NO ERROR
7973 032542 000240 NOP ;RETURN HERE IF ERROR
7974 032544 104000 ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
7975 032546 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
7976 032550 000137 032620 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
7977 032554 300$:
7978 032554 004737 043104 JSR PC,CMPBUF ;GO COMPARE WRITE, READ DATA BUFFERS
7979 032560 107554 .WORD BUFONE+4 ;STARTING ADDRESS OF WRITE BUFFER
7980 032562 110560 .WORD BUFTWO+4 ;STARTING ADDRESS OF READ BUFFER
7981 032564 000404 BR 310$ ;GO TO 310$ IF NO ERROR
7982 032566 000240 NOP ;RETURN HERE IF ERROR
7983 032570 104000 ERROR ;ERROR # DEFINED BY CMPBUF SUBROUTINE
7984 032572 000137 032620 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
7985 032576 310$:
7986 032576 062737 000200 001434 ADD #128,RMDCO ;ADVANCE TO NEXT THRESHOLD
7987 032604 023727 001434 001400 CMP RMDCO,#768. ;DONE??
7988 032612 101002 BHI 350$ ;YES!!
7989 032614 000137 031622 JMP 25$ ;DO NEXT CURRENT LEVEL
7990 350$:
7991 032620 350$:
7992 032620 012737 031574 001122 MOV #10$,SLPADR ;LOOP BACK TO START OF TEST
7993 032626 012737 031574 001124 MOV #10$,SLPERR
7994 *****
7995 ;TEST 24 WRITE, WRITE CHECK W/ CURRENT LEVEL SWITCHING
7996 *****
7997 ;TST24:
7998 032634 000004 SCOPE ;SCOPE CALL
7999 032636 000240 NOP ;START OF TEST
8000 032640 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
8001 032644 013700 001276 MOV $BASE,R0 ;R0=UNIBUS ADDRESS
8002 032650 013701 001450 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
8003 032654 012737 000024 001226 MOV #24,$TESTN ;SET TEST NUMBER IN APT MAIL BOX
8004 *****
8005 ;LOOP #1 FORMAT,WRITE,READ
8006
8007 10$:
8008 032662 10$:
8009 ;PREPARE THE DEVICE FOR FORMAT OPERATION
8010 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
8011 032662 004737 040020

```

```

8012 032666 054130          .WORD 054130 ;TASK DESCRIPTOR
8013 032670 000404          BR      20$      ;GO TO 20$ IF NO ERROR
8014 032672 000240          NOP                    ;RETURN HERE IF ERROR
8015 032674 104000          ERROR # 1 ;ERROR # DEFINED BY TSTPRP SUBROUTINE
8016 032676 000137 033626  JMP      350$     ;GO TO 350$ IF ERROR WAS FOUND
8017 032702
8018
8019
8020 032702 012737 000577 001434 ;LOAD PARAMETERS AND GENERATE DATA BUFFER
8021 032710 012737 002037 001406  MOV      #383, RMDC0 ;CYLINDER = 383.
8022 032716 012737 107550 001404  MOV      #2037, RMDA0 ;TRACK = 4, SECTOR = 31
8023 032724 012737 176774 001402  MOV      #BUFONE, RMBAO ;BUS ADDRESS
8024 032732 012737 010000 001432  MOV      #(<1C<2*{256.+2}>>+1), RMWC0 ;WORD COUNT
8025 032740 012737 000063 001400  MOV      #WH!GO, RMCS10 ;16 BIT FORMAT ;WRITE HEADER AND DATA COMMAND
8026
8027
8028
8029 032746 004737 040734          JSR      PC, BADSCT ;CALL BAD SECTOR MODULE
8030 032752 000405          BR      25$      ;GO TO 25$ IF NO ERROR
8031 032754 104401 067466          TYPE    , SCTMSG ;TYPE BAD SECTOR MESSAGE
8032 032760 104000          ERROR # 1 ;ERROR # DEFINED BY BADSCT SUBROUTINE
8033 032762 000137 033626  JMP      350$     ;GO TO 350$ IF ERROR WAS FOUND
8034 032766
8035 032766 012737 071046 001174 25$: MOV      #ONES, STMP0 ;STARTING ADDRESS OF PATTERN
8036 032774 012737 000001 001176  MOV      #1, STMP1 ;RANGE OF PATTERN
8037 033002 004737 042640          JSR      PC, GENBUF ;GO GENERATE BUFFER FOR FORMAT
8038
8039
8040 033006 012702 001535          ;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATION
8041 033012 112722 000034          MOV      #PUTINX, R2 ;R2 = BYTE ENTRY POSITION
8042 033016 112722 000006          MOVVB   #RMDC, (R2)+
8043 033022 112722 000004          MOVVB   #RMDA, (R2)+
8044 033026 112722 000002          MOVVB   #RMB, (R2)+
8045 033032 112722 000032          MOVVB   #RMWC, (R2)+
8046 033036 112722 000000          MOVVB   #RMOF, (R2)+
8047 033042 112712 000200          MOVVB   #RMCS1, (R2)+
8048 033046          MOVVB   #200, (R2) ;TERMINATE TABLE
8049
8050
8051 033046 004737 044006          30$: ;FORMAT THE DRIVE
8052 033052 000404          JSR      PC, PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
8053 033054 000240          BR      40$      ;GO TO 40$ IF NO ERROR
8054 033056 104000          NOP                    ;RETURN HERE IF ERROR
8055 033060 000137 033626  ERROR # 1 ;ERROR # DEFINED BY PUT SUBROUTINE
8056 033064  JMP      350$     ;GO TO 350$ IF ERROR WAS FOUND
8057
8058
8059 033064 004737 043452          40$: ;SETUP GET REGISTER INDEX TABLE FOR READING STATUS
8060
8061          JSR      PC, GETSTS ;GO TO GETSTS SUBROUTINE
8062 033070 004737 044346          ;WAIT FOR THE FORMAT TO COMPLETE
8063          JSR      PC, TIMEOUT ;GO TO TIMEOUT SUBROUTINE
8064
8065 033074 004737 043536          ;GO READ STATUS FOR FORMAT OPERATION
8066 033100 000404          JSR      PC, GET ;GO READ REGISTERS WITH GET SUBROUTINE
8067 033102 000240          BR      50$      ;GO TO 50$ IF NO ERROR
          NOP                    ;RETURN HERE IF ERROR

```



```

8068 033104 104000          ERROR          ;ERROR # DEFINED BY GET SUBROUTINE
8069 033106 000137 033626  JMP          350$          ;GO TO 350$ IF ERROR WAS FOUND
8070 033112          50$:
8071
8072          ;VERIFY NO ERRORS DURING FORMAT
8073 033112 004737 057036  JSR          PC,DTASTS    ;GO VERIFY RESULTS OF DATA TRANSFER
8074 033116 000405          BF          60$          ;GO TO 60$ IF NO ERROR
8075 033120 000240          NOP          ;RETURN HERE IF ERROR
8076 033122 104000          ERROR        ;ERROR # DEFINED BY DTASTS SUBROUTINE
8077 033124 004736          JSR          PC,(SP)+    ;GO BACK FOR MORE ERROR CHECKS
8078 033126 000137 033626  JMP          350$          ;GO TO 350$ IF ERROR WAS FOUND
8079 033132          60$:
8080
8081          ;MOVE LOOP ADDRESSES TO NEXT OPERATION
8082 033132 012737 033150 001122  MOV          #70$,SLPADR
8083 033140 012737 033150 001124  MOV          #70$,SLPERR
8084 033146 000410          BR          80$          ;SKIP TO WRITE OPERATION
8085
8086          ;*****
8087          ;LOOP #2          WRITE,READ
8088
8089 033150          70$:
8090
8091          ;PREPARE DEVICE FOR WRITE OPERATION
8092
8093 033150 004737 040020  JSR          PC,TSTPRP    ;PREPARE DEVICE FOR TEST
8094 033154 054130          .WORD      054130    ;TASK DESCRIPTOR
8095 033156 000404          BR          80$          ;GO TO 80$ IF NO ERROR
8096 033160 000240          NOP          ;RETURN HERE IF ERROR
8097 033162 104000          ERROR        ;ERROR # DEFINED BY TSTPRP SUBROUTINE
8098 033164 000137 033626  JMP          350$          ;GO TO 350$ IF ERROR WAS FOUND
8099 033170          80$:
8100
8101          120$:
8102 033170
8103
8104          ;WRITE DATA TO THE DRIVE
8105 033170 012737 177000 001402  MOV          #(<C(2*256.)+1),RMWCO ;CHANGE WORD COUNT
8106 033176 012737 107554 001404  MOV          #BUFONE+4,RMBAO    ;CHANGE ADDRESS
8107 033204 012737 000061 001400  MOV          #WD!GO,RMCS10     ;WRITE DATA COMMAND
8108 033212 012702 001535          MCV          #PUTINX,R2      ;LOAD PUT REGISTER INDEX TABLE
8109 033216 112722 000006          MOVB         #RMDA,(R2)+
8110 033222 112722 000034          MOVB         #RMDC,(R2)+
8111 033226 112722 000032          MOVB         #RMOF,(R2)+
8112 033232 112722 000004          MOVB         #RMBA,(R2)+
8113 033236 112722 000002          MOVB         #RMWC,(R2)+
8114 033242 112722 000000          MOVB         #RMCS1,(R2)+
8115 033246 112722 000200          MOVB         #200,(R2)+    ;TERMINATE TABLE
8116
8117 033252 004737 044006          JSR          PC,PUT      ;GO WRITE REGISTERS WITH PUT SUBROUTINE
8118 033256 000404          BR          130$        ;GO TO 130$ IF NO ERROR
8119 033260 000240          NOP          ;RETURN HERE IF ERROR
8120 033262 104000          ERROR        ;ERROR # DEFINED BY PUT SUBROUTINE
8121 033264 000137 033626  JMP          350$        ;GO TO 350$ IF ERROR WAS FOUND
8122
8123          130$:

```

```

8124 ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
8125 033270 004737 043452 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
8126 ;WAIT FOR WRITE COMMAND TO COMPLETE
8127 JSR PC,TIMOUT ;GO TO TIMEOUT SUBROUTINE
8128 033274 004737 044346
8129 ;GO READ STATUS FOR WRITE COMMAND
8130 JSR PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
8131 033300 004737 043536 BR 140$ ;GO TO 140$ IF NO ERROR
8132 033304 000404 ;RETURN HERE IF ERROR
8133 033306 000240 ;ERROR # DEFINED BY GET SUBROUTINE
8134 033310 104000 ;GO TO 350$ IF ERROR WAS FOUND
8135 033312 000137 033626 JMP 350$
8136 033316 140$:
8137 ;CHECK FOR ERRORS DURING WRITE OPERATION
8138 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
8139 033316 004737 044532 BR 150$ ;GO TO 150$ IF NO ERROR
8140 033322 000405 ;RETURN HERE IF ERROR
8141 033324 000240 ;ERROR # DEFINED BY PRIERR SUBROUTINE
8142 033326 104000 ;GO BACK FOR MORE ERROR CHECKS
8143 033330 004736 JSR PC,2(SP)+ ;GO TO 350$ IF ERROR WAS FOUND
8144 033332 000137 033626 JMP 350$
8145 033336 150$:
8146 033336 004737 057036 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
8147 033342 000405 BR 160$ ;GO TO 160$ IF NO ERROR
8148 033344 000240 ;RETURN HERE IF ERROR
8149 033346 104000 ;ERROR # DEFINED BY DTASTS SUBROUTINE
8150 033350 004736 JSR PC,2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
8151 033352 000137 033626 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
8152 033356 160$:
8153 170$:
8154 033356 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
8155 033356 004737 045364 BR 180$ ;GO TO 180$ IF NO ERROR
8156 033362 000405 ;RETURN HERE IF ERROR
8157 033364 000240 ;ERROR # DEFINED BY SECERR SUBROUTINE
8158 033366 104000 ;GO BACK FOR MORE ERROR CHECKS
8159 033370 004736 JSR PC,2(SP)+ ;GO TO 350$ IF ERROR WAS FOUND
8160 033372 000137 033626 JMP 350$
8161 033376 180$:
8162 ;CHANGE LOOP ADDRESSES
8163 MOV #190$,SLPADR
8164 033376 012737 033414 001122 MOV #190$,SLPERR
8165 033404 012737 033414 001124 BR 200$ ;SKIP TO NEXT OPERATION
8166 033412 000410
8167 ;*****
8168 ;LOOP #3 READ
8169
8170 190$:
8171 033414
8172 ;PREPARE DEVICE FOR READ OPERATION
8173 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
8174 033414 004737 040020 .WORD 054130 ;TASK DESCRIPTOR
8175 033420 054130 BR 200$ ;GO TO 200$ IF NO ERROR
8176 033422 000404 ;RETURN HERE IF ERROR
8177 033424 000240 ;ERROR # DEFINED BY TSTPRP SUBROUTINE
8178 033426 104000 ;GO TO 350$ IF ERROR WAS FOUND
8179 033430 000137 033626 JMP 350$

```

```

8180 033434      200$:
8181
8182 033434      240$:
8183
8184      ;READ DATA FROM DEVICE
8185 033434 012737 000051 001400      MOV      #WCD:GO, RMCS10      ;READ DATA COMMAND
8186 033442 012702 001535      MOV      #PUTINX, R2      ;LOAD PUT REGISTER INDEX TABLE
8187 033446 112722 000006      MOVVB   #RMDA, (R2)+
8188 033452 112722 000032      MOVVB   #RMOF, (R2)+
8189 033456 112722 000034      MOVVB   #RMDC, (R2)+
8190 033462 112722 000004      MOVVB   #RMBA, (R2)+
8191 033466 112722 000002      MOVVB   #RMWC, (R2)+
8192 033472 112722 000000      MOVVB   #RMCS1, (R2)+
8193 033476 112712 000200      MOVVB   #200, (R2)
8194
8195 033502 004737 044006      JSR     PC, PUT      ;GO WRITE REGISTERS WITH PUT SUBROUTINE
8196 033506 000404      BR      250$      ;GO TO 250$ IF NO ERROR
8197 033510 000240      NOP
8198 033512 104000      ERROR   ;RETURN HERE IF ERROR
8199 033514 000137 033626      JMP     350$      ;ERROR # DEFINED BY PUT SUBROUTINE
8200 033520      ;GO TO 350$ IF ERROR WAS FOUND
8201
8202      ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
8203 033520 004737 043452      JSR     PC, GETSTS      ;GO TO GETSTS SUBROUTINE
8204
8205      ;WAIT FOR READ OPERATION TO COMPLETE
8206 033524 004737 044346      JSR     PC, TIMEOUT      ;GO TO TIMEOUT SUBROUTINE
8207
8208      ;GO READ STATUS FOR READ OPERATION
8209 033530 004737 043536      JSR     PC, GET      ;GO READ REGISTERS WITH GET SUBROUTINE
8210 033534 000404      BR      260$      ;GO TO 260$ IF NO ERROR
8211 033536 000240      NOP
8212 033540 104000      ERROR   ;RETURN HERE IF ERROR
8213 033542 000137 033626      JMP     350$      ;ERROR # DEFINED BY GET SUBROUTINE
8214 033546      ;GO TO 350$ IF ERROR WAS FOUND
8215
8216      ;CHECK FOR ERRORS DURING READ OPERATION
8217 033546 004737 044532      JSR     PC, PRIERR      ;GO CHECK FOR PRIMARY ERRORS
8218 033552 000405      BR      270$      ;GO TO 270$ IF NO ERROR
8219 033554 000240      NOP
8220 033556 104000      ERROR   ;RETURN HERE IF ERROR
8221 033560 004736      JSR     PC, 2(SP)+      ;ERROR # DEFINED BY PRIERR SUBROUTINE
8222 033562 000137 033626      JMP     350$      ;GO BACK FOR MORE ERROR CHECKS
8223 033566      ;GO TO 350$ IF ERROR WAS FOUND
8224 033566 004737 057036      JSR     PC, DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
8225 033572 000405      BR      280$      ;GO TO 280$ IF NO ERROR
8226 033574 000240      NOP
8227 033576 104000      ERROR   ;RETURN HERE IF ERROR
8228 033600 004736      JSR     PC, 2(SP)+      ;ERROR # DEFINED BY DTASTS SUBROUTINE
8229 033602 000137 033626      JMP     350$      ;GO BACK FOR MORE ERROR CHECKS
8230 033606      ;GO TO 350$ IF ERROR WAS FOUND
8231
8232      280$:
8233 033606      290$:
8234 033612 004737 045364      JSR     PC, SECERR      ;GO CHECK FOR SECONDARY ERRORS
8235 033614 000405      BR      300$      ;GO TO 300$ IF NO ERROR
      000240      NOP      ;RETURN HERE IF ERROR

```

```

8236 033616 104000
8237 033620 004736
8238 033622 000137 033626
8239 033626
8240
8241 033626
8242
8243
8244
8245 033626
8246 033626 000004
8247 033630 000240
8248 033632 012706 001100
8249 033636 013700 001276
8250 033642 013701 001450
8251 033646 012737 000025 001226
8252
8253
8254
8255
8256 033654
8257
8258
8259 033654 004737 040020
8260 033660 054130
8261 033662 000404
8262 033664 000240
8263 033666 104000
8264 033670 000137 034650
8265 033674
8266
8267
8268 033674 012737 000000 001434
8269 033702 012737 000000 001406
8270 033710 012737 107550 001404
8271 033716 012737 177376 001402
8272 033724 012737 010000 001432
8273 033732 012737 000063 001400
8274
8275
8276
8277 033740 004737 040734
8278 033744 000405
8279 033746 104401 067466
8280 033752 104000
8281 033754 000137 034650
8282 033760
8283 033760 012737 071322 001174
8284 033766 012737 000001 001176
8285 033774 004737 042640
8286
8287
8288 034000 012702 001535
8289 034004 112722 000034
8290 034010 112722 000006
8291 034014 112722 000004

```

```

ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
JSR PC,2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND

300$:
350$:
;*****
;TEST 25 WRITE, READ EARLY PEAK SHIFT
;*****
TST25:
MOV SCOPE ;SCOPE CALL
NOP ;START OF TEST
MOV #STACK,SP ;INITIALIZE STACK POINTER
MOV $BASE,R0 ;R0=UNIBUS ADDRESS
MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
MOV #25,$TESTN ;SET TEST NUMBER IN APT MAIL BOX

;*****
;LOOP #1 FORMAT,WRITE,READ

10$:
;PREPARE THE DEVICE FOR FORMAT OPERATION
JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
WORD 054130 ;TASK DESCRIPTOR
BR 20$ ;GO TO 20$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND

20$:
;LOAD PARAMETERS AND GENERATE DATA BUFFER
MOV #0,RMDCO ;CYLINDER = 0
MOV #0,RMDAO ;TRACK = 0, SECTOR = 0
MOV #BUFONE,RMBAO ;BUS ADDRESS
MOV #(<↑C<2+256.>+1),RMWCO ;WORD COUNT
MOV #FMT16,RMFOF ;16 BIT FORMAT
MOV #WH!GO,RMCS10 ;WRITE HEADER AND DATA COMMAND

;VERIFY SECTOR(S) NOT IN BAD SECTOR FILE
JSR PC,BADSCT ;CALL BAD SECTOR MODULE
BR 25$ ;GO TO 25$ IF NO ERROR
TYPE ,SCTMSG ;TYPE BAD SECTOR MESSAGE
ERROR ;ERROR # DEFINED BY BADSCT SUBROUTINE
JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND

25$:
MOV #EARLY,$TMP0 ;STARTING ADDRESS OF PATTERN
MOV #1,$TMP1 ;RANGE OF PATTERN
JSR PC,GENBUF ;GO GENERATE BUFFER FOR FORMAT

;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATIION
MOV #PUTINX,R2 ;R2 = BYTE ENTRY POSITION
MOVB #RMDC,(R2)+
MOVB #RMDA,(R2)+
MOVB #RMBA,(R2)+

```

F13

CZRMEBO RM03/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 161
T25 WRITE, READ EARLY PEAK SHIFT

SEQ 0161

8292	034020	112722	000002		MOVW #RMWC, (R2)+
8293	034024	112722	000032		MOVW #RMOF, (R2)+
8294	034030	112722	000000		MOVW #RMCS1, (R2)+
8295	034034	112712	000200		MOVW #200, (R2) ; TERMINATE TABLE
8296	034040			30\$:	
8297					
8298					
8299	034040	004737	044006		;FORMAT THE DRIVE
8300	034044	000404			JSR PC,PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
8301	034046	000240			BR 40\$;GO TO 40\$ IF NO ERROR
8302	034050	104000			NOP ;RETURN HERE IF ERROR
8303	034052	000137	034650		ERROR ;ERROR # DEFINED BY PUT SUBROUTINE
8304	034056				JMP 350\$;GO TO 350\$ IF ERROR WAS FOUND
8305				40\$:	
8306					
8307	034056	004737	043452		;SETUP GET REGISTER INDEX TABLE FOR READING STATUS
8308					JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
8309					
8310	034062	004737	044346		;WAIT FOR THE FORMAT TO COMPLETE
8311					JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
8312					
8313	034066	004737	043536		;GO READ STATUS FOR FORMAT OPERATION
8314	034072	000404			JSR PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
8315	034074	000240			BR 50\$;GO TO 50\$ IF NO ERROR
8316	034076	104000			NOP ;RETURN HERE IF ERROR
8317	034100	000137	034650		ERROR ;ERROR # DEFINED BY GET SUBROUTINE
8318	034104				JMP 350\$;GO TO 350\$ IF ERROR WAS FOUND
8319				50\$:	
8320					
8321	034104	004737	057036		;VERIFY NO ERRORS DURING FORMAT
8322	034110	000405			JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
8323	034112	000240			BR 60\$;GO TO 60\$ IF NO ERROR
8324	034114	104000			NOP ;RETURN HERE IF ERROR
8325	034116	004736			ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
8326	034120	000137	034650		JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
8327	034124				JMP 350\$;GO TO 350\$ IF ERROR WAS FOUND
8328				60\$:	
8329					
8330	034124	012737	034142	001122	;MOVE LOOP ADDRESSES TO NEXT OPERATION
8331	034132	012737	034142	001124	MOV #70\$, \$LPADR
8332	034140	000410			MOV #70\$, \$LPERR
8333					BR 80\$;SKIP TO WRITE OPERATION
8334					
8335					;*****
8336					;LOOP #2 WRITE,READ
8337	034142				70\$:
8338					
8339					
8340					
8341	034142	004737	040020		;PREPARE DEVICE FOR WRITE OPERATION
8342	034146	054130			JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
8343	034150	000404			.WORD 054130 ;TASK DESCRIPTOR
8344	034152	000240			BR 80\$;GO TO 80\$ IF NO ERROR
8345	034154	104000			NOP ;RETURN HERE IF ERROR
8346	034156	000137	034650		ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
8347	034162				JMP 350\$;GO TO 350\$ IF ERROR WAS FOUND
				80\$:	

```

8348
8349
8350 034162          120$:
8351
8352                ;WRITE DATA TO THE DRIVE
8353 034162 012737 177400 001402      MOV      #(<C256.+1>, RMWCO      ;CHANGE WORD COUNT
8354 034170 012737 107554 001404      MOV      #BUFONE+4, RMBA0      ;CHANGE ADDRESS
8355 034176 012737 000061 001400      MOV      #WD!GO, RMCS10      ;WRITE DATA COMMAND
8356 034204 012702 001535                MOV      #PUTINX, R2          ;LOAD PUT REGISTER INDEX TABLE
8357 034210 112722 000006                MOVB     #RMDA, (R2)+
8358 034214 112722 000034                MOVB     #RMDC, (R2)+
8359 034220 112722 000032                MOVB     #RMOF, (R2)+
8360 034224 112722 000004                MOVB     #RMBA, (R2)+
8361 034230 112722 000002                MOVB     #RMWC, (R2)+
8362 034234 112722 000000                MOVB     #RMCS1, (R2)+
8363 034240 112722 000200                MOVB     #200, (R2)+          ;TERMINATE TABLE
8364
8365 034244 004737 044006                JSR      PC, PUT          ;GO WRITE REGISTERS WITH PUT SUBROUTINE
8366 034250 000404                BR      130$            ;GO TO 130$ IF NO ERROR
8367 034252 000240                NOP                                ;RETURN HERE IF ERROR
8368 034254 104000                ERROR           ;ERROR # DEFINED BY PUT SUBROUTINE
8369 034256 000137 034650                JMP      350$            ;GO TO 350$ IF ERROR WAS FOUND
8370 034262
8371          130$:
8372                ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
8373 034262 004737 043452                JSR      PC, GETSTS      ;GO TO GETSTS SUBROUTINE
8374
8375                ;WAIT FOR WRITE COMMAND TO COMPLETE
8376 034266 004737 044346                JSR      PC, TIMEOUT     ;GO TO TIMEOUT SUBROUTINE
8377
8378                ;GO READ STATUS FOR WRITE COMMAND
8379 034272 004737 043536                JSR      PC, GET        ;GO READ REGISTERS WITH GET SUBROUTINE
8380 034276 000404                BR      140$            ;GO TO 140$ IF NO ERROR
8381 034300 000240                NOP                                ;RETURN HERE IF ERROR
8382 034302 104000                ERROR           ;ERROR # DEFINED BY GET SUBROUTINE
8383 034304 000137 034650                JMP      350$            ;GO TO 350$ IF ERROR WAS FOUND
8384 034310
8385          140$:
8386                ;CHECK FOR ERRORS DURING WRITE OPERATION
8387 034310 004737 044532                JSR      PC, PRIERR      ;GO CHECK FOR PRIMARY ERRORS
8388 034314 000405                BR      150$            ;GO TO 150$ IF NO ERROR
8389 034316 000240                NOP                                ;RETURN HERE IF ERROR
8390 034320 104000                ERROR           ;ERROR # DEFINED BY PRIERR SUBROUTINE
8391 034322 004736                JSR      PC, 2(SP)+      ;GO BACK FOR MORE ERROR CHECKS
8392 034324 000137 034650                JMP      350$            ;GO TO 350$ IF ERROR WAS FOUND
8393 034330
8394          150$:
8395 034330 004737 057036                JSR      PC, DTASTS      ;GO VERIFY RESULTS OF DATA TRANSFER
8396 034334 000405                BR      160$            ;GO TO 160$ IF NO ERROR
8397 034336 000240                NOP                                ;RETURN HERE IF ERROR
8398 034340 104000                ERROR           ;ERROR # DEFINED BY DTASTS SUBROUTINE
8399 034342 004736                JSR      PC, 2(SP)+      ;GO BACK FOR MORE ERROR CHECKS
8400 034344 000137 034650                JMP      350$            ;GO TO 350$ IF ERROR WAS FOUND
8401 034350
8402          160$:
8403          170$:
8403 034350 004737 045364                JSR      PC, SECERR      ;GO CHECK FOR SECONDARY ERRORS

```

H13

CZRMEBO RM03/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 163
T25 WRITE, READ EARLY PEAK SHIFT

SEQ 0163

```

8404 034354 000405 BR 180$ ;GO TO 180$ IF NO ERROR
8405 034356 000240 NOP ;RETURN HERE IF ERROR
8406 034360 104000 ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
8407 034362 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
8408 034364 000137 034650 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
8409 034370 180$:
8410
8411 ;CHANGE LOOP ADDRESSES
8412 034370 012737 034406 001122 MOV #190$,$LPADR
8413 034376 012737 034406 001124 MOV #190$,$LPERR
8414 034404 000410 BR 200$ ;SKIP TO NEXT OPERATION
8415
8416 ;*****
8417 ;LOOP #3 READ
8418
8419 034406 190$:
8420
8421 ;PREPARE DEVICE FOR READ OPERATION
8422 034406 004737 040020 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
8423 034412 054130 WORD 054130 ;TASK DESCRIPTOR
8424 034414 000404 BR 200$ ;GO TO 200$ IF NO ERROR
8425 034416 000240 NOP ;RETURN HERE IF ERROR
8426 034420 104000 ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
8427 034422 000137 034650 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
8428 034426 200$:
8429
8430 034426 240$:
8431
8432 ;READ DATA FROM DEVICE
8433 034426 012737 000071 001400 MOV #RD!GO, RMCS10 ;READ DATA COMMAND
8434 034434 012737 110560 001404 MOV #BUF10+4, RMBA0 ;CHANGE BUFFER
8435 034442 012702 001535 MOV #PUTINX, R2 ;LOAD PUT REGISTER INDEX TABLE
8436 034446 112722 000006 MOVB #RMDA, (R2)+
8437 034452 112722 000032 MOVB #RMOF, (R2)+
8438 034456 112722 000034 MOVB #RMOC, (R2)+
8439 034462 112722 000004 MOVB #RMBA, (R2)+
8440 034466 112722 000002 MOVB #RMWC, (R2)+
8441 034472 112722 000000 MOVB #RMCS1, (R2)+
8442 034476 112712 000200 MOVB #200, (R2)
8443
8444 034502 004737 044006 JSR PC,PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
8445 034506 000404 BR 250$ ;GO TO 250$ IF NO ERROR
8446 034510 000240 NOP ;RETURN HERE IF ERROR
8447 034512 104000 ERROR ;ERROR # DEFINED BY PUT SUBROUTINE
8448 034514 000137 034650 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
8449 034520 250$:
8450
8451 ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
8452 034520 004737 043452 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
8453
8454 ;WAIT FOR READ OPERATION TO COMPLETE
8455 034524 004737 044346 JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
8456
8457 ;GO READ STATUS FOR READ OPERATION
8458 034530 004737 043536 JSR PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
8459 034534 000404 BR 260$ ;GO TO 260$ IF NO ERROR
```

```

8460 034536 000240          NOP          ;RETURN HERE IF ERROR
8461 034540 104000          ERROR        ;ERROR # DEFINED BY GET SUBROUTINE
8462 034542 000137 034650  JMP 350$    ;GO TO 350$ IF ERROR WAS FOUND
8463 034546                260$:
8464
8465                ;CHECK FOR ERRORS DURING READ OPERATION
8466 034546 004737 044532  JSR PC_PRIERR ;GO CHECK FOR PRIMARY ERRORS
8467 034552 000405          BR 270$     ;GO TO 270$ IF NO ERROR
8468 034554 000240          NOP          ;RETURN HERE IF ERROR
8469 034556 104000          ERROR        ;ERROR # DEFINED BY PRIERR SUBROUTINE
8470 034560 004736          JSR PC_2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
8471 034562 000137 034650  JMP 350$    ;GO TO 350$ IF ERROR WAS FOUND
8472 034566                270$:
8473 034566 004737 057036  JSR PC_DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
8474 034572 000405          BR 280$     ;GO TO 280$ IF NO ERROR
8475 034574 000240          NOP          ;RETURN HERE IF ERROR
8476 034576 104000          ERROR        ;ERROR # DEFINED BY DTASTS SUBROUTINE
8477 034600 004736          JSR PC_2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
8478 034602 000137 034650  JMP 350$    ;GO TO 350$ IF ERROR WAS FOUND
8479 034606                280$:
8480
8481                290$:
8482 034606 004737 045364  JSR PC_SECERR ;GO CHECK FOR SECONDARY ERRORS
8483 034612 000405          BR 300$     ;GO TO 300$ IF NO ERROR
8484 034614 000240          NOP          ;RETURN HERE IF ERROR
8485 034616 104000          ERROR        ;ERROR # DEFINED BY SECERR SUBROUTINE
8486 034620 004736          JSR PC_2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
8487 034622 000137 034650  JMP 350$    ;GO TO 350$ IF ERROR WAS FOUND
8488 034626                300$:
8489 034626 004737 043104  JSR PC_CMPBUF ;GO COMPARE WRITE, READ DATA BUFFERS
8490 034632 107554          .WORD BUFOFF+4 ;STARTING ADDRESS OF WRITE BUFFER
8491 034634 110560          .WORD BUFTWO+4 ;STARTING ADDRESS OF READ BUFFER
8492 034636 000404          BR 310$     ;GO TO 310$ IF NO ERROR
8493 034640 000240          NOP          ;RETURN HERE IF ERROR
8494 034642 104000          ERROR        ;ERROR # DEFINED BY CMPBUF SUBROUTINE
8495 034644 000137 034650  JMP 350$    ;GO TO 350$ IF ERROR WAS FOUND
8496 034650                310$:
8497
8498 034650                350$:
8499
8500                ;*****
8501                ;TEST 26 WRITE, READ MIXED PEAK SHIFT
8502                ;*****
8503                TST26:
8504                SCOPE          ;SCOPE CALL
8505                NOP          ;START OF TEST
8506                MOV #STACK, SP ;INITIALIZE STACK POINTER
8507                MOV $BASE, RO  ;RO=UNIBUS ADDRESS
8508                MOV TSTQUE, R1 ;(R1) = DEVICE BEING TESTED
8509                MOV #26, $TESTN ;SET TEST NUMBER IN APT MAIL BOX
8510                ;*****
8511                ;LOOP #1      FORMAT, WRITE, READ
8512
8513 034676                10$:
8514
8515                ;PREPARE THE DEVICE FOR FORMAT OPERATION

```


J13

CZARMEBO RM03/2 FCTNL TST 3
CZARMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 165
T26 WRITE, READ MIXED PEAK SHIFT

SEQ 0165

```

8516 034676 004737 040020      JSR    PC,TSTPRP      ;PREPARE DEVICE FOR TEST
8517 034702 054130              .WORD  054130 ;TASK DESCRIPTOR
8518 034704 000404              BR     20$           ;GO TO 20$ IF NO ERROR
8519 034706 000240              NOP                    ;RETURN HERE IF ERROR
8520 034710 104000              ERROR  # DEFINED BY TSTPRP SUBROUTINE
8521 034712 000137 035672      JMP    350$         ;GO TO 350$ IF ERROR WAS FOUND
8522 034716
8523
8524
20$:
;LOAD PARAMETERS AND GENERATE DATA BUFFER
8525 034716 012737 000000 001434      MOV    #0,RMDC0      ;CYLINDER = 0
8526 034724 012737 000000 001406      MOV    #0,RMDA0      ;TRACK = 0, SECTOR = 0
8527 034732 012737 107550 001404      MOV    #BUFONE,RMBA0 ;BUS ADDRESS
8528 034740 012737 177376 001402      MOV    #(<FC(2+256.)+1),RMWCO ;WORD COUNT
8529 034746 012737 010000 001432      MOV    #FMT16,RMFO0  ;16 BIT FORMAT
8530 034754 012737 000063 001400      MOV    #WH!GO,RMCS10 ;WRITE HEADER AND DATA COMMAND
8531
8532
8533
;VERIFY SECTOR(S) NOT IN BAD SECTOR FILE
8534 034762 004737 040734      JSR    PC,BADSCT     ;CALL BAD SECTOR MODULE
8535 034766 000405              BR     25$           ;GO TO 25$ IF NO ERROR
8536 034770 104401 067466      TYPE   ,SCTMSG      ;TYPE BAD SECTOR MESSAGE
8537 034774 104000              ERROR  # DEFINED BY BADSCT SUBROUTINE
8538 034774 000137 035672      JMP    350$         ;GO TO 350$ IF ERROR WAS FOUND
8539 035000
25$:
8540 035002 012737 071006 001174      MOV    #MIXED,$TMP0  ;STARTING ADDRESS OF PATTERN
8541 035010 012737 000200 001176      MOV    #128,$TMP1   ;RANGE OF PATTERN
8542 035016 004737 042640      JSR    PC,GENBUF    ;GO GENERATE BUFFER FOR FORMAT
8543
8544
;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATIION
8545 035022 012702 001535      MOV    #PUTINX,R2   ;R2 = BYTE ENTRY POSITION
8546 035026 112722 000034      MOVB  #RMDC,(R2)+
8547 035032 112722 000006      MOVB  #RMDA,(R2)+
8548 035036 112722 000004      MOVB  #RMBA,(R2)+
8549 035042 112722 000002      MOVB  #RMWC,(R2)+
8550 035046 112722 000032      MOVB  #RMOF,(R2)+
8551 035052 112722 000000      MOVB  #RMCS1,(R2)+
8552 035056 112712 000200      MOVB  #200,(R2)    ;TERMINATE TABLE
8553 035062
30$:
;FORMAT THE DRIVE
8554
8555
8556 035062 004737 044006      JSR    PC,PUT       ;GO WRITE REGISTERS WITH PUT SUBROUTINE
8557 035066 000404              BR     40$         ;GO TO 40$ IF NO ERROR
8558 035070 000240              NOP                    ;RETURN HERE IF ERROR
8559 035072 104000              ERROR  # DEFINED BY PUT SUBROUTINE
8560 035074 000137 035672      JMP    350$         ;GO TO 350$ IF ERROR WAS FOUND
8561 035100
40$:
;SETUP GET REGISTER INDEX TABLE FOR READING STATUS
8562
8563
8564 035100 004737 043452      JSR    PC,GETSTS    ;GO TO GETSTS SUBROUTINE
8565
8566
;WAIT FOR THE FORMAT TO COMPLETE
8567 035104 004737 044346      JSR    PC,TIMOUT    ;GO TO TIMOUT SUBROUTINE
8568
8569
;GO READ STATUS FOR FORMAT OPERATION
8570 035110 004737 043536      JSR    PC,GET       ;GO READ REGISTERS WITH GET SUBROUTINE
8571 035114 000404              BR     50$         ;GO TO 50$ IF NO ERROR

```

```

8572 035116 000240      NOP      ;RETURN HERE IF ERROR
8573 035120 104000      ERROR    ;ERROR # DEFINED BY GET SUBROUTINE
8574 035122 000137 035672  JMP      350$ ;GO TO 350$ IF ERROR WAS FOUND
8575 035126      50$:
8576
8577      ;VERIFY NO ERRORS DURING FORMAT
8578 035126 004737 057036  JSR      PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
8579 035132 000405      BR      60$      ;GO TO 60$ IF NO ERROR
8580 035134 000240      NOP
8581 035136 104000      ERROR    ;RETURN HERE IF ERROR
8582 035140 004736      JSR      PC,DTASTS ;ERROR # DEFINED BY DTASTS SUBROUTINE
8583 035142 000137 035672  JMP      350$    ;GO BACK FOR MORE ERROR CHECKS
8584 035146      60$:
8585
8586      ;MOVE LOOP ADDRESSES TO NEXT OPERATION
8587 035146 012737 035164 001122  MOV      #70$,SLPADR
8588 035154 012737 035164 001124  MOV      #70$,SLPERR
8589 035162 000410      BR      80$      ;SKIP TO WRITE OPERATION
8590
8591      ;*****
8592      ;LOOP #2      WRITE,READ
8593
8594 035164      70$:
8595
8596
8597      ;PREPARE DEVICE FOR WRITE OPERATION
8598 035164 004737 040020  JSR      PC,TSTPRP ;PREPARE DEVICE FOR TEST
8599 035170 054130      .WORD 054130 ;TASK DESCRIPTOR
8600 035172 000404      BR      80$      ;GO TO 80$ IF NO ERROR
8601 035174 000240      NOP
8602 035176 104000      ERROR    ;RETURN HERE IF ERROR
8603 035200 000137 035672  JMP      350$    ;ERROR # DEFINED BY TSTPRP SUBROUTINE
8604 035204      80$:
8605
8606
8607 035204      120$:
8608
8609      ;WRITE DATA TO THE DRIVE
8610 035204 012737 177400 001402  MOV      #(<C256.+1),RMWCO ;CHANGE WORD COUNT
8611 035212 012737 107554 001404  MOV      #BUFONE+4,RMBA0 ;CHANGE ADDRESS
8612 035220 012737 000061 001400  MOV      #WD!GO,RMCS10 ;WRITE DATA COMMAND
8613 035226 012702 001535      MOV      #PUTINX,R2 ;LOAD PUT REGISTER INDEX TABLE
8614 035232 112722 000006      MOV      #RMDA,(R2)+
8615 035236 112722 000034      MOV      #RMDC,(R2)+
8616 035242 112722 000032      MOV      #RMOF,(R2)+
8617 035246 112722 000004      MOV      #RMBA,(R2)+
8618 035252 112722 000002      MOV      #RMWC,(R2)+
8619 035256 112722 000000      MOV      #RMCS1,(R2)+
8620 035262 112722 000200      MOV      #200,(R2)+ ;TERMINATE TABLE
8621
8622 035266 004737 044006      JSR      PC,PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
8623 035272 000404      BR      130$    ;GO TO 130$ IF NO ERROR
8624 035274 000240      NOP
8625 035276 104000      ERROR    ;RETURN HERE IF ERROR
8626 035300 000137 035672  JMP      350$    ;ERROR # DEFINED BY PUT SUBROUTINE
8627 035304      130$:

```

```

8628
8629 ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
8630 035304 004737 043452 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
8631
8632 ;WAIT FOR WRITE COMMAND TO COMPLETE
8633 035310 004737 044346 JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
8634
8635 ;GO READ STATUS FOR WRITE COMMAND
8636 035314 004737 043536 JSR PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
8637 035320 000404 BR 140$ ;GO TO 140$ IF NO ERROR
8638 035322 000240 NOP ;RETURN HERE IF ERROR
8639 035324 104000 ERROR ;ERROR # DEFINED BY GET SUBROUTINE
8640 035326 000137 035672 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
8641 035332
8642 140$:
8643
8644 ;CHECK FOR ERRORS DURING WRITE OPERATION
8645 035332 004737 044532 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
8646 035336 000405 BR 150$ ;GO TO 150$ IF NO ERROR
8647 035340 000240 NOP ;RETURN HERE IF ERROR
8648 035342 104000 ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
8649 035344 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
8650 035346 000137 035672 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
8651 035352 004737 057036 150$: JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
8652 035356 000405 BR 160$ ;GO TO 160$ IF NO ERROR
8653 035360 000240 NOP ;RETURN HERE IF ERROR
8654 035362 104000 ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
8655 035364 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
8656 035366 000137 035672 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
8657 035372
8658 160$:
8659 035372
8660 035372 004737 045364 170$: JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
8661 035376 000405 BR 180$ ;GO TO 180$ IF NO ERROR
8662 035400 000240 NOP ;RETURN HERE IF ERROR
8663 035402 104000 ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
8664 035404 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
8665 035406 000137 035672 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
8666 035412
8667 180$:
8668 ;CHANGE LOOP ADDRESSES
8669 035412 012737 035430 001122 MOV #190$,SLPADR
8670 035420 012737 035430 001124 MOV #190$,SLPERR
8671 035426 000410 BR 200$ ;SKIP TO NEXT OPERATION
8672
8673 ;*****
8674 ;LOOP #3 READ
8675
8676 035430
8677 190$:
8678 ;PREPARE DEVICE FOR READ OPERATION
8679 035430 004737 040020 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
8680 035434 054130 .WORD 054130 ;TASK DESCRIPTOR
8681 035436 000404 BR 200$ ;GO TO 200$ IF NO ERROR
8682 035440 000240 NOP ;RETURN HERE IF ERROR
8683 035442 104000 ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE

```

M13

CZRMEBO RM03/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 168
T26 WRITE, READ MIXED PEAK SHIFT

SEQ 0168

```

8684 035444 000137 035672          JMP      350$          ;GO TO 350$ IF ERROR WAS FOUND
8685 035450          200$:
8686
8687 035450          240$:
8688
8689          ;READ DATA FROM DEVICE
8690 035450 012737 110560 001404  MOV     #BUFTWO+4,RMBA0      ;CHANGE BUFFER
8691 035456 012737 000071 001400  MOV     #RD!GO,RMCS10      ;READ DATA COMMAND
8692 035464 012702 001535          MOV     #PUTINX,R2        ;LOAD PUT REGISTER INDEX TABLE
8693 035470 112722 000006          MOVB   #RMDA,(R2)+
8694 035474 112722 000032          MOVB   #RMOF,(R2)+
8695 035500 112722 000034          MOVB   #RMDC,(R2)+
8696 035504 112722 000004          MOVB   #RMBB,(R2)+
8697 035510 112722 000002          MOVB   #RMWC,(R2)+
8698 035514 112722 000000          MOVB   #RMCS1,(R2)+
8699 035520 112712 000200          MOVB   #200,(R2)
8700
8701 035524 004737 044006          JSR    PC,PUT      ;GO WRITE REGISTERS WITH PUT SUBROUTINE
8702 035530 000404          BR     250$      ;GO TO 250$ IF NO ERROR
8703 035532 000240          NOP
8704 035534 104000          ERROR  ;RETURN HERE IF ERROR
8705 035536 000137 035672          JMP    350$      ;ERROR # DEFINED BY PUT SUBROUTINE
8706 035542          250$:          ;GO TO 350$ IF ERROR WAS FOUND
8707
8708          ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
8709 035542 004737 043452          JSR    PC,GETSTS   ;GO TO GETSTS SUBROUTINE
8710
8711          ;WAIT FOR READ OPERATION TO COMPLETE
8712 035546 004737 044346          JSR    PC,TIMOUT   ;GO TO TIMOUT SUBROUTINE
8713
8714          ;GO READ STATUS FOR READ OPERATION
8715 035552 004737 043536          JSR    PC,GET      ;GO READ REGISTERS WITH GET SUBROUTINE
8716 035556 000404          BR     260$      ;GO TO 260$ IF NO ERROR
8717 035560 000240          NOP
8718 035562 104000          ERROR  ;RETURN HERE IF ERROR
8719 035564 000137 035672          JMP    350$      ;ERROR # DEFINED BY GET SUBROUTINE
8720 035570          260$:          ;GO TO 350$ IF ERROR WAS FOUND
8721
8722          ;CHECK FOR ERRORS DURING READ OPERATION
8723 035570 004737 044532          JSR    PC,PRIERR   ;GO CHECK FOR PRIMARY ERRORS
8724 035574 000405          BR     270$      ;GO TO 270$ IF NO ERROR
8725 035576 000240          NOP
8726 035600 104000          ERROR  ;RETURN HERE IF ERROR
8727 035602 004736          JSR    PC,@(SP)+   ;ERROR # DEFINED BY PRIERR SUBROUTINE
8728 035604 000137 035672          JMP    350$      ;GO BACK FOR MORE ERROR CHECKS
8729 035610          270$:          ;GO TO 350$ IF ERROR WAS FOUND
8730 035610 004737 057036          JSR    PC,DTASTS   ;GO VERIFY RESULTS OF DATA TRANSFER
8731 035614 000405          BR     280$      ;GO TO 280$ IF NO ERROR
8732 035616 000240          NOP
8733 035620 104000          ERROR  ;RETURN HERE IF ERROR
8734 035622 004736          JSR    PC,@(SP)+   ;ERROR # DEFINED BY DTASTS SUBROUTINE
8735 035624 000137 035672          JMP    350$      ;GO BACK FOR MORE ERROR CHECKS
8736 035630          280$:          ;GO TO 350$ IF ERROR WAS FOUND
8737
8738          290$:
8739 035630 004737 045364          JSR    PC,SECERR   ;GO CHECK FOR SECONDARY ERRORS

```

```

8740 035634 000405 BR 300$ ;GO TO 300$ IF NO ERROR
8741 035636 000240 NOP ;RETURN HERE IF ERROR
8742 035640 104000 ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
8743 035642 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
8744 035644 000137 035672 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
8745 035650 300$:
8746 035650 004737 043104 JSR PC,CMPBUF ;GO COMPARE WRITE, READ DATA BUFFERS
8747 035654 107554 .WORD BUFOONE+4 ;STARTING ADDRESS OF WRITE BUFFER
8748 035656 110560 .WORD BUFTWO+4 ;STARTING ADDRESS OF READ BUFFER
8749 035660 000404 BR 310$ ;GO TO 310$ IF NO ERROR
8750 035662 000240 NOP ;RETURN HERE IF ERROR
8751 035664 104000 ERROR ;ERROR # DEFINED BY CMPBUF SUBROUTINE
8752 035666 000137 035672 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
8753 035672 310$:
8754 035672 350$:
8755 035672 ;*****
8756 ;*****
8757 ;*TEST 27 WRITE, READ EACH TRACK
8758 ;*****
8759 035672 †ST27:
8760 035672 000004 SCOPE ;SCOPE CALL
8761 035674 000240 NOP ;START OF TEST
8762 035676 012706 001100 MOV #STACK,SP ;INITIALIZE STACK POINTER
8763 035702 013700 001276 MOV $BASE,R0 ;R0=UNIBUS ADDRESS
8764 035706 013701 001450 MOV TSTQUE,R1 ;(R1) = DEVICE BEING TESTED
8765 035712 012737 000027 001226 MOV #27,$TESTN ;;SET TEST NUMBER IN APT MAIL BOX
8766 ;*****
8767 ;*****
8768 ;LOOP #1 FORMAT,WRITE,READ
8769 ;*****
8770 035720 10$:
8771 ;PREPARE THE DEVICE FOR FORMAT OPERATION
8772 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
8773 035720 004737 040020 .WORD 054130 ;TASK DESCRIPTOR
8774 035724 054130 BR 20$ ;GO TO 20$ IF NO ERROR
8775 035726 000404 NOP ;RETURN HERE IF ERROR
8776 035730 000240 ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
8777 035732 104000 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
8778 035734 000137 036736 20$:
8779 035740 ;LOAD PARAMETERS AND GENERATE DATA BUFFER
8780 MOV #0,RMDCO ;CYLINDER = 0
8781 MOV #0,RMDAO ;TRACK = 0, SECTOR = 0
8782 035740 012737 000000 001434 MOV #BUFOONE,RMBAO ;BUS ADDRESS
8783 035746 012737 000000 001406 25$: MOV #<↑C<2+256.>+1>,RMWCO ;WORD COUNT
8784 035754 012737 107550 001404 MOV #FMT16,RMOFO ;16 BIT FORMAT
8785 035762 012737 177376 001402 MOV #WH!GO,RMCS10 ;WRITE HEADER AND DATA COMMAND
8786 035770 012737 010000 001432
8787 035776 012737 000063 001400
8788 ;VERIFY SECTOR(S) NOT IN BAD SECTOR FILE
8789 JSR PC,BADSCT ;CALL BAD SECTOR MODULE
8790 BR 26$ ;GO TO 26$ IF NO ERROR
8791 036004 004737 040734 TYPE ,SCTMSG ;TYPE BAD SECTOR MESSAGE
8792 036010 000405 ERROR ;ERROR # DEFINED BY BADSCT SUBROUTINE
8793 036012 104401 067466 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
8794 036016 104000
8795 036020 000137 036736

```

```

8796 036024
8797 036024 012737 071006 001174
8798 036032 012737 000200 001176
8799 036040 004737 042640
8800
8801
8802 036044 012702 001535
8803 036050 112722 000034
8804 036054 112722 000006
8805 036060 112722 000004
8806 036064 112722 000002
8807 036070 112722 000032
8808 036074 112722 000000
8809 036100 112712 000200
8810 036104
8811
8812
8813 036104 004737 044006
8814 036110 000404
8815 036112 000240
8816 036114 104000
8817 036116 000137 036736
8818 036122
8819
8820
8821 036122 004737 043452
8822
8823
8824 036126 004737 044346
8825
8826
8827 036132 004737 043536
8828 036136 000404
8829 036140 000240
8830 036142 104000
8831 036144 000137 036736
8832 036150
8833
8834
8835 036150 004737 057036
8836 036154 000405
8837 036156 000240
8838 036160 104000
8839 036162 004736
8840 036164 000137 036736
8841 036170
8842
8843
8844 036170 012737 036206 001122
8845 036176 012737 036206 001124
8846 036204 000410
8847
8848
8849
8850
8851 036206

```

```

26$:
MOV #MIXED,$TMP0 ;STARTING ADDRESS OF PATTERN
JV #128,$TMP1 ;RANGE OF PATTERN
JSR PC,GENBUF ;GO GENERATE BUFFER FOR FORMAT

;LOAD PUT REGISTER INDEX TABLE FOR FORMAT OPERATION
MOV #PUTINX,R2 ;R2 = BYTE ENTRY POSITION
MOVB #RMD0,(R2)+
MOVB #RMDA,(R2)+
MOVB #RMB0,(R2)+
MOVB #RMC0,(R2)+
MOVB #RMOF,(R2)+
MOVB #RMCS1,(R2)+
MOVB #200,(R2) ;TERMINATE TABLE

30$:
;FORMAT THE DRIVE
JSR PC,PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
BR 40$ ;GO TO 40$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR # ;ERROR # DEFINED BY PUT SUBROUTINE
JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND

40$:
;SETUP GET REGISTER INDEX TABLE FOR READING STATUS
JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE

;WAIT FOR THE FORMAT TO COMPLETE
JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE

;GO READ STATUS FOR FORMAT OPERATION
JSR PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
BR 50$ ;GO TO 50$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR # ;ERROR # DEFINED BY GET SUBROUTINE
JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND

50$:
;VERIFY NO ERRORS DURING FORMAT
JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
BR 60$ ;GO TO 60$ IF NO ERROR
NOP ;RETURN HERE IF ERROR
ERROR # ;ERROR # DEFINED BY DTASTS SUBROUTINE
JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND

60$:
;MOVE LOOP ADDRESSES TO NEXT OPERATION
MOV #70$,$LPADR
MOV #70$,$LPERR
BR 80$ ;SKIP TO WRITE OPERATION

;*****
;LOOP #2 WRITE,READ

70$:

```

```

8852
8853
8854 ;PREPARE DEVICE FOR WRITE OPERATION
8855 036206 004737 040020 JSR PC,TSTPRP ;PREPARE DEVICE FOR TEST
8856 036212 054130 .WORD 054130 ;TASK DESCRIPTOR
8857 036214 000404 BR 80$ ;GO TO 80$ IF NO ERROR
8858 036216 000240 NOP ;RETURN HERE IF ERROR
8859 036220 104000 ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
8860 036222 000137 036736 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
8861 036226
8862
8863
8864 036226 120$:
8865
8866 ;WRITE DATA TO THE DRIVE
8867 036226 012737 107554 001404 MOV #BUFONE+4,RMBAO ;CHANGE BUS ADDRESS
8868 036234 012737 177400 001402 MOV #(<C256.+1> RMWCO ;CHANGE WORD COUNT
8869 036242 012737 000061 001400 MOV #WD!GO,RMCS10 ;WRITE DATA COMMAND
8870 036250 012702 001535 MOV #PUTINX,R2 ;LOAD PUT REGISTER INDEX TABLE
8871 036254 112722 000006 MOVB #RMDA,(R2)+
8872 036260 112722 000034 MOVB #RMOC,(R2)+
8873 036264 112722 000032 MOVB #RMOF,(R2)+
8874 036270 112722 000004 MOVB #RMBA,(R2)+
8875 036274 112722 000002 MOVB #RMWC,(R2)+
8876 036300 112722 000000 MOVB #RMCS1,(R2)+
8877 036304 112722 000200 MOVB #200,(R2)+ ;TERMINATE TABLE
8878
8879 036310 004737 044006 JSR PC,PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
8880 036314 000404 BR 130$ ;GO TO 130$ IF NO ERROR
8881 036316 000240 NOP ;RETURN HERE IF ERROR
8882 036320 104000 ERROR ;ERROR # DEFINED BY PUT SUBROUTINE
8883 036322 000137 036736 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
8884 036326
8885
8886
8887 036326 004737 043452 .SETUP REGISTER INPUT BUFFER FOR READING STATUS
8888 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
8889
8890 036332 004737 044346 ;WAIT FOR WRITE COMMAND TO COMPLETE
8891 JSR PC,TIMOUT ;GO TO TIMEOUT SUBROUTINE
8892
8893 036336 004737 043536 ;GO READ STATUS FOR WRITE COMMAND
8894 JSR PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
8895 036342 000404 BR 140$ ;GO TO 140$ IF NO ERROR
8896 036344 000240 NOP ;RETURN HERE IF ERROR
8897 036346 104000 ERROR ;ERROR # DEFINED BY GET SUBROUTINE
8898 036350 000137 036736 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
8899 036354
8900
8901 036354 004737 044532 ;CHECK FOR ERRORS DURING WRITE OPERATION
8902 036360 000405 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
8903 036362 000240 BR 150$ ;GO TO 150$ IF NO ERROR
8904 036364 104000 NOP ;RETURN HERE IF ERROR
8905 036366 004736 ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
8906 036370 000137 036736 JSR PC,@(SP)+ ;GO BACK FOR MORE ERROR CHECKS
8907 036374 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND

```

```

8908 036374 004737 057036 JSR PC_DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
8909 036400 000405 BR 160$ ;GO TO 160$ IF NO ERROR
8910 036402 000240 NOP ;RETURN HERE IF ERROR
8911 036404 104000 ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
8912 036406 004736 JSR PC_2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
8913 036410 000137 036736 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
8914 036414 160$:
8915
8916 036414 170$:
8917 036414 004737 045364 JSR PC_SECERR ;GO CHECK FOR SECONDARY ERRORS
8918 036420 000405 BR 180$ ;GO TO 180$ IF NO ERROR
8919 036422 000240 NOP ;RETURN HERE IF ERROR
8920 036424 104000 ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
8921 036426 004736 JSR PC_2(SP)+ ;GO BACK FOR MORE ERROR CHECKS
8922 036430 000137 036736 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
8923 036434 180$:
8924
8925 ;CHANGE LOOP ADDRESSES
8926 036434 012737 036452 001122 MOV #190$, $LPADR
8927 036442 012737 036452 001124 MOV #190$, $LPERR
8928 036450 000410 BR 200$ ;SKIP TO NEXT OPERATION
8929
8930 ;*****
8931 ;LOOP #3 READ
8932
8933 036452 190$:
8934
8935 ;PREPARE DEVICE FOR READ OPERATION
8936 036452 004737 040020 JSR PC_TSTPRP ;PREPARE DEVICE FOR TEST
8937 036456 054130 .WORD 054130 ;TASK DESCRIPTOR
8938 036460 000404 BR 200$ ;GO TO 200$ IF NO ERROR
8939 036462 000240 NOP ;RETURN HERE IF ERROR
8940 036464 104000 ERROR ;ERROR # DEFINED BY TSTPRP SUBROUTINE
8941 036466 000137 036736 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
8942 036472 200$:
8943
8944 036472 240$:
8945
8946 ;READ DATA FROM DEVICE
8947 036472 012737 110560 001404 MOV #BUFTWO+4, RMBA0 ;CHANGE BUS ADDRESS
8948 036500 012737 000071 001400 MOV #RD!GO, RMCS10 ;READ DATA COMMAND
8949 036506 012702 001535 MOV #PUTINX, R2 ;LOAD PUT REGISTER INDEX TABLE
8950 036512 112722 000006 MOVB #RMDA, (R2)+
8951 036516 112722 000032 MOVB #RMOF, (R2)+
8952 036522 112722 000034 MOVB #RMDC, (R2)+
8953 036526 112722 000004 MOVB #RMBA, (R2)+
8954 036532 112722 000002 MOVB #RMWC, (R2)+
8955 036536 112722 000000 MOVB #RMCS1, (R2)+
8956 036542 112712 000200 MOVB #200, (R2)
8957
8958 036546 004737 044006 JSR PC_PUT ;GO WRITE REGISTERS WITH PUT SUBROUTINE
8959 036552 000404 BR 250$ ;GO TO 250$ IF NO ERROR
8960 036554 000240 NOP ;RETURN HERE IF ERROR
8961 036556 104000 ERROR ;ERROR # DEFINED BY PUT SUBROUTINE
8962 036560 000137 036736 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
8963 036564 250$:

```



```

8964
8965 ;SETUP REGISTER INPUT BUFFER FOR READING STATUS
8966 036564 004737 043452 JSR PC,GETSTS ;GO TO GETSTS SUBROUTINE
8967
8968 ;WAIT FOR READ OPERATION TO COMPLETE
8969 036570 004737 044346 JSR PC,TIMOUT ;GO TO TIMOUT SUBROUTINE
8970
8971 ;GO READ STATUS FOR READ OPERATION
8972 036574 004737 043536 JSR PC,GET ;GO READ REGISTERS WITH GET SUBROUTINE
8973 036600 000404 BR 260$ ;GO TO 260$ IF NO ERROR
8974 036602 000240 NOP ;RETURN HERE IF ERROR
8975 036604 104000 ERROR ;ERROR # DEFINED BY GET S'UBROUTINE
8976 036606 000137 036736 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
8977 036612 260$:
8978
8979 ;CHECK FOR ERRORS DURING READ OPERATION
8980 036612 004737 044532 JSR PC,PRIERR ;GO CHECK FOR PRIMARY ERRORS
8981 036616 000405 BR 270$ ;GO TO 270$ IF NO ERROR
8982 036620 000240 NOP ;RETURN HERE IF ERROR
8983 036622 104000 ERROR ;ERROR # DEFINED BY PRIERR SUBROUTINE
8984 036624 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
8985 036626 000137 036736 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
8986 036632 270$:
8987 036632 004737 057036 JSR PC,DTASTS ;GO VERIFY RESULTS OF DATA TRANSFER
8988 036636 000405 BR 280$ ;GO TO 280$ IF NO ERROR
8989 036640 000240 NOP ;RETURN HERE IF ERROR
8990 036642 104000 ERROR ;ERROR # DEFINED BY DTASTS SUBROUTINE
8991 036644 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
8992 036646 000137 036736 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
8993 036652 280$:
8994
8995 290$:
8996 036652 004737 045364 JSR PC,SECERR ;GO CHECK FOR SECONDARY ERRORS
8997 036656 000405 BR 300$ ;GO TO 300$ IF NO ERROR
8998 036660 000240 NOP ;RETURN HERE IF ERROR
8999 036662 104000 ERROR ;ERROR # DEFINED BY SECERR SUBROUTINE
9000 036664 004736 JSR PC,(SP)+ ;GO BACK FOR MORE ERROR CHECKS
9001 036666 000137 036736 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
9002 036672 300$:
9003 036672 004737 043104 JSR PC,CMPBUF ;GO COMPARE WRITE, READ DATA BUFFERS
9004 036676 107554 .WORD BUFOFF+4 ;STARTING ADDRESS OF WRITE BUFFER
9005 036700 110560 .WORD BUFTWO+4 ;STARTING ADDRESS OF READ BUFFER
9006 036702 000404 BR 310$ ;GO TO 310$ IF NO ERROR
9007 036704 000240 NOP ;RETURN HERE IF ERROR
9008 036706 104000 ERROR ;ERROR # DEFINED BY CMPBUF SUBROUTINE
9009 036710 000137 036736 JMP 350$ ;GO TO 350$ IF ERROR WAS FOUND
9010 036714 310$:
9011 036714 062737 000400 001406 ADD #400,RMDAO ;ADVANCE TO NEXT TRACK
9012 036722 022737 002037 001406 CMP #2037,RMDAO ;DONE??
9013 036730 103402 BLO 350$ ;YES!!
9014 036732 000137 035754 JMP 255 ;TEST NEXT TRACK
9015
9016 036736 350$:
9017 036736 012737 035720 001122 MOV #10$,$LPADR ;LOOP BACK TO START OF TEST
9018 036744 012737 035720 001124 MOV #10$,$LPERR
9019 ;PUT NEWTEST HERE

```

9020 036752
9021 036752 000240
9022 036754 013700 001450
9023 036760 062700 000002
9024 036764 010037 001450
9025 036770 005710
9026 036772 001402
9027 036774 000137 007100
9028 037000 012737 001452 001450
9029
9030
9031
9032
9033
9034
9035
9036
9037
9038 037006
9039 037006 000240
9040 037010 005037 001116
9041 037014 005037 001206
9042 037020 005237 001230
9043 037024 042737 100000 001230
9044 037032 005327
9045 037034 000001
9046 037036 003063
9047 037040 012737
9048 037042 000001
9049 037044 037034
9050 037046 104401 037054
9051 037052 000407
9052
9053 037072
9054 037072 013746 001230
9055
9056 037076 104405
9057 037100 104401 037106
9058 037104 000421
9059
9060 037150
9061 037150 013746 001126
9062
9063 037154 104405
9064 037156 104401 001217
9065 037162 005037 001126
9066 037166 013700 000042
9067 037172 001405
9068 037174 000005
9069 037176 004710
9070 037200 000240
9071 037202 000240
9072 037204 000240
9073 037206
9074 037206 000137
9075 037210 063404

```

$EOSP:
NOP
MOV TSTQUE,RO
ADD #2,RO
MOV RO,TSTQUE
TST (RO)
BEQ 1$
JMP READY
MOV #TSTQUE+2,TSTQUE
.SBTTL END OF PASS ROUTINE

*****
; INCREMENT THE PASS NUMBER ($PASS)
; TYPE "END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYY"
; WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS
; IF THERES A MONITOR GO TO IT
; IF THERE ISN'T JUMP TO SHUT

$EOP:
NOP
CLR $STNM ;: ZERO THE TEST NUMBER
CLR $TIMES ;: ZERO THE NUMBER OF ITERATIONS
INC $PASS ;: INCREMENT THE PASS NUMBER
BIC #100000,$PASS ;: DON'T ALLOW A NEG. NUMBER
DEC (PC)+ ;: LOOP?

$EOPCT: .WORD 1
BGT $DOAGN ;: YES
MOV (PC)+,2(PC)+ ;: RESTORE COUNTER

$ENDCT: .WORD 1
$EOPCT
TYPE 65$ ;: TYPE ASCIZ STRING
BR 64$ ;: GET OVER THE ASCIZ
;: 65$: .ASCIZ <12><15>/END PASS #/
;: 64$: MOV $PASS,-(SP) ;: SAVE $PASS FOR TYPEOUT
;: ;: TYPE PASS NUMBER
;: GO TYPE--DECIMAL ASCII WITH SIGN
TYPDS ;:
TYPE 67$ ;: TYPE ASCIZ STRING
BR 66$ ;: GET OVER THE ASCIZ
;: 67$: .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
;: 66$: MOV $ERTTL,-(SP) ;: SAVE $ERTTL FOR TYPEOUT
;: ;: TOTAL NUMBER OF ERRORS
;: GO TYPE--DECIMAL ASCII WITH SIGN
TYPDS ;:
TYPE $CRLF ;: TYPE CARRIAGE RETURN, LINE FEED
CLR $ERTTL ;: CLEAR ERROR TOTAL
MOV #42,RO ;: GET MONITOR ADDRESS
BEQ $DOAGN ;: BRANCH IF NO MONITOR
RESET ;: CLEAR THE WORLD
$SENDAD: JSR PC,(RO) ;: GO TO MONITOR
NOP ;: SAVE ROOM
NOP ;: FOR
NOP ;: ACT11

$DOAGN: JMP 2(PC)+ ;: RETURN
$RTNAD: .WORD SHUT

```

G14

CZRMEBO RM03/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 175
END OF PASS ROUTINE

SEQ 0175

9076 037212 377 377 000 \$ENULL: .BYTE -1,-1.0 ;:NULL CHARACTER STRING
9077 037216 .EVEN

```

9078
9079
9080
9081
9082
9083
9084
9085
9086
9087
9088
9089
9090
9091
9092
9093 037216
9094 037216 104414
9095 037220 032777 020000 141726
9096 037226 001402
9097 037230 000137 037746
9098
9099 037234 104401 001217
9100 037240 104401 037762
9101 037244 013746 001234
9102
9103 037250 104403
9104 037252 003
9105 037253 000
9106 037254 005037 037752
9107 037260 013737 001226 037752
9108 037266 104401 037770
9109 037272 013746 037752
9110
9111 037276 104403
9112 037300 003
9113 037301 000
9114 037302 005037 037754
9115 037306 113737 001130 037754
9116 037314 001406
9117 037316 104401 040000
9118 037322 013746 037754
9119
9120 037326 104403
9121 037330 003
9122 037331 000
9123 037332 104401 040007
9124 037336 013746 001132
9125
9126 037342 104403
9127 037344 006
9128 037345 001
9129
9130 037346 005737 037754
9131 037352 001575
9132 037354 104401 001217
9133 037360 105037 037760

```

```

.SBTTL SUBROUTINES
;*****
.SBTTL ERROR TYPEOUT ROUTINE
;*THE ERROR TYPEOUT ROUTINE ASSEMBLES AND PRINTS INFORMATION
;*REGARDING THE DETECTION OF AN ERROR AS FOLLOWS:
;*
;* .UNIT NUMBER, TEST NUMBER, ERROR NUMBER AND PROGRAM COUNTER ARE
;*PRINTED ON THE FIRST LINE;
;* .ERROR MESSAGE IS ASSEMBLED, FORMATTED AND PRINTED ON
;*ONE OR MORE SUCCEEDING LINES;
;* .PAIRED LINES OF ERROR HEADERS AND ERROR DATA
;*ARE PRINTED AFTER THE ERROR MESSAGE.

ERRTYP:
    SAVREG
    BIT #SW13, @SWR ;INHIBIT TYPEOUTS??
    BEQ 1$ ;NO!!
    JMP 21$ ;YES!!
;TYPE UNIT NUMBER, TEST NUMBER, ERROR NUMBER, AND PROGRAM COUNTER
1$:
    TYPE ' $CRLF
    TYPE 'ERTY00 ;TYPE "UNT#"
    MOV $UNIT, -(SP) ;SAVE $UNIT FOR TYPEOUT
    ;TYPE UNIT NUMBER
    ;GO TYPE--OCTAL ASCII
    ;TYPE 3 DIGIT(S)
    ;SUPPRESS LEADING ZEROS
    ;LOAD TEST NUMBER FOR
    TYPOS
    .BYTE 3
    .BYTE 0
    CLR TSTNMB ;TYPE "TST#"
    MOV $TSTN, TSTNMB ;SAVE TSTNMB FOR TYPEOUT
    TYPE 'ERTY01 ;TYPE TEST NUMBER
    MOV $TSTNMB, (SP) ;GO TYPE--OCTAL ASCII
    ;TYPE 3 DIGIT(S)
    ;SUPPRESS LEADING ZEROS
    ;LOAD ERROR NUMBER FOR
    TYPOS
    .BYTE 3
    .BYTE 0
    CLR ERRNMB ;TYPE "ERR#"
    MOV $ITEMB, ERRNMB ;SAVE ERRNMB FOR TYPEOUT
    BEQ 2$ ;TYPE ERROR NUMBER
    TYPE 'ERTY02 ;GO TYPE--OCTAL ASCII
    MOV $ERRNMB, -(SP) ;TYPE 3 DIGIT(S)
    ;SUPPRESS LEADING ZEROS
    ;TYPE "PC="
    TYPOS
    .BYTE 3
    .BYTE 0
    TYPE 'ERTY03 ;SAVE $ERRPC FOR TYPEOUT
    MOV $ERRPC, -(SP) ;TYPE PROGRAM COUNTER
    ;GO TYPE--OCTAL ASCII
    ;TYPE 6 DIGIT(S)
    ;TYPE LEADING ZEROS
    ;GENERATE POINTER TO ERROR TABLE UNLESS ERROR NUMBER IS 0
3$:
    TST ERRNMB ;WAS AN ERROR CALLED?
    BEQ 21$ ;NO!!
    TYPE ' $CRLF ;YES-TYPE CRLF
    CLRB BOTFLG ;CLEAR BOT FLAG

```



```

9190 037630          13$:  MOV      2(R0),R1      ;R1 POINTS TO ERROR HEADER TABLE
9191 037630 016001 000002 BEQ      21$          ;BRANCH IF NO HEADER
9192 037634 001444          TYPE     $CRLF          ;(ASSUME NO DATA)
9193 037636 104401 001217 MOV      4(R0),R2      ;R2 POINTS TO DATA ADDRESS TABLE
9194 037642 016002 000004 MOV      6(R0),R3      ;R3 POINTS TO FORMAT TABLE
9195 037646 016003 000006 14$:  MOV      (R1)+,15$    ;PUT HEADER ADDRESS FOR TYPE
9196 037652 012137 037662 BEQ      21$          ;BRANCH IF END OF HEADERS
9197 037656 001433          ;(ASSUME END OF DATA)
9198
9199 037660 104401          TYPE     0              ;HEADER ADDRESS GOES HERE
9200 037662 000000 15$:  .WORD    0              ;
9201 037664 104401 001217 TYPE     $CRLF          ;DATA WITH HEADER??
9202 037670 005702          TST      R2              ;NO!!
9203 037672 001767          BEQ      14$              ;R4 POINTS TO DATA ADDRESS
9204 037674 012204          MOV      (R2)+,R4        ;R5 POINTS TO FORMAT
9205 037676 012305          MOV      (R3)+,R5        ;WHAT KIND OF DATA??
9206 037700 105725 16$:  TSTB     (R5)+          ;BINARY
9207 037702 100407          BMI      18$              ;OCTAL
9208 037704 001403          BEQ      17$              ;DECIMAL
9209 037706 013446          MOV      2(R4)+,-(SP)
9210 037710 104405          TYPDS   19$              ;
9211 037712 000405          BR       19$              ;
9212 037714 013446 17$:  MOV      2(R4)+,-(SP)
9213 037716 104402          TYPOC   19$              ;
9214 037720 000402          BR       19$              ;
9215 037722 013446 18$:  MOV      2(R4)+,-(SP)
9216 037724 104406          TYPBN   19$              ;
9217 037726 005714          TST      (R4)           ;MORE DATA??
9218 037730 001403          BEQ      20$              ;NO!!
9219 037732 104401 040015 TYPE     ERTY04          ;YES-TYPE 2 SPACES
9220 037736 000760          BR       16$              ;AND CONTINUE
9221 037740 104401 001217 20$:  TYPE     $CRLF          ;TYPE ONE BLANK LINE
9222 037744 000742          BR       14$              ;BEFORE NEXT HEADER
9223 037746 104415 21$:  RESREG   PC              ;
9224 037750 000207          RTS
9225
9226 037752 000000 TSTNMB: .WORD    0          ;TEST NUMBER
9227 037754 000000 ERRNMB: .WORD    0          ;ERROR NUMBER
9228 037756 000000 BOTADR: .WORD    0          ;BEGINNING OF TEXT ADDRESS
9229 037760 000          BOTFLG: .BYTE    0          ;BOT FLAG
9230 037761 000          CHRCNT: .BYTE    0          ;CHARACTER COUNT
9231
9232 037762 047125 052111 000043 ERTY00: .ASCIZ  @UNIT#@
9233 037770 020054 042524 052123 ERTY01: .ASCIZ  @, TEST#@
9234 037776 000043
9235 040000 020054 051105 021522 ERTY02: .ASCIZ  @, ERR#@
9236 040006 000
9237 040007 054 050040 036503 ERTY03: .ASCIZ  @, PC=@
9238 040014 000
9239 040015 040 000040 ERTY04: .ASCIZ  @ @
9240
9241 .EVEN

```

9242
9243
9244
9245
9246
9247
9248
9249
9250
9251
9252
9253
9254
9255
9256
9257
9258
9259
9260
9261
9262
9263
9264
9265
9266
9267
9268
9269
9270
9271
9272
9273
9274
9275
9276
9277
9278
9279
9280
9281
9282
9283
9284
9285
9286
9287
9288
9289
9290
9291
9292
9293
9294
9295
9296
9297

040020

040020 017637 000000 040732
040026 062716 000006
040032 105076 000000
040036 162716 000004

040042 032737 100000 040732
040050 001414
040052 004737 051226
040056 000411
040060 000401
040062 000000
040064 062716 000004
040070 113776 040062 000000
040076 000137 040720
040102

040102 032737 040000 040732
040110 001453

```
.SBTTL TEST PREPARATION MODULE

; THIS MODULE PREPARES THE RMO3 SUBSYSTEM FOR THE EXECUTION OF A TEST,
; REPORTING AN ERROR TO THE USER IF AN ERROR IS DETECTED. THE USER
; SPECIFIES TASKS TO BE PERFORMED, WHICH THE MODULE EXECUTES
; USING SUBROUTINES.

CALL:
      JSR      PC,TSTPRP      TASK/VERIFY DESCRIPTOR
      .WORD    ?              RETURN HERE IF NO ERROR
      BR       ?              RETURN HERE IF ERROR
      NOP
      ERROR    ERROR DEFINED BY MODULE

; TASK/VERIFY DESCRIPTOR
      BIT 15 = 1      SELECT DEVICE AND VERIFY DEVICE IS AVAILABLE
      BIT 14 = 1      CLEAR CONTROLLER AND SELECT DEVICE
      BIT 13          (RESERVED FOR DRIVE CLEAR)
      BIT 12 = 1      PACK ACKNOWLEDGE IF VOLUME NOT VALID
      BIT 11 = 1      RECALIBRATE IF POSITIONING IN PROGRESS
      BIT 10
      BIT 9
      BIT 8
      BIT 7
      BIT 6 = 1       VERIFY CONTROLLER CLEAR OPERATION
      BIT 5          (RESERVED FOR DRIVE CLEAR)
      BIT 4 = 1       VERIFY PACK ACKNOWLEDGE
      BIT 3 = 1       VERIFY RECALIBRATION
      BIT 2
      BIT 1
      BIT 0

TSTPRP:
; STORE TASK DESCRIPTOR AND CLEAR USER'S ERROR CALL
      MOV      2(SP),500$      ;STORE DESCRIPTOR
      ADD      #6,(SP)         ;MOVE SP TO USERS ERROR CALL
      CLRB    2(SP)           ;CLEAR ERROR CALL
      SUB      #4,(SP)         ;MOVE SP TO NO ERROR RETURN
; *****
; SELECT DEVICE AND VERIFY DEVICE AVAILABLE IF BIT 15 SET IN TASK
      BIT     #BIT15,500$      ;SELECT DEVICE??
      BEQ     30$              ;NO!!
      JSR     PC,DEVSEL        ;GO SELECT DEVICE
      BR      30$              ;NO ERROR - CONTINUE
      BR      20$

10$: .WORD
20$: ADD      #4,(SP)          ;ERROR NUMBER FROM DEVSEL
      MOVB   10$,2(SP)        ;TRANSFER ERROR TO USER
      JMP    400$

30$:
; *****
; CLEAR CONTROLLER IF BIT 14 IS SET IN TASK
      BIT     #BIT14,500$      ;CLEAR CONTROLLER??
      BEQ     120$             ;NO!!
```

```

9298 040112 004737 052700      JSR    PC,CNTCLR      ;GO CLEAR CONTROLLER
9299 040116 000411              BR     60$            ;CONTINUE - NO ERROR
9300 040120 000401              BR     50$
9301 040122 000000      40$: .WORD              ;ERROR NUMBER FROM CNTCLR
9302 040124 062716 000004      50$: ADD     #4,(SP)    ;TRANSFER ERROR TO USER
9303 040130 113776 040122 000000  MOVB   40$,2(SP)
9304 040136 000137 040720      JMP    400$
9305 040142
9306
9307      60$:
;*****
;VERIFY CONTROLLER CLEAR IF BIT6 SET IN TASK
9308 040142 032737 000100 040732  BIT    #BIT6,500$    ;VERIFY??
9309 040150 001433              BEQ    120$          ;NO!!
9310 040152 004737 043452      JSR    PC,GETSTS    ;SETUP INDEX TABLE
9311 040156 004737 043536      JSR    PC,GET
9312 040162 000411              BR     90$            ;GO GET STATUS
9313 040164 000401              BR     80$            ;NO ERROR GETTING STATUS
9314 040166 000000      70$: .WORD              ;ERROR FROM GETTING STATUS
9315 040170 062716 000004      80$: ADD     #4,(SP)    ;TRANSFER ERROR TO USER
9316 040174 113776 040166 000000  MOVB   70$,2(SP)
9317 040202 000137 040720      JMP    400$
9318 040206 004737 053016      90$: JSR    PC,CLRSTS ;GO VERIFY STATUS CLEAR
9319 040212 000412              BR     120$          ;NO ERROR IN CLEAR
9320 040214 000401              BR     110$
9321 040216 000000      100$: .WORD             ;ERROR IN STATUS CLEAR
9322 040220 005726              TST   (SP)+          ;STRIP RETURN ADDRESS TO
9323 040222 062716 000004      110$: ADD     #4,(SP)    ;SUBROUTINE AND TRANSFER
9324 040226 113776 040216 000000  MOVB   100$,2(SP)   ;ERROR TO USER
9325 040234 000137 040720      JMP    400$
9326 040240
9327
9328      120$:
;*****
;EXECUTE PACK ACKNOWLEDGE IF BIT12 SET IN TASK AND VOLUME IS
9329
9330 ;NOT VALID
9331 040240 032737 010000 040732  BIT    #BIT12,500$  ;PACK ACKNOWLEDGE??
9332 040246 001511              BEQ    240$          ;NO!!
9333 040250 112737 000012 001506  MOVB   #RMDS,GETINX ;GET RMDS
9334 040256 112737 000200 001507  MOVB   #200,GETINX+1
9335 040264 004737 043536      JSR    PC,GET
9336 040270 000411              BR     150$          ;NO ERROR GETTING RMDS
9337 040272 000401              BR     140$
9338 040274 000000      130$: .WORD             ;TRANSFER ERROR TO USER
9339 040276 062716 000004      140$: ADD     #4,(SP)
9340 040302 113776 040274 000000  MOVB   130$,2(SP)
9341 040310 000137 040720      JMP    400$
9342 040314 032737 000100 001342  150$: BIT    #VV,RMDSI ;IS VOLUME VALID??
9343 040322 001063              BNE   240$          ;YES!!
9344 040324 005037 001474              CLR   MEDENB        ;CLEAR MEDIA ENABLE
9345 040330 012737 000023 001400  MOV    #PAKACK!GO,RMCS10 ;LOAD PACK ACK COMMAND
9346 040336 112737 000000 001535  MOVB   #RMCS1,PUTINX ;SETUP REGISTER INDEX TABLE
9347 040344 112737 000200 001536  MOVB   #200,PUTINX+1
9348 040352 004737 044006      JSR    PC,PUT
9349 040356 000410              BR     180$          ;GO WRITE COMMAND
9350 040360 000401              BR     170$          ;NO ERROR LOADING REGISTER
9351 040362 000000      160$: .WORD             ;ERROR FROM PUT SUB
9352 040364 062716 000004      170$: ADD     #4,(SP)    ;TRANSFER ERROR TO USER
9353 040370 113776 040362 000000  MOVB   160$,2(SP)

```



```

9354 040376 000550          BR      400$
9355 040400          180$:
9356
9357
9358
9359 040400 032737 000020 040732 ;*****
;VERIFY PACK ACKNOWLEDGE IF #BIT4 SET IN TASK
9360 040406 001431          BIT      #BIT4,500$ ;VERIFY PACK ACKNOWLEDGE??
9361 040410 004737 043452          BEQ      240$ ;NO!!
9362 040414 004737 043536          JSR     PC,GETSTS ;SETUP FOR STATUS
9363 040420 000410          JSR     PC,GET ;GO GET STATUS
9364 040422 000401          BR      210$ ;NO ERROR GETTING STATUS
9365 040424 000000          190$: .WORD ;ERROR FROM GET SUB
9366 040426 062716 000004 200$: ADD     #4,(SP) ;TRANSFER ERROR TO USER
9367 040428 113776 040424 000000          MOVB   190$,2(SP)
9368 040440 000527          BR      400$
9369 040442 004737 053676          210$: JSR     PC,ACKSTS ;GO CHECK ACKNOWLEDGE
9370 040446 000411          BR      240$ ;NO ERROR
9371 040450 000401          BR      230$
9372 040452 000000          220$: .WORD ;PACK ACKNOWLEDGE ERROR
9373 040454 005726          230$: TST     (SP)+ ;STRIP RETURN TO SUB AND
9374 040456 062716 000004          ADD     #4,(SP) ;TRANSFER ERROR TO USER
9375 040462 113776 040452 000000          MOVB   220$,2(SP)
9376 040470 000513          BR      400$
9377 040472          240$:
9378
9379
9380 ;*****
9381 040472 032737 004000 040732 ;RECALIBRATE DRIVE IF BIT 11 SET IN TASK AND PIP IS ACTIVE
9382 040500 001513          BIT      #BIT11,500$ ;RECALIBRATE??
9383 040502 112737 000012 001506          BEQ      410$ ;NO!!
9384 040510 112737 000200 001507          MOVB   #RMS,GETINX ;LOAD REGISTER INDEX TABLE
9385 040516 004737 043536          MOVB   #200,GETINX+1
9386 040522 000410          JSR     PC,GET ;GO GET RMS
9387 040524 000401          BR      270$ ;NO ERROR GETTING RMS
9388 040526 000000          BR      260$
9389 040530 062716 000004 250$: .WORD ;ERROR FROM GET SUB
9390 040534 113776 040526 000000 260$: ADD     #4,(SP) ;TRANSFER ERROR TO USER
9391 040542 000466          MOVB   250$,2(SP)
9392 040544 032737 020000 001342 270$: BIT      #PIP,RMSI ;IS PIP ACTIVE??
9393 040552 001466          BEQ      410$ ;NO!!
9394 040554 012737 000007 001400          MOV     #RECAL!GO,RMCS10 ;LOAD RECALIBRATE COMMAND
9395 040562 112737 000000 001535          MOVB   #RMCSI,PUTINX ;AND REGISTER INDEX
9396 040570 112737 000000 001536          MOVB   #200,PUTINX+1
9397 040576 004737 043536          JSR     PC,PUT ;GO ISSUE RECALIBRATE
9398 040602 000410          BR      300$ ;NO ERROR
9399 040604 000401          BR      290$
9400 040606 000000          280$: .WORD ;ERROR IN REGISTER TRANSFER
9401 040610 062716 000004 290$: ADD     #4,(SP) ;TRANSFER ERROR TO USER
9402 040614 113776 040606 000000          MOVB   280$,2(SP)
9403 040622 000436          BR      400$
9404 040624 004737 043452          300$: JSR     PC,GETSTS ;SETUP FOR STATUS
9405 040630 004737 044346          JSR     PC,TIMOUT ;WAIT FOR COMPLETION
9406
9407
9408 ;*****
9409 040634 032737 000010 040732 ;VERIFY RECALIBRATE IF BIT 3 SET IN TASK
          BIT      #BIT3,500$ ;VERIFY RECALIBRATE??

```

9410	040642	001432			BEG	410\$;NO!!
9411	040644	004737	043536		JSR	PC,GET		;GO GET STATUS
9412	040650	000410			BR	330\$;NO ERROR GETTING STATUS
9413	040652	000401			BR	320\$		
9414	040654	000000						;ERROR FROM GET
9415	040656	062716	000004		310\$:	.WORD		;TRANSFER ERROR TO USER
9416	040662	113776	040654	000000	320\$:	ADD #4,(SP)		
9417	040670	000413			MOVB	310\$,2(SP)		
9418	040672	004737	054472		BR	400\$		
9419	040676	000414			330\$:	JSR PC,RCLSTS		;GO CHECK RECALIBRATE
9420	040700	000401			BR	410\$;NO ERROR DURING RECALIBRATE
9421	040702	000000			BR	350\$		
9422	040704	005726			340\$:	.WORD		;ERROR DURING RECALIBRATE
9423	040706	062716	000004		350\$:	TST (SP)+		;STRIP RETURN TO SUB AND
9424	040712	113776	040702	000000	ADD	#4,(SP)		;TRANSFER ERROR TO USER
9425	040720	162716	000002		MOVB	340\$,2(SP)		
9426	040724	000240			400\$:	SUB #2,(SP)		;MOVE SP BACK BEFORE ERROR
9427	040726	000240			NOP			
9428	040730	000207			NOP			
9429					410\$:	RTS PC		;RETURN TO USER
9430	040732	000000			500\$:	.WORD		;TASK/VERIFY DESCRIPTOR

91731
91732
91733
91734
91735
91736
91737
91738
91739
91740
91741
91742
91743
91744
91745
91746
91747
91748
91749
91750
91751
91752
91753
91754
91755
91756
91757
91758
91759
91760
91761
91762
91763
91764
91765
91766
91767
91768
91769
91770
91771
91772
91773
91774
91775
91776
91777
91778
91779
91780
91781
91782
91783
91784
91785
91786

.SBTTL BAD SECTOR MODULE

THE BAD SECTOR MODULE PERFORMS TWO MAJOR FUNCTIONS, I.E.
RECOVER THE BAD SECTOR FILE FROM THE LAST TRACK, AND,
APPROVE THE USAGE OF A SECTOR BASED ON INFORMATION IN
THE BAD SECTOR FILE OR ASSIGN A NEW SECTOR IF THE ONE ELECTED IS
NOT AVAILABLE FOR USE.

INFORMATION REQUIRED BY THE MODULE INCLUDES
.RMDCO, THE DESIRED CYLINDER,
.RMDAO, THE TRACK AND SECTOR ADDRESS,
.RMWCO, THE WORD COUNT,
.RMCS10, THE COMMAND.

MODULE CALL IS AS FOLLOWS,

JSR PC,BADSCT ;RETURN HERE IF NO ERROR
BR ??? ;RETURN HERE IF THE BAD SECTOR FILE
TYPE ,MESSAGE ;CANNOT BE RECOVERED
ERROR ;THE MODULE DEFINES THE ERROR NUMBER

THE MODULE IS INTENDED TO BE CALLED PRIOR TO CALLING THE BUFFER
GENERATOR SUBROUTINE, AND PRESERVES THE "PUT BUFFER" SO THAT THE
BUFFER NEED ONLY BE FILLED ONCE FOR THE EXECUTION OF A FORMAT
OPERATION.

THE MODULE RETURNS TO THE CALLING TEST WITH THE APPROVED OR ASSIGNED
SECTOR IN THE PUT BUFFER AND ALSO IN LOCATIONS "ASNDA" AND "ASNDC"
SO THAT A REFERENCE IS AVAILABLE TO THE TEST OUTSIDE OF THE PUT BUFFER.

040734

BADSCT:

;TEST "MEDIA ENABLE" TO DETERMINE WHETHER OR NOT THE BAD SECTOR FILES
;HAVE BEEN RECOVERED.
TS MEDENB
BEQ 10\$;BAD SECTOR FILE IS AVAILABLE
JMP 300\$

040734 005737 001474
040740 001402
040742 000137 042160

;RECOVER THE BAD SECTOR FILE FROM THE LAST TRACK
10\$:

040746

;SAVE THE USER'S PUT BUFFER
MOV RO,-(SP) ;;PUSH RO ON STACK
CLR RO
20\$: MOV PUTBUF(RO),BUFFER(RO)
ADD #2,RO ;ADVANCE TO NEXT BUFFER POSITION
CMP #46,RO ;END OF BUFFER
BHS 20\$;NO !!

040746 010046
040750 005000
040752 016060 001400 107550
040760 062700 000002
040764 022700 000046
040770 103370

;SET RETRY COUNT AND LOAD PUT BUFFER AND REGISTER INDEX TABLE
MOV #3,500\$;RETRY COUNT
MOV #002000,RMDAO ;STARTING SECTOR ADDRESS
MOV #822.,RMDCO ;DESIRED CYLINDER

040772 012737 000003 042626
041000 012737 002000 001406
041006 012737 001466 001434

```

9487 041014 012737 177376 001402      MOV      #(<C258,+1>,RMWCO      ;WORD COUNT
9488 041022 012737 010000 001432      MOV      #FMT16,RMOF0        ;16 BIT FORMAT
9489 041030 012737 105530 001404      MOV      #MFGFIL,RMBA0       ;BUFFER ADDRESS
9490
9491 041036 012700 001535      MOV      #PUTINX,RO          ;RO POINTS TO REGISTER INDEX TABLE
9492 041042 112720 000006      MOV      #RMDA,(RO)+
9493 041046 112720 000034      MOV      #RMDC,(RO)+
9494 041052 112720 000002      MOV      #RMWC,(RO)+
9495 041056 112720 000032      MOV      #RMOF,(RO)+
9496 041062 112720 000004      MOV      #RMBA,(RO)+
9497 041066 112720 000000      MOV      #RMCS1,(RO)+
9498 041072 112720 000200      MOV      #200,(RO)+
9499 041076 012600      MOV      (SP)+,RO            ;;POP STACK INTO RO
9500
9501      ;SET GET INDEX TABLE FOR READING STATUS
9502 041100 004737 043452      JSR      PC,GETSTS           ;SETUP GET INDEX REGISTER FOR STATUS
9503 041104      30$:
9504
9505      ;CLEAR THE DEVICE USING DRIVE CLEAR COMMAND
9506 041104 012737 000011 001400      MOV      #DRVCLR!GO,RMCS10   ;LOAD COMMAND IN PUT BUFFER
9507 041112 004737 044006      JSR      PC,PUT              ;OUTPUT COMMAND
9508 041116 000411      BR      45$                 ;RETURN HERE IF NO ERROR
9509 041120 000401      BR      40$                 ;GET AROUND ERROR #
9510 041122 000000      35$:      .WORD              ;ERROR # GOES HERE
9511
9512 041124 062716 000006      40$:      ADD      #6,(SP)            ;MOVE SP TO USERS ERROR CALL
9513 041130 113776 041122 000000      MOV      35$,2(SP)          ;MOVE ERROR NUMBER TO USER
9514 041136 000137 041636      JMP      215$
9515 041142 004737 043536      45$:      JSR      PC,GET              ;GO GET STATUS
9516 041146 000411      BR      60$                 ;RETURN HERE IF NO ERROR
9517 041150 000401      BR      55$                 ;GET AROUND ERROR #
9518 041152 000000      50$:      .WORD              ;ERROR # GOES HERE
9519
9520 041154 062716 000006      55$:      ADD      #6,(SP)            ;MOVE SP TO USERS ERROR CALL
9521 041160 113776 041152 000000      MOV      50$,2(SP)          ;MOVE ERROR # TO USERS ERROR CALL
9522 041166 000137 041636      JMP      215$
9523
9524 041172 004737 056234      60$:      JSR      PC,DRVSTS           ;GO VERIFY DRIVE CLEAR COMMAND
9525 041176 000412      BR      75$                 ;RETURN HERE IF NO ERROR
9526 041200 000401      BR      70$                 ;GET AROUND ERROR
9527 041202 000000      65$:      .WORD              ;ERROR # GOES HERE
9528
9529 041204 005726      70$:      TST      (SP)+              ;STRIP RETURN TO SUBROUTINE
9530 041206 062716 000006      ADD      #6,(SP)            ;MOVE SP TO USERS ERROR CALL
9531 041212 113776 041202 000000      MOV      65$,2(SP)          ;MOVE ERROR # TO USER CALL
9532 041220 000137 041636      JMP      215$
9533
9534 041224      75$:
9535
9536      ;ISSUE A PACK ACKNOWLEDGE IF VOLUME VALID IS RESET
9537 041224 032737 000100 001342      BIT      #VV,RMDSI           ;IS VV RESET ??
9538 041232 001050      BNE     120$                 ;NO !!
9539 041234 012737 000023 001400      MOV      #PACACK!GO,RMCS10   ;LOAD COMMAND
9540 041242 004737 044006      JSR      PC,PUT              ;GO PUT COMMAND TO DRIVE
9541 041246 000411      BR      90$                 ;RETURN HERE IF NO OUTPUT ERROR
9542 041250 000401      BR      85$                 ;GET AROUND ERROR #

```

```

9543 041252 000000      80$:      .WORD      ;ERROR # GOES HERE
9544
9545 041254 062716 000006      85$:      ADD      #6,(SP)      ;MOVE SP TO USERS ERROR CALL
9546 041260 113776 041252 000000      MOVB     80$,2(SP)      ;MOVE ERROR # TO ERROR CALL
9547 041266 000137 041636      JMP      215$
9548 041272 004737 043536      90$:      JSR      PC,GET      ;GO GET STATUS FOR PACK ACK
9549 041276 000411      BR      105$      ;RETURN HERE IF NO ERROR
9550 041300 000401      BR      100$      ;GET AROUND ERROR #
9551 041302 000000      95$:      .WORD      ;ERROR # GOES HERE
9552
9553 041304 062716 000006      100$:     ADD      #6,(SP)      ;MOVE SP TO USERS ERROR CALL
9554 041310 113776 041302 000000      MOVB     95$,2(SP)      ;MOVE ERROR # TO CALL
9555 041316 000137 041636      JMP      215$
9556
9557 041322 004737 053676      105$:     JSR      PC,ACKSTS     ;GO VERIFY ACKNOWLEDGE STATUS
9558 041326 000412      BR      120$      ;RETURN HERE IF NO ERROR
9559 041330 000401      BR      115$      ;GET AROUND ERROR #
9560 041332 000000      110$:     .WORD      ;ERROR # GOES HERE
9561
9562 041334 005726      115$:     TST      (SP)+      ;STRIP RETURN TO SUBROUTINE
9563 041336 062716 000006      ADD      #6,(SP)      ;MOVE SP TO USERS ERROR CALL
9564 041342 113776 041332 000000      MOVB     110$,2(SP)     ;MOVE ERROR # TO USERS ERROR CALL
9565 041350 000137 041636      JMP      215$
9566
9567 041354      120$:
9568
9569      ;RECALIBRATE THE DRIVE IF PIP IS SET
9570 041354 032737 020000 001342      BIT      #PIP,RMSI      ;IS PIP SET ??
9571 041362 001452      BEQ      165$      ;NO !!
9572
9573 041364 012737 000007 001400      MOV      #RECAL!GO,RMCS10 ;LOAD RECALIBRATE COMMAND
9574 041372 004737 044006      JSR      PC,PUT      ;PUT THE RECAL COMMAND
9575 041376 000411      BR      135$      ;RETURN HERE IF NO ERROR
9576 041400 000401      BR      130$      ;GET AROUND ERROR #
9577 041402 000000      125$:     .WORD      ;ERROR # GOES HERE
9578
9579 041404 062716 000006      130$:     ADD      #6,(SP)      ;MOVE SP TO USERS ERROR CALL
9580 041410 113776 041402 000000      MOVB     125$,2(SP)     ;MOVE ERROR # TO USERS CALL
9581 041416 000137 041636      JMP      215$
9582 041422
9583 041422 004737 044346      135$:     JSR      PC,TIMOUT     ;WAIT FOR RECALIBRATE TO COMPLETE
9584 041426 004737 043536      JSR      PC,GET      ;GO GET RECAL STATUS
9585 041432 000411      BR      150$      ;RETURN HERE IF NO ERROR
9586 041434 000401      BR      145$      ;GET AROUND ERROR #
9587 041436 000000      140$:     .WORD      ;ERROR # GOES HERE
9588
9589 041440 062716 000006      145$:     ADD      #6,(SP)      ;MOVE SP TO USERS ERROR CALL
9590 041444 113776 041436 000000      MOVB     140$,2(SP)     ;MOVE ERROR TO USERS CALL
9591 041452 000137 041636      JMP      215$
9592
9593 041456 004737 054472      150$:     JSR      PC,RCLSTS     ;GO VERIFY RECALIBRATE STATUS
9594 041462 000412      BR      165$      ;RETURN HERE IF NO ERROR
9595 041464 000401      BR      160$      ;GET AROUND ERROR #
9596 041466 000000      155$:     .WORD      ;ERROR # GOES HERE
9597
9598 041470 005726      160$:     TST      (SP)+      ;STRIP RETURN TO SUBROUTINE

```

```

9599 041472 062716 000006          ADD    #6,(SP)          ;MOVE SP TO USERS ERROR CALL
9600 041476 113776 041466 000000    MOVB   155$,2(SP)      ;MOVE ERROR # TO USERS CALL
9601 041504 000137 041636          JMP    215$
9602
9603 041510          165$:
9604
9605          ;READ THE SECTOR IDENTIFIED BY RMDAO, INCLUDING HEADER AND DATA
9606
9607 041510 012737 000073 001400    MOV    #RH!GO,RMCS10  ;LOAD READ HEADER AND DATA COMMAND
9608 041516 004737 044006          JSR    PC,PUT          ;PUT COMMAND
9609 041522 000411          BR    180$            ;RETURN HERE IF NO ERROR
9610 041524 000401          BR    175$            ;GET AROUND ERROR #
9611 041526 000000          170$: .WORD          ;ERROR # GOES HERE
9612
9613 041530 062716 000006          175$: ADD    #6,(SP)          ;MOVE SP TO USERS ERROR CALL
9614 041534 113776 041526 000000    MOVB   170$,2(SP)      ;MOVE ERROR # TO USERS ERROR CALL
9615 041542 000137 041636          JMP    215$
9616
9617 041546 004737 044346          180$: JSR    PC,TIMOUT    ;WAIT FOR READ OPERATION TO COMPLETE
9618 041552 004737 043536          JSR    PC,GET          ;GO GET STATUS FOR READ OPERATION
9619 041556 000411          BR    195$            ;RETURN HERE IF NO ERROR
9620 041560 000401          BR    190$            ;GET AROUND ERROR #
9621 041562 000000          185$: .WORD          ;ERROR # GOES HERE
9622
9623 041564 062716 000006          190$: ADD    #6,(SP)          ;MOVE SP TO USERS ERROR CALL
9624 041570 113776 041562 000000    MOVB   185$,2(SP)      ;MOVE ERROR # TO CALL
9625 041576 000137 041636          JMP    215$
9626
9627 041602 004737 057036          195$: JSR    PC,DASTS    ;GO VERIFY RESULTS OF READ OPERATION
9628 041606 000412          BR    210$            ;RETURN HERE IF NO ERROR
9629 041610 000401          BR    205$            ;GET AROUND ERROR #
9630 041612 000000          200$: .WORD          ;ERROR # GOES HERE
9631
9632 041614 005726          205$: TST    (SP)+        ;STRIP RETURN ADDRESS TO SUBROUTINE
9633 041616 062716 000006          ADD    #6,(SP)          ;MOVE SP TO USERS ERROR CALL
9634 041622 113776 041612 000000    MOVB   200$,2(SP)      ;MOVE ERROR # TO USERS CALL
9635 041630 000137 041636          JMP    215$
9636
9637 041634 000456          210$: BR    240$
9638
9639 041636          215$:
9640
9641          ;AN ERROR HAS BEEN DETECTED IN TRYING TO READ THE BAD SECTOR FILE.
9642          ;THE SECTOR WILL BE RETRIED IF POSSIBLE.
9643
9644 041636 032737 010000 001342    BIT    #MOL,RMSDI      ;IS MEDIUM ON LINE ??
9645 041644 001446          BEQ    230$            ;YES !!
9646
9647 041646 005337 042626          DEC    500$            ;DECREMENT RETRY COUNT
9648 041652 100037          BPL    225$            ;RETRY IF COUNT NOT NEG
9649
9650          ;THE RETRY COUNT HAS EXPIRED - SEE IF THE ERROR IS MEDIA RELATED
9651 041654 013746 001344          MOV    #RMR1I,-(SP)    ;GET EPI
9652 041657 042716 100720          BIC    #DCK!HCRC!HCE!FER!ECH,(SP)
9653 041658 032737 000010 001372    BIT    #DPE,RMR2I      ;WAS THERE A DATA PARITY ERROR ??
9654 041659 001402          BEG    220$            ;NO !!

```

```

9655 041674 042716 070010
9656 041700 005726
9657 041702 001027
9658 041704 013746 001372
9659 041710 042726 100010
9660 041714 001022
9661
9662
9663
9664
9665
9666 041716 062737 000002 001406
9667 041724 122737 000012 001406
9668 041732 001413
9669 041734 122737 000040 001406
9670 041742 001407
9671 041744 012737 000003 042626
9672 041752 162716 000006
9673 041756 000137 041104
9674
9675 041762
9676
9677
9678 041762 162716 000004
9679 041766 000137 042622
9680
9681 041772
9682
9683
9684
9685
9686
9687 041772 122737 000011 001406
9688 042000 103451
9689
9690
9691
9692 042002 032737 140000 105530
9693 042010 001410
9694
9695 042012 012737 002012 001406
9696 042020 012737 000003 042626
9697 042026 000137 041104
9698 042032
9699
9700
9701 042032 010046
9702 042034 010146
9703 042036 012701 000374
9704 042042 012700 000014
9705 042046 012760 177777 105530
9706 042054 012760 177777 106540
9707 042062 062700 000002
9708 042066 005301
9709 042070 001366
9710 042072 012701 000006

220$: BIC #PAR,(SP) ;YES - CLEAR PAR
TST (SP)+ ;ARE THERE ANY ERRORS NOT DUE TO MEDIA ?
BNE 230$ ;YES !!
MOV RMER2I,-(SP) ;GET ER2
BIC #BSE!DPE,(SP)+ ;CLEAR MEDIA ERRORS
BNE 230$ ;BRANCH IF NONMEDIA ERRORS DETECTED

;THE ERRORS DETECTED WHILE TRYING TO RECOVER THE BAD SECTOR FILE ARE
;DUE TO THE MEDIA. SEE IF THE BAD SECTOR FILE CAN BE RECOVERED FROM
;ANOTHER AREA ON THE LAST TRACK

225$: ADD #2,RMDAO ;ADVANCE SECTOR ADDRESS
CMPB #10.,RMDAO ;QUIT IF ALL MFG SECTORS HAVE BEEN
BEQ 230$ ;TRIED
CMPB #32.,RMDAO ;QUIT IF ALL USER SECTORS HAVE
BEQ 230$ ;BEEN TRIED
MOV #3,500$ ;REINSTATE RETRY COUNT FOR THIS SECTOR
SUB #6,(SP) ;MOVE SP BACK TO NO ERROR
JMP 30$ ;RETRY THE READ OPERATION

230$:
;THE BAD SECTOR FILE CANNOT BE READ
SUB #4,(SP) ;MOVE SP TO ERROR RETURN
JMP 410$ ;GO TO MODULE EXIT

240$:
;THE SECTOR WAS RECOVERED WITHOUT ERROR -
; READ THE USER FILE IF THIS IS THE MFG FILE
; OR ELSE DONE

CMPB #9.,RMDAO ;WAS THE USER FILE READ ??
BLO 260$ ;YES - READ IS COMPLETE

;IF 144 IS IMPLEMENTED THEN READ THE USER FILE -
; ELSE DUMMY THE BAD SECTOR FILE
BIT #MSE!USE,MFGFIL ;ARE THE BAD SECTOR FLAGS OFF ??
BEQ 250$ ;YES - 144 IS DISABLED

MOV #002012,RMDAO ;READ THE USER FILE
MOV #3,500$ ;RELOAD THE RETRY COUNT FOR THIS SECTOR
JMP 30$ ;GO READ THE USER FILE

250$:
;DUMMY THE BAD SECTOR FILES
MOV R0,-(SP) ;;PUSH R0 ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV #252.,R1 ;R1 = NUMBER OF ENTRIES IN FILES
MOV #14,R0 ;R0 = ADDRESS INDEX TO FILE STORAGE
255$: MOV #-1,MFGFIL(R0) ;ENTER ALL ONES IN MFG FILE
MOV #-1,USRFIL(R0) ;ENTER ALL ONES IN USER FILE
ADD #2,R0 ;ADVANCE ADDRESS
DEC R1 ;DECREMENT COUNT
BNE 255$ ;CONTINUE IF NOT DONE
MOV #6.,R1 ;CLEAR ID, SERIAL NUMBER ETC

```

```

9711 042076 005000
9712 042100 005060 105530
9713 042104 005060 106540
9714 042110 062700 000002
9715 042114 005301
9716 042116 001370
9717
9718 042120 012601
9719 042122 012600
9720 042124
260$:
9721
9722
9723 042124 012737 177777 001474
9724
9725 042132 010046
9726 042134 005000
9727 042136 016060 107550 001400 265$:
9728 042144 062700 000002
9729 042150 022700 000046
9730 042154 103370
9731
9732 042156 012600
9733 042160
270$:
9734
9735 042160
300$:
9736
9737
9738
9739
9740
9741
9742
9743 042160 010046
9744 042162 010146
9745 042164 010246
9746 042166 013737 001434 001476
9747 042174 013737 001406 001500
9748 042202 005002
9749 042204 013700 001402
9750 042210 005300
9751 042212 005100
9752 042214 012701 000400
9753 042220 032737 000002 001400
9754 042226 001402
9755 042230 012701 000402
9756 042234 020100
305$:
9757 042236 101404
9758 042240 005700
9759 042242 001405
9760 042244 005202
9761 042246 000403
9762 042250 160100
310$:
9763 042252 005202
9764 042254 000767
9765 042256 010237 042626
315$:
9766

```

```

CLR RO
CLR MFGFIL(RO)
CLR USRFIL(RO)
ADD #2,RO
DEC R1
BNE 256$

MOV (SP)+,R1 ;;POP STACK INTO R1
MOV (SP)+,RO ;;POP STACK INTO RO

;SET MEDIA ENABLE AND RESTORE THE USERS PUT BUFFER
MOV #1,MEDENB

MOV RO,-(SP) ;;PUSH RO ON STACK
CLR RO ;;RO IS REGISTER INDEX
MOV BUFFER(RO),PUTBUF(RO)
ADD #2,RO ;;ADVANCE RO
CMP #46,RO ;;DONE ??
BHIS 265$

MOV (SP)+,RO ;;POP STACK INTO RO

;*****
;VERIFY THAT THE DESIRED SECTOR IS NOT IN THE MFG BAD SECTOR FILE
;AND NOT IN THE USERS BAD SECTOR FILE. ASSIGN A NEW SECTOR IF THE
;SECTOR IS IN EITHER FILE.

;LOAD INITIAL VARIABLES AND COMPUTE THE NUMBER OF SECTORS
MOV RO,-(SP) ;;PUSH RO ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV R2,-(SP) ;;PUSH R2 ON STACK
MOV RMDCO,ASMDC ;;LOAD REQUESTED CYLINDER, TRACK,
MOV RMDAO,ASMDA ;;AND SECTOR ADDRESS IN ASSIGNED STORAGE
CLR R2 ;;R2 = NUMBER OF SECTORS
MOV RMDCO,RO ;;RO = WORD COUNT
DEC RO ;;CONVERT FROM 2'S COMPLEMENT
COM RO
MOV #256,R1 ;;R1 = NUMBER OF WORDS PER SECTOR
BIT #BIT1,RMCS10 ;;IS THIS A HEADER AND DATA COMMAND ??
BEQ 305$ NO !!
MOV #258,R1 ;;CHANGE WORDS PER SECTOR
CMP R1,RO ;;IS THERE A FULL SECTOR ??
BLOS 310$ YES !!
TST RO ;;IS RO ZERO ??
BEQ 315$ YES !!
INC R2 ;;INCREMENT FOR PARTIAL SECTOR
BR 315$
SUB R1,RO ;;SUBTRACT ONE SECTOR FROM WORD COUNT
INC R2 ;;INCREMENT SECTOR COUNT
BR 305$
MOV R2,500$ ;;SAVE SECTOR COUNT

```



```

9767 042262 316$:
9768
9769 ;LOAD PARAMETERS AND SEARCH THE MFG AND USER BAD SECTOR FILES FOR
9770 ;THE ASSIGNED SECTOR AND ADJACENT SECTORS IF THE SECTOR COUNT IS
9771 ;MORE THAN ONE.
9772
9773 042262 012737 105544 042636 MOV #MFGFIL+14,540$ ;MOVE THE STARTING ADDRESS OF MFG
9774 ;FILE TO BASE ADDRESS STORAGE
9775 042270 013737 001476 042632 320$: MOV ASNDC,520$ ;LOAD COMPARING CYLINDER ADDRESS
9776 042276 013737 001500 042634 MOV ASNDA,530$ ;LOAD COMPARING TRACK, SECTOR ADDRESS
9777 042304 013737 042626 042630 MOV 500$,510$ ;LOAD NUMBER OF SECTORS TO CONFIRM
9778
9779 ;SETUP FOR A BINARY SEARCH OF THE CURRENT FILE FOR THE COMPARING
9780 ;CYLINDER, TRACK AND SECTOR ADDRESS
9781 042312 013700 042636 325$: MOV 540$,R0 ;LOAD THE BASE ADDRESS IN R0
9782 042316 012701 000376 MOV #((127.*4)/2),R1 ;R1 = LENGTH OF FILE
9783 042322 060100 ADD R1,R0 ;START BINARY DIVIDE AT CENTER
9784 042324 021037 042632 330$: CMP (R0),520$ ;DOES TABLE ENTRY = COMPARING CYLINDER ?
9785 042330 001405 BEQ 345$ ;YES !!
9786 042332 101002 BHI 340$ ;BRANCH IF ENTRY > COMPARING CYLINDER
9787 042334
9788 335$:
9789 ;FILE ENTRY IS LESS THAN COMPARING CYLINDER (OR TRACK AND SECTOR),
9790 ;SO ADD DISPLACEMENT TO CURRENT POSITION
9791 042334 060100 ADD R1,R0
9792 042336 000410 BR 350$
9793 042340 340$:
9794
9795 ;FILE ENTRY IS GREATER THAN COMPARING CYLINDER SO SUBTRACT DISPLACEMENT
9796 ;FROM CURRENT POSITION
9797 042340 160100 SUB R1,R0
9798 042342 000406 BR 350$
9799 042344 345$:
9800
9801 ;FILE ENTRY EQUALS COMPARING CYLINDER. SEE IF THE NEXT ENTRY EQUALS
9802 ;THE COMPARING TRACK, AND SECTOR.
9803 042344 026037 000002 042634 CMP 2(R0),530$ ;ARE THEY EQUAL ??
9804 042352 001413 BEQ 360$ ;YES !!
9805 042354 101371 BHI 340$ ;BRANCH IF HIGHER
9806 042356 000766 BR 335$ ;BRANCH IF LOWER
9807 042360 350$:
9808
9809 ;THE POINTER (R0) HAS BEEN ADJUSTED. HALVE THE DISPLACEMENT AND
9810 ;CONTINUE THE SEARCH IF DISPLACEMENT NOT ZERO
9811 042360 005701 TST R1 ;IS DISPLACEMENT ZERO ??
9812 042362 001440 BEQ 370$ ;YES !!
9813 042364 006201 ASR R1 ;HALVE THE DISPLACEMENT
9814 042366 042701 000003 BIC #3,R1
9815 042372 020037 042636 CMP R0,540$ ;IS THE NEW POINTER WITHIN BOUNDS ??
9816 042376 103432 BLO 370$ ;NO !!
9817 042400 000751 BR 330$ ;CONTINUE SEARCH
9818 042402 360$:
9819
9820 ;THE COMPARING CYLINDER, TRACK AND SECTOR IS IN THE BAD SECTOR FILE.
9821 ;ADVANCE THE ASSIGNED SECTOR AND START THE SEARCH ALL OVER.
9822 042402 105237 001500 INCB ASNDA ;INCREMENT SECTOR

```

```

9823 042406 122737 000037 001500          CMPB    #31.,ASNDA      ;SECTOR OK ??
9824 042414 103022                BHIS    365$          ;YES !!
9825 042416 105037 001500          CLRB    ASNDA        ;CLEAR SECTOR AND ADVANCE TRACK
9826 042422 105237 001501          INCB    ASNDA+1
9827 042426 122737 000004 001501          CMPB    #4,ASNDA+1    ;TRACK OK ??
9828 042434 103012                BHIS    365$          ;YES !!
9829 042436 005037 001500          CLR     ASNDA        ;CLEAR TRACK AND SECTOR
9830 042442 005237 001476          INC     ASNDC        ;INCREMENT CYLINDER
9831 042446 022737 001466 001476          CMP     #822.,ASNDC   ;CYLINDER OK ??
9832 042454 103002                BHIS    365$          ;YES !!
9833 042456 005037 001476          CLR     ASNDC
9834 042462 000677                365$:  BR            316$ ;REPEAT SEARCH
9835
9836 042464                370$:
9837
9838 ;THE COMPARING SECTOR IS NOT IN THE BAD SECTOR FILES. DECREMENT THE
9839 ;NUMBER OF SECTORS TO COMPARE AND SEARCH THE NEXT SECTOR IF THE NUMBER
9840 ;IS NOT ZERO.
9841
9842 042464 005337 042630          DEC     510$         ;DECREMENT NUMBER OF SECTORS TO COMPARE
9843 042470 001432                BEQ     380$         ;DONE IF ZERO
9844 042472 105237 042634          INCB    530$         ;INCREMENT THE COMPARING SECTOR
9845 042476 122737 000037 042634          CMPB    #31.,530$    ;SECTOR OK ??
9846 042504 103022                BHIS    375$         ;YES !!
9847 042506 105037 042634          CLRB    530$         ;CLEAR SECTOR
9848 042512 105237 042635          INCB    530$+1      ;INCREMENT TRACK
9849 042516 122737 000004 042635          CMPB    #4,530$+1    ;TRACK OK ??
9850 042524 103012                BHIS    375$         ;YES !!
9851 042526 005037 042634          CLR     530$         ;CLEAR SECTOR TRACK
9852 042532 005237 042632          INC     520$         ;INCREMENT CYLINDER
9853 042536 022737 001466 042632          CMP     #822.,520$   ;CYLINDER OK ??
9854 042544 103002                BHIS    375$         ;YES !!
9855 042546 005037 042632          CLR     520$         ;CLEAR CYLINDER
9856 042552 000137 042312          375$:  JMP            325$ ;SEARCH NEXT SECTOR
9857
9858 042556                380$:
9859
9860 ;THE ASSIGNED SECTOR (AND REQUIRED ADJACENT SECTORS) ARE NOT IN
9861 ;THE BAD SECTOR FILE JUST SEARCHED. SEARCH THE USER FILE IF IT
9862 ;HAS NOT YET BEEN SEARCHED, ELSE ASSIGN THE SECTOR AND RETURN TO USER.
9863
9864 042556 022737 106554 042636          CMP     #USRFIL+14,540$ ;TEST BASE ADDRESS
9865 042564 001405                BEQ     400$         ;DONE IF BASE = USER
9866 042566 012737 106554 042636          MOV     #USRFIL+14,540$ ;LOAD BASE ADDRESS
9867 042574 000137 042270          390$:  JMP            320$ ;SEARCH USER FILE
9868
9869
9870 042600                400$:
9871
9872 ;ASSIGN THE SECTOR AND RETURN TO USER
9873 042600 013737 001476 001434          MOV     ASNDC,RMDCO  ;LOAD CYLINDER
9874 042606 013737 001500 001406          MOV     ASNDA,RMDAO  ;LOAD TRACK AND SECTOR
9875 042614 012602                MOV     (SP)+,R2     ;POP STACK INTO R2
9876 042616 012601                MOV     (SP)+,R1     ;POP STACK INTO R1
9877 042620 012600                MOV     (SP)+,R0     ;POP STACK INTO R0
9878 042622 000240          410$:  NOP

```

9879 042624 000207
9880
9881
9882
9883 042626 000000
9884 042630 000000
9885 042632 000000
9886 042634 000000
9887 042636 000000

RTS PC

;THE FOLLOWING ARE STORAGE LOCATIONS FOR THE MODULE

500\$: .WORD ;RETRY COUNT/ NUMBER OF SECTORS REQUIRED
510\$: .WORD ;NUMBER OF SECTORS TO COMPARE
520\$: .WORD ;COMPARING CYLINDER
530\$: .WORD ;COMPARING TRACK AND SECTOR
540\$: .WORD ;BASE ADDRESS OF BAD SECTOR FILE BEING SEARCHED

9888
9889
9890
9891
9892
9893
9894
9895
9896
9897
9898
9899
9900
9901
9902
9903
9904
9905
9906
9907
9908
9909
9910
9911
9912
9913
9914
9915
9916
9917
9918
9919
9920
9921
9922
9923
9924
9925
9926
9927
9928
9929
9930
9931
9932
9933
9934
9935
9936
9937
9938
9939
9940
9941
9942
9942

042640
042640 010046
042642 010146
042644 010246
042646 010346
042650 010446
042652 013700 001404
042656 013701 001402
042662 013737 001434 043100
042670 013737 001406 043102
042676 032737 000002 001400 10\$:
042704 001450
042706 013710 043100
042712 042710 176000
042716 052710 140000
042722 012702 000035
042726 032737 010000 001432
042734 001404
042736 052710 010000
042742 012702 000037
042746 005201 15\$:
042750 001444
042752 062700 000002
042756 013720 043102
042762 005201
042764 001436
042766 012703 043102
042772 105213
042774 120213
042776 103013
043000 105013
043002 105263 000001
043006 122763 000004 000001
043014 103004
043016 105063 000001
043022 105237 043100
043026 012704 000400 25\$:
043032 013702 001174 30\$:
043036 013703 001176
043042 012220 40\$:
043044 005201
043046 001405
043050 005304

```
.SBTTL BUFFER GENERATOR SUBROUTINE
; THIS SUBROUTINE GENERATES A DATA BUFFER FOR WRITE COMMANDS. THE
; BUFFER STARTS AT RMBR AND IS RMWC WORDS LONG. THE BUFFER
; CONTAINS A REPETITIVE DATA PATTERN CONSISTING OF $TMP1 WORDS
; FROM THE DATA PATTERN TABLE, BEGINNING AT ADDRESS $TMP0.
; HEADER INFORMATION FOR THE BUFFER IS EXTRACTED FROM RMDC,
; RMDA AND RMOF.
;CALL:
;(1) JSR PC,GENBUF
;(2) ?? RETURN HFRE
GENBUF:
MOV RO,-(SP) ;:PUSH RO ON STACK
MOV R1,-(SP) ;:PUSH R1 ON STACK
MOV R2,-(SP) ;:PUSH R2 ON STACK
MOV R3,-(SP) ;:PUSH R3 ON STACK
MOV R4,-(SP) ;:PUSH R4 ON STACK
MOV RMDA,R0 ;:LOAD DATA BUFFER ADDRESS
MOV RMWC,R1 ;:LOAD WORD COUNT
MOV RMDCO,60$ ;:LOAD STARTING CYLINDER ADDRESS
MOV RMDAO,65$ ;:LOAD STARTING TRACK,SECTOR ADDRESS
BIT #BIT1,RMCS10 ;:WRITE HEADER & DATA??
BEQ 25$ ;:NO!!
MOV 60$,(R0) ;:WRITE HEADER WORD #1
BIC #CCYLMSK,(R0) ;:SET (DISABLE) BAD SECTOR FLAGS
BIS #MSE!USE,(R0) ;:R2 = MAXIMUM SECTOR ADDRESS
MOV #29,R2 ;:16 BIT FORMAT??
BIT #FMT16,RMOF0 ;:NO!!
BEQ 15$ ;:SET FORMAT BIT IN HEADER
BIS #FMT16,(R0) ;:CHANGE MAXIMUM SECTOR ADDRESS
MOV #31.,R2 ;:INCREMENT WORD COUNT
INC R1 ;:EXIT IF DONE
BEQ 50$ ;:MOVE RO TO HEADER WORD #2
ADD #2,RO ;:WRITE HEADER WORD #2
MOV 65$,(RO)+ ;:INCREMENT WORD COUNT AND
INC R1 ;:EXIT IF DONE
BEQ 50$ ;:ADVANCE SECTOR ADDRESS
MOV #65$,R3 ;:SECTOR OVERFLOW ??
INCB (R3) ;:NO !!
CMPB R2,(R3) ;:YES - CLEAR SECTOR ADDRESS
BHS 25$ ;:ADVANCE TRACK ADDRESS
CLRB (R3) ;:TRACK OVERFLOW ??
INCB 1(R3) ;:NO !!
CMPB #4,1(R3) ;:YES - CLEAR TRACK ADDRESS
BHS 25$ ;:ADVANCE CYLINDER ADDRESS
CLRB 1(R3) ;:LOAD SECTOR DATA COUNT
INCB 60$ ;:LOAD PATTERN ADDRESS
MOV #256.,R4 ;:LOAD PATTERN COUNT
MOV $TMP0,R2 ;:WRITE DATA PATTERN
MOV $TMP1,R3 ;:INCREMENT WORD COUNT AND
MOV (R2)+,(R0)+ ;:EXIT IF DONE
INC R1 ;:DECREMENT SECTOR COUNT
BEQ 50$
DEC R4
```

9944	043052	001711	BEQ	10\$; START NEXT SECTOR IF 0
9945	043054	005303	DEC	R3	; DECREMENT PATTERN COUNT
9946	043056	001765	BEQ	30\$; RESTART PATTERN IF 0
9947	043060	000770	BR	40\$; CONTINUE DATA PATTERN
9948					
9949	043062		50\$:		
9950	043062	012604	MOV	(SP)+,R4	:: POP STACK INTO R4
9951	043064	012603	MOV	(SP)+,R3	:: POP STACK INTO R3
9952	043066	012602	MOV	(SP)+,R2	:: POP STACK INTO R2
9953	043070	012601	MOV	(SP)+,R1	:: POP STACK INTO R1
9954	043072	012600	MOV	(SP)+,R0	:: POP STACK INTO R0
9955	043074	000240	NOP		
9956	043076	000207	RTS	PC	
9957					
9958	043100	000000	60\$:	.WORD	; CYLINDER ADDRESS STORAGE
9959	043102	000000	65\$:	.WORD	; TRACK, SECTOR ADDRESS STORAGE

.SBTTL COMPARE BUFFER SUBROUTINE

```

; THIS SUBROUTINE COMPARES THE CONTENTS OF BUFONE AND BUFTWO
; ASSUMING THAT BUFONE IS THE BUFFER FROM WHICH DATA WAS WRITTEN
; AND BUFTWO IS THE BUFFER TO WHICH DATA WAS READ. ERRORS IN BUFFER
; COMPARISON ARE REPORTED TO THE USER VIA THE USER'S ERROR CALL.

```

;CALL:

```

(1) JSR PC,CMPBUF
(2) .WORD WRITE BUFFER ADDRESS
(3) .WORD READ BUFFER ADDRESS
(4) BR ?? RETURN HERE IF NO ERROR
(5) NOP RETURN HERE IF ERROR
(6) ERROR ERROR DEFINED BY SUBROUTINE

```

CMPBUF:

```

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

CLR 150$ ;: CLEAR CORRECTION FLAG
; DETERMINE IF DATA SHOULD BE CORRECTED
BIT #DCK,RMER11 ;: WAS THERE A DATA CHECK ??
BEQ 80$ ;: NO !!
BIT #ECH,RMER11 ;: IS IT A HARD ERROR ??
BNE 80$ ;: YES !!
BIT #ECI,RMOFI ;: WAS ECC CORRECTION ALLOWED ??
BNE 80$ ;: NO !!
BIT #FMT16,RMOFI ;: IS THIS 16 BIT FORMAT ??
BEC 80$ ;: NO !!

; CORRECT DATA USING ECC INFORMATION
MOV RMBAD,R0 ;: R0 = STARTING BUFFER ADDRESS
MOV RMEC11,R1 ;: R1 = ECC POSITION
BIS #BIT15,150$ ;: SET CORRECTION FLAG
; MOVE R0 TO WORD BOUNDARY OF ERROR BURST
10$: CMP #16.,R1 ;: IS BIT POSITION > 1 WORD
BHIS 20$ ;: NO !!
SUB #16.,R1 ;: SUBTRACT 1 WORDS WORTH
ADD #2,R0 ;: ADVANCE BUFFER ADDRESS 1 WORD
BR 10$
20$: MOV #1,R2 ;: R2 = BIT POINTER
MOV R2,R3 ;: R3 = BIT NUMBER
; MOVE R2 TO STARTING BIT OF ERROR BURST
30$: CMP R3,R1 ;: IS R3 SAME AS R1 ??
BEQ 35$ ;: YES !!
ASL R2 ;: SHIFT BIT POINTER
INC R3 ;: INCREMENT BIT NUMBER
BR 30$
35$: MOV #11.,R3 ;: R3 = LENGTH OF ERROR BURST
; CORRECT THE ERROR BURST

```

```

9960
9961
9962
9963
9964
9965
9966
9967
9968
9969
9970
9971
9972
9973
9974
9975
9976 043104
9977 043104 010046
9978 043106 010146
9979 043110 010246
9980 043112 010346
9981
9982 043114 005037 043450
9983
9984 043120 032737 100000 001344
9985 043126 001465
9986 043130 032737 000100 001344
9987 043136 001061
9988 043140 033737 004000 001362
9989 043146 001055
9990 043150 032737 010000 001362
9991 043156 001451
9992
9993
9994 043160 013700 001404
9995 043164 013701 001374
9996 043170 052737 100000 043450
9997
9998
9999 043176 022701 000020
10000 043202 103005
10001 043204 162701 000020
10002 043210 062700 000002
10003 043214 000770
10004 043216 012702 000001
10005 043222 010203
10006
10007
10008 043224 020301
10009 043226 001403
10010 043230 006302
10011 043232 005203
10012 043234 000773
10013 043236 012703 000013
10014
10015

```

```

10016 043242 030237 001376      40$: BIT R2,RMEC2I ;IS THIS BIT SET IN ECC PATTERN ??
10017 043246 001405      BEQ 60$ ;NO - DO NOT CORRECT THIS BIT
10018 043250 030210      BIT R2,(R0) ;IS THE BIT PRESENTLY SET ??
10019 043252 001402      BEQ 50$ ;NO
10020 043254 040210      BIC R2,(R0) ;RESET THE BIT
10021 043256 000401      BR 60$
10022 043260 050210      50$: BIS R2,(R0) ;SET THE BIT
10023 043262 006302      60$: ASL R2 ;SHIFT TO NEXT BIT
10024 043264 001004      BNE 70$
10025 043266 012702 000001      MOV #1,R2 ;CONTINUE WITH FIRST BIT OF NEXT WORD
10026 043272 062700 000002      ADD #2,R0
10027 043276 005303      70$: DEC R3 ;END OF BURST ??
10028 043300 001360      BNE 40$ ;NO !!
10029 043302 017600 000010      80$: MOV #10(SP),R0 ;RO = WRITE BUFFER
10030 043306 062766 000002 000010      ADD #2,10(SP) ;MOVE SP TO READ ADDRESS
10031 043314 017601 000010      MOV #10(SP),R1 ;R1 = READ BUFFER
10032 043320 062766 000002 000010      ADD #2,10(SP) ;MOVE SP TO RETURN ADDRESS
10033 043326 013702 001332      MOV RMWCI,R2 ;R2 = NUMBER OF WORDS TRANSFER
10034 043332 163702 001402      SUB RMWCO,R2
10035 043336 022021      90$: CMP (R0)+,(R1)+ ;COMPARE DATA WORDS
10036 043340 001003      BNE 100$ ;EXIT IF NOT EQUAL
10037 043342 005302      DEC R2 ;DECREMENT WORD COUNT
10038 043344 001374      BNE 90$ ;CONTINUE IF NOT DONE
10039 043346 000433      BR 110$ ;DONE COMPARE - NO ERROR
10040 043350      100$:
10041 043350 014037 001140      MOV -(R0),SGDDAT ;STORE GOOD DATA FOR TYPEOUT
10042 043354 014137 001142      MOV -(R1),SBDDAT ;STORE BAD DATA FOR TYPEOUT
10043 043360 010037 001134      MOV R0,SGDAOR ;STORE ADDRESS OF GOOD DATA
10044 043364 010137 001136      MOV R1,SBDAOR ;STORE ADDRESS OF BAD DATA
10045 043370 010237 001174      MOV R2,STMPO ;STORE WORD COUNT OF ERROR
10046 043374 062766 000004 000010      ADD #4,10(SP) ;MOVE SP TO USER'S ERROR CALL
10047 043402 112776 000336 000010      MOVB #336,10(SP) ;WRITE ERROR NUMBER IN CALL
10048
10049
10050 043410 032737 100000 043450 ;CHANGE ERROR NUMBER IF ECC CORRECTION FAILED
10051 043416 001405      BIT #BIT15,150$ ;WAS ECC CORRECTION USED ??
10052 043420 112776 000163 000010      BEQ 105$ ;NO !!
10053 043426 162766 000002 000010      MOVB #163,10(SP) ;ECC CORRECTION FAILED
10054 043434 000240      105$: SUB #2,10(SP) ;MOVE SP TO RETURN IF ERROR
10055 043436      NOP
10056 043436 012603      110$:
10057 043440 012602      MOV (SP)+,R3 ;POP STACK INTO R3
10058 043442 012601      MOV (SP)+,R2 ;POP STACK INTO R2
10059 043444 012600      MOV (SP)+,R1 ;POP STACK INTO R1
10060 043446 000207      MOV (SP)+,R0 ;POP STACK INTO R0
10061      RTS PC ;RETURN TO USER
10062 043450 000000      150$: .WORD ;ECC CORRECTION FLAG
    
```

```

10063
10064
10065
10066
10067
10068
10069
10070
10071
10072 013452
10073 013454 010046
10074 013454 010146
10075 013456 010246
10076 013460 012700 001506
10077 013464 012701 001400
10078 013470 012702 000046
10079 013474 110220
10080 013476 005041
10081 013500 162702 000002
10082 013504 100405
10083 013506 022702 000022
10084 013512 001370
10085 013514 005041
10086 013516 000770
10087 013520 112720 000200
10088 013524 012602
10089 013526 012601
10090 013530 012600
10091 013532 000240
10092 013534 000207
10093

```

```

.SBTTL GET STATUS SUBROUTINE
; THIS SUBROUTINE SETS UP THE "GET INDEX TABLE" AND THE "GET
; BUFFER" FOR READING ALL SUBSYSTEM REGIS. RS VIA THE GET SUBROUTINE
; AND THEN RETURNS TO THE USER.
; CALL: JSR PC,GETSTS RETURN HERE
;
GETSTS:
MOV R0,-(SP) ;: PUSH R0 ON STACK
MOV R1,-(SP) ;: PUSH R1 ON STACK
MOV R2,-(SP) ;: PUSH R2 ON STACK
MOV #GETINX,R0 ;: R0= ADDRESS OF INDEX TABLE
MOV #RMEC2I+2,R1 ;: R1 = ADDRESS OF GET BUFFER
MOV #RMEC2,R2 ;: R2 = REGISTER INDE
2$: MOVB R2,(R0)+ ;: WRITE REGISTER INDEX IN TABLE
CLR -(R1) ;: CLEAR CORRESPONDING LOCATION
3$: SUB #2,R2 ;: DECREMENT TO NEXT INDEX
BMI 4$ ;: BRANCH OUT IF DONE
CMP #RMDB,R2 ;: DONT WRITE RMDB INDEX
BNE 2$
CLR -(R1)
BR 3$
4$: MOVB #200,(R0)+ ;: WRITE TERMINATOR
MOVB (SP)+,R2 ;: POP STACK INTO R2
MOV (SP)+,R1 ;: POP STACK INTO R1
MOV (SP)+,R0 ;: POP STACK INTO R0
NOP
RTS PC

```



```

10094
10095
10096
10097
10098
10099
10100
10101
10102
10103
10104
10105
10106
10107
10108
10109
10110
10111
10112
10113
10114
10115
10116
10117
10118 043536 000240
10119 043540 062716 000004
10120 043544 105076 000000
10121 043550 162716 000004
10122 043554 010046
10123 043556 010146
10124 043560 010246
10125 043562 010346
10126 043564 010446
10127 043566 013746 000004
10128 043572 013746 000006
10129 043576 013700 001276
10130 043602 012702 001330
10131 043606 012704 001506
10132 043612 012737 043720 000004
10133 043620 012737 000300 000006
10134 043626 016037 000010 001174
10135 043634 016037 000000 001176
10136 043642 032737 004000 001176
10137 043650 001007
10138 043652 062766 000004 000016
10139 043660 112776 000112 000016
10140 043666 000423
10141 043670 105714
10142 043672 100433
10143 043674 111401
10144 043676 042701 177700
10145 043702 060001
10146 043704 112403
10147 043706 042703 177700
10148 043712 060203
10149 043714 011113

```

.SBTTL GET SUBROUTINE

```

; THIS SUBROUTINE READS THE REGISTERS WHICH ARE LISTED IN THE
; "GET INDEX TABLE" AND STORES THEIR VALUES IN THE CORRESPONDING
; LOCATION IN THE "GET REGISTER BUFFER". FOR EXAMPLE, AN
; ENTRY OF 04 IN THE TABLE WILL CAUSE THE SUBROUTINE TO
; READ "R0BA" AND STORE ITS CONTENTS AT THE LOCATION IN
; THE BUFFER ASSIGNED TO THAT REGISTER. THE NUMBER OF
; REGISTERS TO BE READ IS VARIABLE FROM 1 TO 22; THE INDEX
; TABLE MUST BE TERMINATED WITH A CONTROL BYTE (200)
; WHICH SHOULD FOLLOW THE LAST ENTRY.

```

SUBROUTINE CALL:

- (1) "GET INDEX TABLE" HAS BEEN LOADED WITH REGISTER INDEX VALUES AND TERMINATED WITH A CONTROL BYTE
- (2) "GET INPUT BUFFER" IS AVAILABLE FOR USE. (NOTE THAT UNUSED LOCATIONS, I.E., ENTRIES IN BUFFER CORRESPONDING TO REGISTERS NOT READ, ARE NOT CHANGED.)
- (3) JSR PC GET
BR ??? RETURN HERE IF NO ERROR FOUND
NOP RETURN HERE IF ANY ERROR FOUND
ERROR SUB DEFINES ERROR NUMBER
???

```

GET:  NOP
      ADD #4,(SP) ;CLEAR ERROR NUMBER IN USER'S
      CLRB 0(SP) ;ERROR CALL
      SUB #4,(SP)
      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 ERVVEC,-(SP) ;;PUSH ERVVEC ON STACK
      MOV ERVVEC+2,-(SP) ;;PUSH ERVVEC+2 ON STACK
      MOV $BASE,R0
      MOV #GETBUF,R2
      MOV #GETINX,R4
      MOV #5,ERVVEC ;SETUP FOR TIMEOUT
      MOV #PR6,ERVVEC+2
15:   MOV RMCS2(RO),STMP0 ;GET "MCS" STATUS
      MOV RMCS1(RO),STMP1 ;GET "DVA" STATUS
      BIT #DVA,STMP1 ;DEVICE AVAILABLE??
      BNE 3$ ;YES!!
      ADD #4,16(SP) ;WRITE ERROR NUMBER IN USER'S
      MOVB #12,216(SP) ;ERROR CALL
      BR 7$
35:   TSTB (R4) ;DONE??
      BMI 9$ ;YES!!
      MOVB (R4),R1 ;R1 = REGISTER ADDRESS
      BIC #1C10XMSK,R1 ;CLEAR ANY SIGN EXTENSION
      ADD R0,R1
      MOVB (R4)+,R3 ;R3 = STORAGE ADDRESS FOR REGISTER
      BIC #1C10XMSK,R3 ;CLEAR ANY SIGN EXTENSION
      ADD R2,R3
      MOVB (R1),(R3) ;READ REGISTER

```

```

10150 043716 000764 BR 3$
10151
10152 043720 022626 5$: CMP (SP)+,(SP)+ ;RESTORE STACK
10153 043722 062766 000004 000016 ADD #4,16(SP) ;WRITE ERROR NUMBER IN
10154 043730 112776 000007 000016 MOVB #7,216(SP) ;USER'S ERROR CALL
10155 043736 162766 000002 000016 7$: SUB #2,16(SP)
10156 043744 105714 8$: TSTB (R4) ;DONE CLEARING??
10157 043746 100405 BMI 9$ ;YES!!
10158 043750 005003 CLR R3 ;CLEAR REMAINING STORAGE
10159 043752 112403 MOVB (R4)+,R3 ;LOCATIONS
10160 043754 060203 ADD R2,R3
10161 043756 005013 CLR (R3)
10162 043760 000771 BR 8$
10163 043762
10164 043762 012637 000006 9$: MOV (SP)+,ERRVEC+2 ;POP STACK INTO ERRVEC+2
10165 043766 012637 000004 MOV (SP)+,ERRVEC ;POP STACK INTO ERRVEC
10166 043772 012604 MOV (SP)+,R4 ;POP STACK INTO R4
10167 043774 012603 MOV (SP)+,R3 ;POP STACK INTO R3
10168 043776 012602 MOV (SP)+,R2 ;POP STACK INTO R2
10169 044000 012601 MOV (SP)+,R1 ;POP STACK INTO R1
10170 044002 012600 MOV (SP)+,R0 ;POP STACK INTO R0
10171 044004 000207 RTS PC ;RETURN
10172

```

10173
10174
10175
10176
10177
10178
10179
10180
10181
10182
10183
10184
10185
10186
10187
10188
10189
10190
10191
10192
10193
10194
10195
10196
10197
10198
10199
10200
10201
10202
10203
10204
10205
10206
10207
10208
10209
10210
10211
10212
10213
10214
10215
10216
10217
10218
10219
10220
10221
10222
10223
10224
10225
10226
10227
10228

044006 000240
044010 010046
044012 010146
044014 010246
044016 010346
044020 010446
044022 013746 000004
044026 013746 000006
044032 013700 001276
044036 012702 001400
044042 012704 001535
044046 012737 044154 000004
044054 012737 000300 000006
044062 016037 000010 001174
044070 016037 000000 001176
044076 032737 004000 001176
044104 001007
044106 062766 000004 000016
044114 112776 000112 000016
044122 000423
044124 105714
044126 100424
044130 111401
044132 042701 177700
044136 060001
044140 112403
044142 042703 177700
044146 060203
044150 011311
044152 000764
044154 022626
044156 062766 000004 000016
044164 112776 000007 000016
044172 162766 000002 000016

.SBTTL PUT SUBROUTINE

: THIS SUBROUTINE WRITES THE REGISTERS WHICH ARE LISTED IN THE
: "PUT INDEX TABLE" WITH THE CONTENTS OF THE CORRESPONDING
: LOCATION IN THE "PUT REGISTER BUFFER". THE NUMBER OF
: REGISTERS WRITTEN IS VARIABLE; THE INDEX TABLE MUST
: BE TERMINATED WITH A CONTROL BYTE (200) WHICH SHOULD
: FOLLOW THE LAST ENTRY.

: SUBROUTINE CALL:

- : (1) "PUT INDEX TABLE" HAS BEEN LOADED WITH INDEX VALUES
: OF REGISTERS TO BE WRITTEN.
- : (2) "PUT REGISTER BUFFER" CONTAINS CONTENTS OF EACH
: REGISTER TO BE WRITTEN.
- : (3) JSR PC,PIJT
: BR ??? RETURN HERE IF NO ERROR FOUND
: NOP RETURN HERE IF ANY ERROR FOUND
: ERROR SUB DEFINES ERROR NUMBER
: ???

```

PUT:  NOP
      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    ERVEC,-(SP)   ;; PUSH ERVEC ON STACK
      MOV    ERVEC+2,-(SP) ;; PUSH ERVEC+2 ON STACK
      MOV    $BASE,R0
      MOV    #PUTBUF,R2
      MOV    #PUTINX,R4
      MOV    #5$,ERRVEC   ; SETUP FOR TIMEOUT
      MOV    #PR6,ERRVEC+2
1$:   MOV    RMCS2(R0),$TMP0 ; GET "NED" STATUS
      MOV    RMCS1(R0),$TMP1 ; GET "DVA" STATUS
      BIT    #DVA,$TMP1     ; DEVICE AVAILABLE??
      BNE    3$            ; YES!!
      ADD    #4,16(SP)      ; WRITE ERROR NUMBER IN
      MOVB  #12,216(SP)    ; USER'S ERROR CALL
      BR    7$
3$:   TSTB  (R4)           ; DONE??
      BMI  9$            ; YES!!
      MOVB  (R4),R1        ; R1 = REGISTER ADDRESS
      BIC  #1CIDXMSK,R1   ; CLEAR ANY SIGN EXTENSION
      ADD  R0,R1
      MOVB  (R4)+,R3       ; R3 = STORAGE ADDRESS
      BIC  #1CIDXMSK,R3   ; CLEAR ANY SIGN EXTENSION
      ADD  R2,R3
      MOV  (R3),(R1)      ; WRITE REGISTER
      BR    3$
5$:   CMP  (SP)+,(SP)+    ; ADJUST STACK
      ADD  #4,16(SP)      ; WRITE ERROR NUMBER IN
      MOVB  #7,216(SP)    ; USER'S ERROR CALL
7$:   SUB  #2,16(SP)

```

```

10229
10230 044200
10231 044200 012637 000006
10232 044204 012637 000004
10233 044210 012604
10234 044212 012603
10235 044214 012602
10236 044216 012601
10237 044220 012600
10238 044222 000207
10239

          95:
          MOV      (SP)+,ERRVEC+2      ;:POP STACK INTO ERRVEC+2
          MOV      (SP)+,ERRVEC        ;:POP STACK INTO ERRVEC
          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
          RTS      PC                  ;RETURN

```

```

10240          .SBTTL  SIZE CLOCK SUBROUTINE
10241
10242          SIZCLK:
10243          044224 013 46 000004      MOV    ERRVEC, -(SP)      ;; PUSH ERRVEC ON STACK
10244          044230 013746 000006      MOV    ERRVEC+2, -(SP)   ;; PUSH ERRVEC+2 ON STACK
10245          044234 012737 044272 000004      MOV    #15, ERRVEC      ;; SET UP FOR BUS TIMEOUT
10246          044242 012737 000300 000006      MOV    #PR6, ERRVEC+2
10247          044250 012737 177546 001502      MOV    #177546, CLKADR  ;; LOAD ADDRESSES FOR KW11-L
10248          044256 012737 000100 001504      MOV    #100, CLKVCT
10249          044264 005777 135212      TST    @CLKADR          ;; TEST FOR KW11-L PRESENT
10250          044270 000421                BR     3$              ;; YES - KW11-L IS PRESENT
10251          044272 022626                CMP    (SP)+, (SP)+    ;; RESTORE SP
10252          044274 012737 044324 000004      MOV    #25, ERRVEC     ;; SET UP FOR BUS TIMEOUT
10253          044302 012737 172540 001502      MOV    #172540, CLKADR ;; LOAD ADDRESSES FOR KW11-P CLOCK
10254          044310 012737 000104 001504      MOV    #104, CLKVCT
10255          044316 005777 135160      TST    @CLKADR          ;; TEST FOR KW11-P PRESENT
10256          044322 000404                BR     3$              ;; YES - KW11-P IS PRESENT
10257          044324 022626                CMP    (SP)+, (SP)+    ;; RESTORE SP
10258          044326 062766 000002 000004      ADD    #2, 4(SP)       ;; MOVE RETURN TO ERROR
10259          044334
10260          044334 012637 000006      MOV    (SP)+, ERRVEC+2 ;; POP STACK INTO ERRVEC+2
10261          044340 012637 000004      MOV    (SP)+, ERRVEC   ;; POP STACK INTO ERRVEC
10262          044344 000207                RTS                    ;; RETURN TO USER

```

```

.SBTTL TIMEOUT SUBROUTINE
: THIS SUBROUTINE WAITS FOR GO TO RESET OR FOR A TIMEOUT GREATER THAN
: 500 MS, WHICH EVER OCCURRS FIRST, AND THEN RETURNS.
: CALL: JSR PC, TIMEOUT RETURN HERE
:
TIMEOUT:
MOV RO, -(SP) ; PUSH RO ON STACK
MOV R1, -(SP) ; PUSH R1 ON STACK
MOV R2, -(SP) ; PUSH R2 ON STACK
MOV ERRVEC, -(SP) ; PUSH ERRVEC ON STACK
MOV ERRVEC+2, -(SP) ; PUSH ERRVEC+2 ON STACK
MOV #45, ERRVEC ; SETUP FOR BUS TIMEOUT - 04 TRAP
MOV #PR6, ERRVEC+2
MOV $BASE, RO ; RO=RM03 ADDRESS
MOV CLKADR, R1 ; R1=CLOCK ADDRESS
MOV #31, R2 ; R2=NUMBER OF CLOCK CYCLES
1$: CMP R1, #172540 ; KW11-P CLOCK??
BNE 2$ ; NO!!
MOV #1, 2(R1) ; SET COUNTER
2$: MOV #BIT2!BIT0, (R1) ; START COUNTER
3$: MOV RMCS1(RO), -(SP) ; GET STATUS
BIC #C<RDY!GO>, (SP)
CMP #RDY, (SP)+ ; RDY=1, GO=0??
BEQ 5$ ; YES!!
BIT #BIT7, (R1) ; TIMER DONE??
BEQ 3$ ; NO!!
DEC R2 ; DEC NUMBER OF CYCLES
BNE 1$ ; CONTINUE IF NOT DONE
4$: CMP (SP)+, (SP)+ ; ADJUST STACK
ADD #4, 12(SP) ; MOVE SP TO USER'S CALL
MOV #7, 12(SP) ; WRITE ERROR NUMBER
SUB #2, 12(SP)
5$: MOV (SP)+, ERRVEC+2 ; POP STACK INTO ERRVEC+2
MOV (SP)+, ERRVEC ; POP STACK INTO ERRVEC
MOV (SP)+, R2 ; POP STACK INTO R2
MOV (SP)+, R1 ; POP STACK INTO R1
MOV (SP)+, RO ; POP STACK INTO RO
RTS PC ; RETURN TO USER

```

```

10263
10264
10265
10266
10267
10268
10269
10270
10271 044346
10272 044346 010046
10273 044350 010146
10274 044352 010246
10275 044354 013746 000004
10276 044360 013746 000006
10277 044364 012737 044466 000004
10278 044372 012737 000300 000006
10279 044400 013700 001276
10280 044404 013701 001502
10281 044410 012702 000037
10282 044414 020127 172540 1$:
10283 044420 001003
10284 044422 012761 000001 000002
10285 044430 012711 000005 2$:
10286
10287 044434 016046 000000 3$:
10288 044440 042716 177576
10289 044444 022726 000200
10290 044450 001420
10291 044452 032711 000200
10292 044456 001766
10293 044460 005302
10294 044462 001354
10295 044464 007412
10296 044466 022326 4$:
10297 044470 062766 000004 000012
10298 044476 112776 000007 000012
10299 044504 162766 000002 000012
10300
10301 044512 5$:
10302 044512 012637 000006
10303 044516 012637 000004
10304 044522 012602
10305 044524 012601
10306 044526 012600
10307 044530 000207
10308

```

10309
10310
10311
10312
10313
10314
10315
10316
10317
10318
10319
10320
10321
10322
10323
10324
10325
10326
10327
10328
10329
10330
10331
10332
10333
10334
10335
10336
10337
10338
10339
10340
10341
10342
10343
10344
10345
10346
10347 044532
10348
10349
10350 044532 062716 000004
10351 044536 105076 000000
10352 044542 162716 000004
10353
10354
10355
10356 044546 013737 001340 001142
10357 044554 042737 177770 001142
10358 044562 013737 001234 001140
10359 044570 042737 177770 001140
10360 044576 123737 001140 001142
10361
10362 044604 001415
10363 044606 062716 000004
10364 044612 112776 000001 000000

```

.SBTTL PRIMARY ERROR CHECK SUBROUTINE
: THE PURPOSE OF THIS SUBROUTINE IS TO VERIFY THAT STAUTS IS VALID AND
: THAT FURTHER ERROR AND STAT'S CHECKING SHOULD BE PERFORMED. THE
: FOLLOWING CHECKS ARE MADE:
:
: .CURRENT UNIT IS SELECTED, I.E., THE UNIT SELECT BITS OF RMCS2
: (BITS 0-2) EQUAL THE UNIT BEING TESTED;
:
: .SELECTED UNIT IS AVAILABLE, I.E., DVA (BIT 11 OF RMCS1) IS SET
: AND NED (BIT 12 OF RMCS2) IS RESET;
:
: .LAST COMMAND WAS COMPLETED, I.E., THE MASSBUS CONTROLLER IS
: READY (BIT 7 OF RMCS1) AND THE GO BIT IS RESET (BIT 0 OF RMCS1) OR THE
: DRIVE READY BIT (BIT 7 OF RMDS) IS SET.
:
: .NO PARITY ERROR OCCURRED WHEN READING REMOTE REGISTERS,
: I.E., MCPE = 0.
:
: .NO PARITY ERROR OCCURRED WHEN WRITING REMOTE REGISTERS,
: I.E., PAR = 0, OR, PAR = DPE = 1
:
: THE SUBROUTINE ASSUMES THAT:
:
: .STATUS HAS BEEN STORED IN THE REGISTER INPUT BUFFER,
: IN PARTICULAR, RMCS1, RMCS2 AND RMDS HAVE BEEN STORED IN THEIR
: CORRESPONDING LOCATIONS OF THE "GET" BUFFER.
:
: .($UNIT) CONTAINS THE DRIVE NUMBER
:
: THE SUBROUTINE IS CALLED AS FOLLOWS:
:
: (1) JSR PC,PRIERR          RETURN HERE IF NO ERROR
:     BR   ???              RETURN HERE TO REPORT AN ERROR
:     NOP                                ERROR NUMBER DEFINED BY SUB
:     ERROR                          GO BACK TO SUB FOR MORE ERROR CHECKS
:     JSR PC,@(SP)+          RETURN HERE IF NO MORE ERRORS
:     ???
:
: PRIERR:
: CLEAR USER'S ERROR CALL
: ADD #4,(SP)                ; MOVE (SP) TO ERROR CALL
: CLRB @ (SP)               ; CLEAR ERROR NUMBER
: SUB #4,(SP)               ; MOVE (SP) TO NO ERROR RETURN
:
: REPORT AN ERROR IF THE WRONG UNIT IS SELECTED
: MOV RMCS2I,$BDDAT        ; CORRECT UNIT SELECTED??
: BIC #1CUNTMSK,$BDDAT
: MOV $UNIT,$GDDAT        ; GOOD DATA FOR TYPEOUT
: BIC #1CUNTMSK,$GDDAT
: CMPB $GDDAT,$BDDAT      ; COMPARE EXPECTED AND RECEIVED
:                                ; DRIVE NUMBERS
: BEQ 1$                  ; YES!!
: ADD #4,(SP)
: MOVB #1,@(SP)           ; ERROR 1
    
```

```

10365 044620 162716 000002          SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
10366 044624 004736                JSR      PC,@(SP)+      ;REPORT WRONG UNIT SELECTED
10367 044626 162716 000010          SUB      #10,(SP)       ;RESTORE (SP)
10368 044632 000240                NOP
10369 044634 000137 045354          JMP      10$           ;SKIP OTHER CHECKS
10370 044640
10371
10372
10373          ;REPORT AN ERROR IF THE DEVICE IS NOT AVAILABLE OR IF
          ;THE DEVICE IS NONEXISTANT
10374 044640 032737 004000 001330      BIT      #DVA, RMCS1I    ;DEVICE AVAILABLE??
10375 044646 001045                BNE      5$            ;YES!!
10376 044650 013737 001330 001140      MOV      RMCS1I,$GDDAT  ;EXPECTED STATUS
10377 044656 052737 004000 001140      BIS      #DVA,$GDDAT
10378 044664 013737 001330 001142      MOV      RMCS1I,$BDDAT  ;RECEIVED STATUS
10379 044672 062716 000004          ADD      #4,(SP)
10380 044676 112776 000002 000000      MOV      #2,@(SP)      ;ERROR #2
10381 044704 032737 010000 001340      BIT      #NED, RMCS2I    ;WAS NED SET??
10382 044712 001414                BEQ      2$            ;NO!!
10383 044714 013737 001340 001140      MOV      RMCS2I,$GDDAT  ;EXPECTED STATUS
10384 044722 013737 001340 001142      MOV      RMCS2I,$BDDAT  ;RECEIVED STATUS
10385 044730 042737 010000 001140      BIC      #NED,$GDDAT
10386 044736 112776 000003 000000      MOV      #3,@(SP)      ;YES - CHANGE ERROR NUMBER
10387 044744 162716 000002          SUB      #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
10388 044750 004736                JSR      PC,@(SP)+      ;REPORT DEVICE NOT AVAILABLE
10389 044752 162716 000010          SUB      #10,(SP)      ;RESTORE (SP)
10390 044756 000240                NOP
10391 044760 000575                BR       10$           ;SKIP OTHER CHECKS
10392 044762
10393
10394          ;REPORT AN ERROR IF MASSBUS CONTROLLER IS NOT READY
10395 044762 032737 000200 001330      BIT      #RDY, RMCS1I    ;CONTROLLER READY??
10396 044770 001030                BNE      7$            ;YES!!
10397 044772 013737 001330 001140      MOV      RMCS1I,$GDDAT  ;EXPECTED STATUS
10398 045000 052737 000200 001140      BIS      #RDY,$GDDAT
10399 045006 042737 160001 001140      BIC      #SC!TRE!MCPE!GO,$GDDAT
10400 045014 013737 001330 001142      MOV      RMCS1I,$BDDAT  ;RECEIVED STATUS
10401 045022 062716 000004          ADD      #4,(SP)
10402 045026 112776 000004 000000      MOV      #4,@(SP)      ;ERROR #4
10403 045034 162716 000002          SUB      #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
10404 045040 004736                JSR      PC,@(SP)+      ;REPORT CONTROLLER NOT READY
10405 045042 162716 000010          SUB      #10,(SP)      ;RESTORE (SP)
10406 045046 000240                NOP
10407 045050 000541                BR       10$           ;SKIP OTHER CHECKS
10408 045052
10409
10410          ;REPORT AN ERROR IF GO IS NOT ZERO AND DRY IS NOT ONE
10411 045052 032737 000001 001330      BIT      #GO, RMCS1I     ;GO RESET??
10412 045060 001431                BEQ      8$            ;YES!!
10413 045062 032737 000200 001342      BIT      #DRY, RMDSI     ;DRIVE READY??
10414 045070 001025                BNE      8$            ;YES!!
10415 045072 013737 001330 001140      MOV      RMCS1I,$GDDAT  ;EXPECTED STATUS
10416 045100 042737 160001 001140      BIC      #SC!TRE!MCPE!GO,$GDDAT
10417 045106 013737 001330 001142      MOV      RMCS1I,$BDDAT  ;RECEIVED STATUS
10418 045114 062716 000004          ADD      #4,(SP)
10419 045120 112776 000005 000000      MOV      #5,@(SP)      ;ERROR #5
10420 045126 162716 000002          SUB      #2,(SP)      ;MOVE SP TO RETURN FOR ERROR

```



```

10421 045132 004736      JSR    PC,2(SP)+      ;REPORT DRIVE NOT READY
10422 045134 162716 000010    SUB    #10,(SP)      ;RESTORE (SP)
10423 045140 000240      NOP
10424 045142 000504      BR     10$
10425 045144
10426
10427      ;REPORT AN ERROR IF THE RM CONTROLLER DETECTED BAD
10428      ;PARITY ON THE MASSBUS CONTROL BUS
10429 045144 032737 020000 001330    BIT    #MCPE,RMCSI1  ;PARITY ERROR ??
10430 045152 001425      BEQ    9$           ;NO!!
10431 045154 013737 001330 001140    MOV    RMCSI1,$GDDAT ;EXPECTED STATUS
10432 045162 042737 160001 001140    BIC    #SC:TRÉ!MCPE!GO,$GDDAT
10433 045170 013737 001330 001142    MOV    RMCSI1,$BDDAT ;RECEIVED STATUS
10434 045176 062716 000004      ADD    #4,(SP)      ;MOVE STACK TO USER'S ERROR
10435 045202 112776 000013 000000    MOV    #13,2(SP)    ;ERROR #47
10436 045210 162716 000002      SUB    #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
10437 045214 004736      JSR    PC,2(SP)+      ;REPORT ERROR VIA USER
10438 045216 162716 000010    SUB    #10,(SP)     ;RESTORE STACK
10439 045222 000240      NOP
10440 045224 000453      BR     10$
10441 045226
10442
10443      ;REPORT AN ERROR IF THE RM03 DETECTED A CONTROL BUS PARITY ERROR
10444 045226 032737 000010 001344    BIT    #PAR,RMER11  ;WAS THERE A PARITY ERROR??
10445 045234 001451      BEQ    11$          ;NO!!
10446 045236 032737 000010 001372    BIT    #DPE,RMER2I  ;WAS IT THE CONTROL BUS??
10447 045244 001045      BNE    11$          ;NOT SURE!!
10448 045246 032737 000010 001414    BIT    #PAR,RMER10  ;DID TEST SET PAR ??
10449 045254 001413      BEQ    93$          ;NO!!
10450 045256 010046      MOV    RO,-(SP)     ;PUSH RO ON STACK
10451 045260 012700 001535      MOV    #PUTINX,RO   ;RO POINTS TO INDEX TABLE
10452 045264 122710 000014      CMP    #RMER1,(RO)  ;SEARCH TABLE FOR RMER1
10453 045270 001002      BNE    92$          ;
10454 045272 012600      MOV    (SP)+,RO     ;POP STACK INTO RO
10455 045274 000431      BR     11$          ;PAR WAS SET BY TEST
10456 045276 105720 92$:      TST    (RO)+        ;END OF TABLE??
10457 045300 100371      BPL    91$          ;NO!!
10458 045302 012600      MOV    (SP)+,RO     ;POP STACK INTO RO
10459 045304 013737 001344 001140 93$:      MOV    RMER11,$GDDAT ;EXPECTED STATUS
10460 045312 042737 000010 001140      BIC    #PAR,$GDDAT
10461 045320 013737 001344 001142      MOV    RMER11,$BDDAT ;RECEIVED STATUS
10462 045326 062716 000004      ADD    #4,(SP)      ;MOVE SP
10463 045332 112776 000050 000000    MOV    #50,2(SP)    ;WRITE THE ERROR NUMBER
10464 045340 162716 000002      SUB    #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
10465 045344 004736      JSR    PC,2(SP)+      ;REPORT THE ERROR
10466 045346 162716 000010      SUB    #10,(SP)     ;MOVE SP TO NO ERROR RETURN
10467 045352 000240      NOP
10468 045354 062716 000010 10$:      ADD    #10,(SP)     ;RETURN TO ERROR
10469 045360 000240 11$:      NOP                ;RETURN TO NO ERROR
10470 045362 000207      RTS
10471

```

```

10472
10473
10474
10475
10476
10477
10478
10479
10480
10481
10482
10483
10484
10485
10486
10487
10488
10489
10490
10491
10492
10493
10494
10495 045364
10496
10497
10498
10499 045364 013737 001400 051224
10500 045372 042737 177701 051224
10501 045400 062716 000004
10502 045404 105076 000000
10503 045410 162716 000004
10504
10505
10506
10507
10508
10509 045414 032737 000200 001342
10510 045422 001024
10511 045424 013737 001342 001142
10512 045432 042737 177577 001142
10513 045440 012737 000200 001140
10514 045446 062716 000004
10515 045452 112776 000010 000000
10516 045460 162716 000002
10517 045464 004736
10518 045466 162716 000010
10519 045472 000240
10520
10521
10522 045474 032737 000001 001330
10523 045502 001423
10524 045504 013737 001330 001142
10525 045512 042737 177776 001142
10526 045520 005037 001140
10527 045524 062716 000004

```

```

.SBTTL SECONDARY ERROR CHECK SUBROUTINE
;THE ERROR CHECK SUBROUTINE PROVIDES DETECTION OF SECONDARY ERRORS
;SUCH AS UNEXPECTED ERRORS AND UNEXPECTED REGISTER
;CONTENTS. THESE ERRORS ARE DEEMED
;SECONDARY IN THAT THEY ARE NOT NECESSARILY ASSOCIATED WITH THE OPERATION
;BEING PERFORMED. WHEN THE SUBROUTINE IDENTIFIES SUCH AN ERROR, IT MOVES
;THE ERROR NUMBER TO THE ERROR CALL IN THE TEST ROUTINE AND THEN RETURNS
;TO THE TEST ROUTINE WHICH MAKES THE ERROR CALL. AFTER THE TEST ROUTINE
;MAKES THE ERROR CALL, IT RETURNS TO THE SUBROUTINE WHICH THEN LOOKS FOR
;OTHER ERRORS. WHEN ALL ERRORS HAVE BEEN REPORTED, THE SUBROUTINE
;RETURNS TO THE ADDRESS FOLLOWING THE SUBROUTINE CALL.
CALL: JSP PC,SECERR
      BR ??? RETURN HERE IF NO ERROR
      NOP RETURN HERE TO REPORT AN ERROR
      ERROR ERROR NUMBER DEFINED BY SUB
      JSR PC,@(SP)+ GO BACK TO SUB FOR MORE ERROR CHECKS
      ??? RETURN HERE IF NO MORE ERRORS

;NOTE: THE SUBROUTINE ASSUMES THAT REGISTERS HAVE BEEN STORED AT THE
;INPUT REGISTER BUFFER.
SECERR:
;*****
;STORE FUNCTION CODE AND CLEAR USER'S ERROR NUMBER
MOV RMCS10,515$ ;STORE FUNCTION CODE
BIC #1C<F0!F1!F2!F3!F4>,515$
ADD #4,(SP) ;MOVE (SP) TO ERROR CALL
CLRB @(SP) ;CLEAR ERROR NUMBER
SUB #4,(SP) ;MOVE (SP) TO NO ERROR RETURN
;*****
;CHECK SECONDARY ERRORS COMMON TO ALL COMMANDS
;REPORT ERROR IF DRIVE IS NOT READY, I.E. IF DRY = 0
BIT #DRY,RMDSI ;DRIVE READY??
BNE $$ ;YES!!
MOV RMDSI,$BDDAT ;BAD DATA FOR TYPEOUT
BIC #1CDRY,$BDDAT
MOV #DRY,$GDDAT ;GOOD DATA FOR TYPEOUT
ADD #4,(SP)
MOVB #10,@(SP) ;ERROR NUMBER
SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
JSR PC,@(SP)+ ;REPORT NOT READY
SUB #10,(SP) ;RESTORE (SP) TO ERROR N
NOP
;REPORT ERROR IF GO BIT IS NOT RESET
SS: BIT #GO,RMCS11 ;GO BIT RESET??
BEQ 10$ ;YES!!
MOV RMCS11,$BDDAT ;BAD DATA FOR TYPEOUT
BIC #1CGO,$BDDAT
CLR $GDDAT ;GOOD DATA FOR TYPEOUT
ADD #4,(SP)

```

```

10528 045530 112776 000011 000000      MOVB    #11,2(SP)      ;ERROR NUMBER
10529 045536 162716 000002      SUB     #2,(SP)       ;MOVE SP TO RETURN FOR ERROR
10530 045542 004736      JSR     PC,2(SP)+     ;REPORT DEVICE NOT AVAILABLE
10531 045548 162716 000010      SUB     #10,(SP)      ;RESTORE (SP)
10532 045550 000240      NOP
10533
10534      ;REPORT ERROR IF FUNCTION CODE READ FROM DEVICE IS NOT CORRECT
10535 045552 013737 001330 001142 10$:  MOV     RMCS1I,$BDDAT ;IS FUNCTION CODE CORRECT??
10536 045560 042737 177701 001142    BIC     #1C76,$BDDAT
10537 045566 013737 051224 001140    MOV     5155,$GDDAT ;EXPECTED FUNCTION CODE
10538 045574 023737 001142 001140    CMP     $BDDAT,$GDDAT
10539 045602 001413      BEQ     15$           ;YES!!
10540 045604 062716 000004      ADD     #4,(SP)
10541 045610 112776 000012 000000    MOVB   #12,2(SP)     ;ERROR NUMBER
10542 045616 162716 000002      SUB     #2,(SP)       ;MOVE SP TO RETURN FOR ERROR
10543 045622 004736      JSR     PC,2(SP)+     ;REPORT WRONG FUNCTION CODE
10544 045624 162716 000010      SUB     #10,(SP)      ;RESTORE (SP)
10545 045630 000240      NOP
10546 045632
10547      15$:
10548      ;REPORT AN ERROR IF COMPOSITE ERROR IS SET AND NO OTHER
10549      ;ERRORS ARE SET, OR IF COMPOSITE ERROR IS NOT SET AND
10550      ;OTHER ERRORS ARE SET
10551 045632 005037 001140      CLR     $GDDAT       ;EXPECT "ERR"=0
10552 045636 005737 001344      TST     RMER1I       ;IS RMER1 =0??
10553 045642 001003      BNE     20$         ;NO!!
10554 045644 005737 001372      TST     RMER2I       ;IS RMER2=0??
10555 045652 001403      BEQ     25$         ;YES!!
10556 045658 052737 040000 001140 20$:  BIS     #ERR,$GDDAT   ;"ERR" SHOULD BE SET
10557 045660 013737 001342 001142 25$:  MOV     RMDSI,$BDDAT
10558 045666 012737 137777 001142    BIC     #1CERR,$BDDAT
10559 045674 023737 001140 001142    CMP     $GDDAT,$BDDAT ;IS "ERR" OK??
10560 045702 001412      BEQ     30$         ;YES!!
10561 045704 062716 000004      ADD     #4,(SP)       ;MOVE SP TO USER'S ERROR
10562 045710 112776 000047 000000    MOVB   #47,2(SP)     ;WRITE ERROR NUMBER
10563 045716 162716 000002      SUB     #2,(SP)       ;MOVE SP TO ERROR RETURN
10564 045722 004736      JSR     PC,2(SP)+     ;REPORT INVALID COMP ERROR
10565 045724 162716 000010      SUB     #10,(SP)
10566
10567      ;REPORT AN ERROR IF "TRE" IS SET AND NONE OF THE BITS WHICH SET
10568      ;TRE IS SET, OR IF TRE IS NOT SET AND ONE OR MORE BITS WHICH
10569      ;SET TRE IS SET
10570 045730 005037 001140 30$:  CLR     $GDDAT       ;EXPECT "TRE" =0
10571 045734 013746 001340      MOV     RMCS2I,-(SP) ;WAS DLT, WCE, UPE, NED, NEM
10572 045740 042726 000377      RIC     #377,(SP)+   ;PGE, MXF OR MOPE SET
10573 045744 001010      BNE     35$         ;YES!!
10574 045746 032737 040000 001342    BIT     #ERR,RMDSI   ;WAS EXCEPTION RECEIVED??
10575 045754 001407      BEQ     40$         ;NO!!
10576 045756 022737 000030 051224    CMP     #SEARCH,515$ ;WAS DATA TRANSFERRED??
10577 045764 103003      BHIS   40$         ;NO!!
10578 045766 052737 040000 001140 35$:  BIS     #TRE,$GDDAT  ;"TRE" SHOULD BE SET
10579 045774 013737 001330 001142 40$:  MOV     RMCS1I,$BDDAT ;BAD DATA FOR TYPEOUT
10580 046002 042737 137777 001142    BIC     #1CTRE,$BDDAT
10581 046010 023737 001140 001142    CMP     $GDDAT,$BDDAT ;IS "TRE" OK??
10582 046016 001413      BEQ     45$         ;YES!!
10583 046020 062716 000004      ADD     #4,(SP)       ;MOVE SP TO USER'S ERROR CALL
10584 046024 112776 000014 000000    MOVB   #14,2(SP)    ;WRITE ERROR NUMBER

```

B01

CZRMEBO RMO3/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 208
SECONDARY ERROR CHECK SUBROUTINE

SEQ 0208

10584 046032 162716 000002
10585 046036 004736
10586 046040 162716 000010
10587 046044 000240
10588 046046

SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
JSR PC,2(SP)+ ;REPORT THE ERROR
SUB #10,(SP) ;RESTORE (SP)
NOP

45\$:

; USING THE FUNCTION CODE TABLE, CHECK FOR THE FOLLOWING ERRORS:
; STATUS BITS NOT SET THAT SHOULD BE SET, E.G., ATA AND ILF
; STATUS BITS SET THAT SHOULD NOT BE SET, E.G., WCE AND ECH
; NOTE THAT SOME ERROR BITS ARE CONDITIONAL ON THE COMMAND AND OTHER
; STATUS CONDITIONS, E.G. WRITE LOCK ERROR SHOULD ONLY BE SET IF
; WRITE LOCK IS ON AND THE COMMAND IS A WRITE.

; GET AND STORE THE ENTRY FROM THE FUNCTION CODE TABLE

10599 046046 010046
10600 046050 013700 051224
10601 046054 016037 070676 051216
10602 046062 012600

MOV RO,-(SP) ;PUSH RO ON STACK
MOV 500\$,RO ;RO = FUNCTION CODE
MOV FNCDTB(RO),500\$;STORE ENTRY
MOV (SP)+,RO ;POP STACK INTO RO

; REPORT AN ERROR IF AN UNEXPECTED ATTENTION OCCURRED OR IF
; ATA IS NOT SET AND SHOULD BE SET.

10606 046064 013737 051216 001140
10607 046072 032737 040000 001342
10608 046100 001403
10609 046102 052737 100000 001140
10610 046110 042737 077777 001140
10611 046116 013737 001342 001142
10612 046124 042737 077777 001142
10613 046132 023737 001140 001142
10614 046140 001413
10615 046142 062716 000004
10616 046146 112776 000006 000000
10617 046154 162716 000002
10618 046160 004736
10619 046162 162716 000010
10620 046166 000240
10621 046170

MOV 500\$,SGDAT ;GET EXPECTED ATA STATUS
BIT #ERR,RMSI ;IS COMPOSITE ERROR SET ??
BEQ 50\$;NO !!
BIS #ATA,SGDAT ;EXPECT AN ATTENTION
BIC #1CAT,SGDAT ;STRIP DONT CARES
MOV RMSI,\$BDAT ;GET RECEIVED ATA
BIC #1CAT,\$BDAT ;STRIP DONT CARES
CMP \$GDAT,\$BDAT ;IS ATA OK ??
BEQ 55\$;YES !!
ADD #4,(SP) ;MOVE SP TO USERS ERROR CALL
MOVB #6,2(SP) ;LOAD ERROR # IN CALL
SUB #2,(SP) ;MOVE SP TO ERROR RETURN
JSR PC,2(SP)+ ;REPORT ERROR
SUB #10,(SP) ;RESTORE SP
NOP

55\$:

; REPORT ERROR IF ILF IS INCORRECT, I.E., IF ILF DOES NOT COMPARE
; WITH FUNCTION CODE TABLE

10625 046170 013737 051216 001140
10626 046176 042737 177776 001140
10627 046204 012737 001344 001142
10628 046212 042737 177776 001142
10629 046220 023737 001140 001142
10630 046226 001412
10631 046230 062716 000004
10632 046234 112776 000254 000000
10633 046242 162716 000002
10634 046246 004736
10635 046250 162716 000010
10636 046254 005037 001140

MOV 500\$,SGDAT ;GET EXPECTED ILF
BIC #1CILF,SGDAT ;CLEAR ALL OTHER BITS
MOV #MER11,\$BDAT ;GET RECEIVED ILF
BIC #1CILF,\$BDAT ;CLEAR ALL OTHER BITS
CMP \$GDAT,\$BDAT ;IS ILF OK ??
BEQ 60\$;YES !!
ADD #4,(SP) ;MOVE SP TO USERS ERROR CALL
MOVB #254,2(SP) ;WRITE ERROR NUMBER IN CALL
SUB #2,(SP) ;MOVE SP TO ERROR RETURN
JSR PC,2(SP)+ ;REPORT ERROR AND RETURN
SUB #10,(SP) ;MOVE SP TO NO ERROR
CLR \$GDAT ;CLEAR EXPECTED STATUS

60\$:

; REPORT AN ERROR IF WCE IS SET AND SHOULD NOT BE SET
MOV 500\$,-(SP) ;GET WCE STATUS ENABLE

10639 046260 013746 051216

10640	046264	052716	137777		BIS	#1CWCE,(SP)	:SET ALL OTHER BITS
10641	046270	013737	001340	001142	MOV	RMCS21,\$BDDAT	:RECEIVED STATUS
10642	046276	042637	001142		BIC	(SP)+,\$BDDAT	:CLEAR WCE IF ENABLED
10643	046302	001412			BEQ	90\$:BRANCH IF WCE OK
10644	046304	062716	000004		ADD	#4,(SP)	:MOVE SP TO USER'S ERROR CALL
10645	046310	112776	000026	000000	MOVB	#26,@(SP)	:WRITE ERROR NUMBER
10646	046316	162716	000002		SUB	#2,(SP)	:MOVE SP TO ERROR RETURN
10647	046322	004736			JSR	PC,@(SP)+	:REPORT ERROR
10648	046324	162716	000010		SUB	#10,(SP)	:RESTORE ERROR
10649	046330						
10650							
10651							
10652	046330	013746	051216				:REPORT ERROR IF OPI STATUS IS SET AND SHOULD NOT BE SET
10653	046334	052716	157777		MOV	500\$ -(SP)	:GET OPI STATUS ENABLE
10654	046340	013737	001344	001142	BIS	#1COPI,(SP)	:SET ALL OTHER BITS
10655	046346	042637	001142		MOV	RMER11,\$BDDAT	:GET RECEIVED STATUS
10656	046352	001412			BIC	(SP)+,\$BDDAT	:CLEAR OPI IF ENABLED
10657	046354	062716	000004		BEQ	100\$:BRANCH IF OPI OK
10658	046360	112776	000164	000000	ADD	#4,(SP)	:MOVE SP TO USER'S ERROR CALL
10659	046366	162716	000002		MOVB	#164,@(SP)	:WRITE ERROR NUMBER IN CALL
10660	046372	004736			SUB	#2,(SP)	:MOVE SP TO ERROR RETURN
10661	046374	162716	000010		JSR	PC,@(SP)+	:REPORT ERROR
10662	046400				SUB	#10,(SP)	:RESTORE SP
10663							
10664							
10665							:REPORT ERROR IF IVC IS SET AND IS NOT ENABLED OR IF IVC IS
10666	046400	013746	051216				:SET AND VV IS NOT RESET
10667	046404	032737	000100	001342	MOV	500\$ -(SP)	:GET IVC STATUS ENABLE
10668	046412	001402			BIT	#VV,RMDSI	:IS VV SET
10669	046414	042716	010000		BEQ	105\$:NO !!
10670	046420	052716	167777		BIC	#IVC,(SP)	:YES - IVC SHOULD BE 0
10671	046424	013737	001372	001142	BIS	#1CIVC,(SP)	:SET ALL OTHER BITS
10672	046432	042637	001142		MOV	RMER21,\$BDDAT	:GET RECEIVED STATUS
10673	046436	001412			BIC	(SP)+,\$BDDAT	:CLEAR IVC IF ENABLED
10674	046440	062716	000004		BEQ	110\$:BRANCH IF IVC OK
10675	046444	112776	000165	000000	ADD	#4,(SP)	:MOVE SP TO USERS ERROR CALL
10676	046452	162716	000002		MOVB	#165,@(SP)	:WRITE ERROR NUMBER IN CALL
10677	046456	004736			SUB	#2,(SP)	:MOVE SP TO ERROR RETURN
10678	046460	162716	000010		JSR	PC,@(SP)+	:REPORT ERROR
10679	046464				SUB	#10,(SP)	:RESTORE SP TO NO ERROR
10680							
10681							
10682							:BIT 11 (WLE) OF THE FUNCTION CODE TABLE IS THE ENABLING BIT FOR
10683							: ALL WRITE ERRORS, I.E.,
10684							: RMER1 - WLE, WCF
10685							: RMER2 - DPE
10686							: RMCS2 - UPE
10687							: EACH OF THESE ERRORS IS CHECKED TO SEE IF AN ERROR IS SET WHEN THE
10688							: WRITE ERROR ENABLE BIT IS RESET.
10689							
10690							:REPORT AN ERROR IF WLE IS SET AND WRITE ERRORS ARE NOT ENABLED, OR IF
10691	046464	012746	177777				:THE DRIVE IS NOT WRITE PROTECTED
10692	046470	032737	004000	051216	MOV	#-1 -(SP)	:ASSUME WRITE ERRORS ENABLED
10693	046476	001404			BIT	#WLE,500\$:ARE WRITE ERRORS ENABLED ??
10694	046500	032737	004000	001342	BEQ	115\$:NO !!
10695	046506	001002			BIT	#WRL,RMDSI	:IS THE DRIVE WRITE PROTECTED ??
					BNE	120\$:YES !!

```

10696 046510 042716 004000      115$: BIC      #WLE, (SP)      ; RESET WLE ENABLE
10697 046514 013737 001344      001142 120$: MOV      RMER1I, $BDDAT ; GET RECEIVED STATUS
10698 046522 042637 001142      BIC      (SP)+, $BDDAT ; CLEAR WLE IF ENABLED
10699 046526 001412      BEQ      125$ ; BRANCH IF WLE OK
10700 046530 062716 000004      ADD      #4, (SP) ; MOVE SP TO USERS ERROR CALL
10701 046534 112776 000023      000000  MOVB     #23, 2(SP) ; WRITE ERROR NUMBER IN CALL
10702 046542 162716 000002      SUB      #2, (SP) ; MOVE SP TO ERROR RETURN
10703 046546 004736      JSR      PC, 2(SP)+ ; REPORT ERROR AND RETURN
10704 046550 162716 000010      SUB      #10, (SP) ; RESTORE SP TO NO ERROR
10705 046554
10706
10707 ;REPORT ERROR IF WCF IS SET AND WRITE ERRORS ARE NOT ENABLED
10708 046554 012746 177777      MOV      #-1, -(SP) ; ASSUME WRITE ERRORS ENABLED
10709 046560 032737 004000      051216  BIT      #WLE, 500$ ; ARE WRITE ERRORS ENABLED ??
10710 046566 001002      BNE      130$ ; YES !!
10711 046570 042716 000040      BIC      #WCF, (SP) ; DISABLE WCF ERROR
10712 046574 013737 001344      001142 130$: MOV      RMER1I, $BDDAT ; GET RECEIVED STATUS
10713 046602 042637 001142      BIC      (SP)+, $BDDAT ; RESET WCF IF ENABLED
10714 046606 001412      BEQ      135$ ; BRANCH IF WCF OK
10715 046610 062716 000004      ADD      #4, (SP) ; MOVE SP TO USERS ERROR CALL
10716 046614 112776 000025      000000  MOVB     #25, 2(SP) ; WRITE ERROR NUMBER IN CALL
10717 046622 162716 000002      SUB      #2, (SP) ; MOVE SP TO ERROR RETURN
10718 046626 004736      JSR      PC, 2(SP)+ ; REPORT ERROR
10719 046630 162716 000010      SUB      #10, (SP) ; RESTORE SP TO NO ERROR
10720 046634
10721
10722 ;REPORT ERROR IF DPE IS SET AND WRITE ERRORS ARE NOT ENABLED
10723 046634 012746 177777      MOV      #-1, -(SP) ; ASSUME WRITE ERRORS ARE ENABLED
10724 046640 032737 004000      051216  BIT      #WLE, 500$ ; ARE WRITE ERRORS ENABLED ??
10725 046646 001002      BNE      140$ ; YES !!
10726 046650 042716 000010      BIC      #DPE, (SP) ; RESET DPE ENABLE
10727 046654 013737 001372      001142 140$: MOV      RMER2I, $BDDAT ; GET RECEIVED STATUS
10728 046662 042637 001142      BIC      (SP)+, $BDDAT ; RESET DPE IF ENABLED
10729 046666 001412      BEQ      145$ ; BRANCH IF DPE OK
10730 046670 062716 000004      ADD      #4, (SP) ; MOVE SP TO USERS ERROR CALL
10731 046674 112776 000040      000000  MOVB     #40, 2(SP) ; WRITE ERROR NUMBER IN CALL
10732 046702 162716 000002      SUB      #2, (SP) ; MOVE SP TO ERROR RETURN
10733 046706 004736      JSR      PC, 2(SP)+ ; REPORT ERROR
10734 046710 162716 000010      SUB      #10, (SP) ; RESTORE SP TO NO ERROR
10735 046714
10736
10737 ;REPORT AN ERROR IF UPE IS SET AND WRITE ERRORS ARE NOT ENABLED
10738 046714 012746 177777      MOV      #-1, -(SP) ; ASSUME WRITE ERRORS ARE ENABLED
10739 046720 032737 004000      051216  BIT      #WLE, 500$ ; ARE WRITE ERRORS ENABLED ??
10740 046726 001002      BNE      150$ ; YES !!
10741 046730 042716 020000      BIC      #UPE, (SP) ; DISABLE UPE ERROR
10742 046734 013737 001340      001142 150$: MOV      RMCS2I, $BDDAT ; GET RECEIVED STATUS
10743 046742 042637 001142      BIC      (SP)+, $BDDAT ; RESET UPE IF ENABLED
10744 046746 001412      BEQ      155$ ; BRANCH IF UPE OK
10745 046750 062716 000004      ADD      #4, (SP) ; MOVE SP TO USERS ERROR CALL
10746 046754 112776 000024      000000  MOVB     #24, 2(SP) ; WRITE ERROR NUMBER IN CALL
10747 046762 162716 000002      SUB      #2, (SP) ; MOVE SP TO ERROR RETURN
10748 046766 004736      JSR      PC, 2(SP)+ ; REPORT ERROR AND RETURN
10749 046770 162716 000010      SUB      #10, (SP) ; RESTORE SP TO NO ERROR
10750 046774
10751
155$:

```

EO1

CZRMEBO RM03/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 211
SECONDARY ERROR CHECK SUBROUTINE

SEQ 0211

```

10752 ;REPORT AN ERROR IF IAE IS SET AND IS NOT ENABLED
10753 046774 013746 051216      MOV      500$ -(SP)      ;GET IAE ENABLE
10754 047000 052716 175777      BIS      #+CIAE,(SP)    ;SET ALL OTHER BITS
10755 047004 013737 001344 001142  MOV      RMER1I,$BDDAT  ;GET RECEIVED STATUS
10756 047012 042637 001142      BIC      (SP)+,$BDDAT  ;CLEAR IAE IF ENABLED
10757 047016 001412              BEQ      160$          ;BRANCH IF IAE IS OK
10758 047020 062716 000004      ADD      #4,(SP)       ;MOVE SP TO USERS ERROR CALL
10759 047024 112776 000166 000000  MOVB     #166,@(SP)    ;WRITE ERROR NUMBER
10760 047032 162716 000002      SUB      #2,(SP)       ;MOVE SP TO ERROR RETURN
10761 047036 004736              JSR      PC,@(SP)+     ;REPORT ERROR AND RETURN
10762 047040 162716 000010      SUB      #10,(SP)     ;MOVE SP TO NO ERROR
10763 047044
10764
10765 ;BIT 09 (AOE) OF THE FUNCTION CODE TABLE IS THE ENABLING BIT FOR
10766 ; ALL READ/WRITE ERRORS, I.E.,
10767 ;
10768 ; RMCS1 - TRE
10769 ; RMCS2 - DLT,NEM,MXF
10770 ; RMD5 - LBT
10771 ; RMER1 - AOE
10772 ;NOTE:
10773 ; LBT IS NOT CHECKED BECAUSE IT ONLY RESETS WHEN THE DESIRED
10774 ; CYLINDER REGISTER IS WRITTEN
10775 ;NOTE:
10776 ; AOE CANNOT BE SET IF LBT IS NOT ALSO SET
10777 ;NOTE:
10778 ; TRE IS CHECKED AS A FUNCTION OF OTHER ERROR CONDITONS ABOVE
10779
10780 ;REPORT AN ERROR IF DLT IS SET AND READ/WRITE ERRORS ARE NOT ENABLED
10781 047044 012746 177777      MOV      #-1, -(SP)    ;ASSUME ERRORS ARE ENABLED
10782 047050 032737 001000 051216  BIT      #AOE,500$    ;ARE ERRORS ENABLED ??
10783 047056 001002              BNE      165$          ;YES !!
10784 047060 042716 100000      BIC      #DLT,(SP)    ;RESET DLT ENABLE
10785 047064 013737 001340 001142 165$: MOV      RMCS2I,$BDDAT ;GET RECEIVED STATUS
10786 047072 042637 001142      BIC      (SP)+,$BDDAT ;CLEAR DLT IF ENABLED
10787 047076 001412              BEQ      170$          ;BRANCH IF DLT IS OK
10788 047100 062716 000004      ADD      #4,(SP)       ;MOVE SP TO USERS ERROR CALL
10789 047104 112776 000032 000000  MOVB     #32,@(SP)    ;WRITE ERROR NUMBER IN CALL
10790 047112 162716 000002      SUB      #2,(SP)       ;MOVE SP TO ERROR RETURN
10791 047116 004736              JSR      PC,@(SP)+     ;REPORT ERROR AND RETURN
10792 047120 162716 000010      SUB      #10,(SP)     ;MOVE SP TO NO ERROR
10793 047124
10794
10795 ;REPORT ERROR IF NEM IS SET AND READ/WRITE ERRORS ARE NOT ENABLED
10796 047124 012746 177777      MOV      #-1, -(SP)    ;ASSUME ERRORS ARE ENABLED
10797 047130 032737 001000 051216  BIT      #AOE,500$    ;ARE ERRORS ENABLED ??
10798 047136 001002              BNE      175$          ;YES !!
10799 047140 042716 004000      BIC      #NEM,(SP)    ;DISABLE NEM
10800 047144 013737 001340 001142 175$: MOV      RMCS2I,$BDDAT ;GET RECEIVED STATUS
10801 047152 042637 001142      BIC      (SP)+,$BDDAT ;CLEAR NEM IF ENABLED
10802 047156 001412              BEQ      180$          ;BRANCH IF NEM IS OK
10803 047160 062716 000004      ADD      #4,(SP)       ;MOVE SP TO USERS ERROR CALL
10804 047164 112776 000167 000000  MOVB     #167,@(SP)    ;WRITE ERROR NUMBER IN CALL
10805 047172 162716 000002      SUB      #2,(SP)       ;MOVE SP TO ERROR RETURN
10806 047176 004736              JSR      PC,@(SP)+     ;REPORT ERROR AND RETURN
10807 047200 162716 000010      SUB      #10,(SP)     ;MOVE SP TO NO ERROR

```

```

10808 047204      180$:
10809
10810      ;REPORT ERROR IF MXF IS SET AND READ/WRITE ERRORS ARE NOT ENABLED
10811 047204 012746 177777      MOV      #-1, -(SP)      ;ASSUME ERRORS ARE ENABLED
10812 047210 032737 001000 051216      BIT      #AOE, 500$      ;ARE DATA ERRORS ENABLED ??
10813 047216 001002      BNE      185$           ;YES !!
10814 047220 042716 001000      BIC      #MXF, (SP)      ;DISABLE MXF ERROR
10815 047224 013737 001340 001142 185$:      MOV      RMCS2I, $BDDAT      ;GET RECEIVED STATUS
10816 047232 042637 001142      BIC      (SP)+, $BDDAT      ;CLEAR MXF IF ENABLED
10817 047236 001412      BEQ      190$           ;BRANCH IF MXF IS OK
10818 047240 062716 000004      ADD      #4, (SP)         ;MOVE SP TO USERS ERROR CALL
10819 047244 112776 000033 000000      MOVVB   #33, 2(SP)        ;WRITE ERROR NUMBER IN CALL
10820 047252 162716 000002      SUB      #2, (SP)         ;MOVE SP TO ERROR RETURN
10821 047256 004736      JSR      PC, 2(SP)+       ;REPORT ERROR AND RETURN
10822 047260 162716 000010      SUB      #10, (SP)        ;MOVE SP TO NO ERROR
10823 047264
10824
10825      ;REPORT ERROR IF AOE IS SET AND DATA ERRORS ARE NOT ENABLED
10826 047264 012746 177777      MOV      #-1, -(SP)      ;ASSUME DATA ERRORS ARE ENABLED
10827 047270 032737 001000 051216      BIT      #AOE, 500$      ;ARE DATA ERRORS ENABLED ??
10828 047276 001404      BEQ      191$           ;NO !!
10829 047300 032737 002000 001342      BIT      #LBT, RMDSI      ;IS LBT ALSO SET ??
10830 047306 001002      BNE      195$           ;YES !!
10831 047310 042716 001000 191$:      BIC      #AOE, (SP)       ;DISABLE AOE
10832 047314 013737 001344 001142 195$:      MOV      RMER1I, $BDDAT      ;GET RECEIVED STATUS
10833 047322 042637 001142      BIC      (SP)+, $BDDAT      ;CLEAR AOE IF ENABLED
10834 047326 001412      BEQ      200$           ;BRANCH IF AOE IS OK
10835 047330 062716 000004      ADD      #4, (SP)         ;MOVE SP TO USERS ERROR CALL
10836 047334 112776 000020 000000      MOVVB   #20, 2(SP)        ;WRITE ERROR NUMBER
10837 047342 162716 000002      SUB      #2, (SP)         ;MOVE SP TO ERROR RETURN
10838 047346 004736      JSR      PC, 2(SP)+       ;REPORT ERROR AND RETURN
10839 047350 162716 000010      SUB      #10, (SP)        ;MOVE SP TO NO ERROR
10840 047354
10841
10842      ;BIT 07 (HCE) OF THE FUNCTION CODE TABLE IS THE ENABLING BIT FOR
10843 ;HEADER ERRORS, I.E.,
10844 ;      RMER1 - HCRC, HCE, FER
10845 ;      RMER2 - BSE
10846
10847      ;RESET THE ENABLING BIT (HCE) IF HEADER COMPARE INHIBIT IS SET
10848 047354 032737 002000 001362      BIT      #HCI, RMOFI      ;IS HCI SET ??
10849 047362 001403      BEQ      201$           ;NO !!
10850 047364 042737 000200 051216      BIC      #HCE, 500$       ;YES - DISABLE ALL HEADER ERRORS
10851 047372
10852 201$:
10853      ;REPORT AN ERROR IF HCRC IS SET AND HEADER ERRORS ARE NOT ENABLED
10854 047372 012746 177777      MOV      #-1, -(SP)      ;ASSUME ERRORS ENABLED
10855 047376 032737 000200 051216      BIT      #HCE, 500$      ;ARE HEADER ERRORS ENABLED ??
10856 047404 001002      BNE      205$           ;YES !!
10857 047406 042716 000400      BIC      #HCRC, (SP)      ;DISABLE HCRC
10858 047412 013737 001344 001142 205$:      MOV      RMER1I, $BDDAT      ;GET RECEIVED STATUS
10859 047420 042637 001142      BIC      (SP)+, $BDDAT      ;RESET HCRC IF ENABLED
10860 047424 001412      BEQ      210$           ;BRANCH IF HCRC IS OK
10861 047426 062716 000004      ADD      #4, (SP)         ;MOVE SP TO USERS ERROR CALL
10862 047432 112776 000035 000000      MOVVB   #35, 2(SP)        ;WRITE ERROR NUMBER IN CALL
10863 047440 162716 000002      SUB      #2, (SP)         ;MOVE SP TO ERROR RETURN

```



```

10864 047444 004736          JSR    PC,2(SP)+      ;REPORT ERROR AND RETURN
10865 047446 162716 000010  SUB    #10,(SP)      ;MOVE SP TO NO ERROR
10866 047452          210$:
10867
10868          ;REPORT ERROR IF HCE IS SET AND HEADER ERRORS ARE NOT ENABLED
10869 047452 012746 177777      MOV    #-1,-(SP)    ;ASSUME ERRORS ENABLED
10870 047456 032737 000200 051216  BIT    #HCE,500$   ;ARE ERRORS ENABLED ??
10871 047464 001002          BNE    215$        ;YES !!
10872 047466 042716 000200      BIC    #HCE,(SP)   ;DISABLE HCE
10873 047472 013737 001344 001142 215$:  MOV    RMER1I,$BDDAT ;GET RECEIVED STATUS
10874 047500 042637 001142      BIC    (SP)+,$BDDAT ;CLEAR HCE IF ENABLED
10875 047504 001412          BEQ    220$        ;BRANCH IF HCE IS OK
10876 047506 062716 000004      ADD    #4,(SP)    ;MOVE SP TO USERS ERROR CALL
10877 047512 112776 000036 000000  MOVB   #36,2(SP)   ;WRITE ERROR NUMBER IN CALL
10878 047520 162716 000002      SUB    #2,(SP)    ;MOVE SP TO ERROR RETURN
10879 047524 004736          JSR    PC,2(SP)+   ;REPORT ERROR AND RETURN
10880 047526 162716 000010  SUB    #10,(SP)   ;MOVE SP TO NO ERROR
10881 047532          220$:
10882
10883          ;REPORT ERROR IF FER IS SET AND HEADER ERRORS ARE NOT ENABLED
10884 047532 012746 177777      MOV    #-1,-(SP)  ;ASSUME FER IS ENABLED
10885 047536 032737 000200 051216  BIT    #HCE,500$   ;ARE HEADER ERRORS ENABLED ??
10886 047544 001002          BNE    225$        ;YES !!
10887 047546 042716 000020      BIC    #FER,(SP)  ;DISABLE FER
10888 047552 013737 001344 001142 225$:  MOV    RMER1I,$BDDAT ;GET RECEIVED STATUS
10889 047560 042637 001142      BIC    (SP)+,$BDDAT ;RESET FER IF ENABLED
10890 047564 001412          BEQ    230$        ;BRANCH IF FER OK
10891 047566 062716 000004      ADD    #4,(SP)   ;MOVE SP TO USERS ERROR CALL
10892 047572 112776 000037 000000  MOVB   #37,2(SP)   ;WRITE ERROR NUMBER IN CALL
10893 047600 162716 000002      SUB    #2,(SP)   ;MOVE SP TO ERROR RETURN
10894 047604 004736          JSR    PC,2(SP)+   ;REPORT ERROR AND RETURN
10895 047606 162716 000010  SUB    #10,(SP)   ;MOVE SP TO NO ERROR
10896 047612          230$:
10897
10898          ;REPORT ERROR IF BSE IS SET AND HEADER ERRORS ARE NOT ENABLED
10899 047612 012746 177777      MOV    #-1,-(SP)  ;ASSUME ERRORS ENABLED
10900 047616 032737 000200 051216  BIT    #HCE,500$   ;ARE THEY ENABLED ??
10901 047624 001002          BNE    235$        ;YES !!
10902 047626 042716 100000      BIC    #BSE,(SP)  ;DISABLE BSE
10903 047632 013737 001372 001142 235$:  MOV    RMER2I,$BDDAT ;GET RECEIVED STATUS
10904 047640 042637 001142      BIC    (SP)+,$BDDAT ;CLEAR BSE IF ENABLED
10905 047644 001412          BEQ    240$        ;BRANCH IF BSE OK
10906 047646 062716 000004      ADD    #4,(SP)   ;MOVE SP TO USERS ERROR CALL
10907 047652 112776 000354 000000  MOVB   #35,2(SP)   ;WRITE ERROR NUMBER
10908 047660 162716 000002      SUB    #2,(SP)   ;MOVE SP TO ERROR RETURN
10909 047664 004736          JSR    PC,2(SP)+   ;REPORT ERROR AND RETURN
10910 047666 162716 000010  SUB    #10,(SP)   ;MOVE SP TO NO ERROR
10911 047672          240$:
10912
10913          ;BIT 06 OF THE FUNCTION CODE TABLE IS THE ENABLING BIT FOR DATA
10914          ;FIELD ERRORS, I.E.
10915          ; RMCS2 - MDPÉ
10916          ; RMER1 - DCK,ECH
10917          ;NOTE:
10918          ; ECH CANNOT SET UNLESS IT IS ENABLED AND ECI IS RESET AND
10919          ; DCK IS SET.

```

H01

CZRMEBO RMO3.2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 214
SECONDARY ERROR CHECK SUBROUTINE

SEQ 0214

```

10920
10921 ;REPORT ERROR IF MDPE IS SET AND IS NOT ENABLED
10922 047672 012746 177777 MOV #1, -(SP) ;ASSUME ENABLED
10923 047676 032737 000100 051216 BIT #ECH, 500$ ;ARE DATA FIELD ERRORS ENABLED ??
10924 047704 001002 BNE 245$ ;YES !!
10925 047706 042716 000400 BIC #MDPE, (SP) ;DISABLE MDPE
10926 047712 013737 001340 001142 245$: MOV RMCS21, $BDDAT ;GET RECEIVED STATUS
10927 047720 042637 001142 BIC (SP)+, $BDDAT ;CLEAR MDPE IF ENABLED
10928 047724 001412 BEQ 250$ ;BRANCH IF MDPE OK
10929 047726 062716 000004 ADD #4, (SP) ;MOVE SP TO USERS ERROR CALL
10930 047732 112776 000027 000000 MOV# #27, 2(SP) ;WRITE ERROR NUMBER IN CALL
10931 047740 162716 000002 SUB #2, (SP) ;MOVE SP TO ERROR RETURN
10932 047744 004736 JSR PC, 2(SP)+ ;REPORT ERROR AND RETURN
10933 047746 162716 000010 SUB #10, (SP) ;MOVE SP TO NO ERROR
10934 047752 250$:
10935
10936 ;REPORT ERROR IF DCK IS SET AND DATA FIELD ERRORS ARE NOT ENABLED
10937 047752 012746 177777 MOV #1, -(SP) ;ASSUME ENABLED
10938 047756 032737 000100 051216 BIT #ECH, 500$ ;ARE THEY ENABLED ??
10939 047764 001002 BNE 255$ ;YES !!
10940 047766 042716 100000 BIC #DCK, (SP) ;DISABLE DCK
10941 047772 013737 001344 001142 255$: MOV RMER11, $BDDAT ;GET RECEIVED STATUS
10942 050000 042637 001142 BIC (SP)+, $BDDAT ;CLEAR DCK IF ENABLED
10943 050004 001412 BEQ 260$ ;BRANCH IF DCK IS OK
10944 050006 062716 000004 ADD #4, (SP) ;MOVE SP TO USERS ERROR CALL
10945 050012 112776 000030 000000 MOV# #30, 2(SP) ;WRITE ERROR NUMBER IN CALL
10946 050020 162716 000002 SUB #2, (SP) ;MOVE SP TO ERROR RETURN
10947 050024 004736 JSR PC, 2(SP)+ ;REPORT ERROR AND RETURN
10948 050026 162716 000010 SUB #10, (SP) ;MOVE SP TO NO ERROR
10949 050032 260$:
10950
10951 ;REPORT ERROR IF ECH IS SET AND
10952 ;DATA FIELD ERRORS ARE NOT ENABLED, OR
10953 ;ECI IS SET, OR
10954 ;DCK IS NOT SET
10955 050032 012746 177777 MOV #1, -(SP) ;ASSUME ENABLED
10956 050036 032737 000100 051216 BIT #ECH, 500$ ;ARE ERRORS ENABLED ??
10957 050044 001410 BNE 265$ ;NO !!
10958 050046 032737 004000 001362 BIT #ECI, RMOFI ;IS ECI SET ??
10959 050054 001004 BNE 265$ ;YES !!
10960 050056 032737 100000 001344 BIT #DCK, RMER11 ;IS DCK ALSO SET ??
10961 050064 001002 BNE 270$ ;YES !!
10962 050066 042716 000100 265$: BIC #ECH, (SP) ;DISABLE ECH
10963 050072 013737 001344 001142 270$: MOV RMER11, $BDDAT ;GET RECEIVED STATUS
10964 050100 042637 001142 BIC (SP)+, $BDDAT ;CLEAR ECH IF ENABLED
10965 050104 001412 BEQ 275$ ;BRANCH IF ECH IS OK
10966 050106 062716 000004 ADD #4, (SP) ;MOVE SP TO USERS ERROR CALL
10967 050112 112776 000031 000000 MOV# #31, 2(SP) ;WRITE ERROR NUMBER IN CALL
10968 050120 162716 000002 SUB #2, (SP) ;MOVE SP TO ERROR RETURN
10969 050124 004736 JSR PC, 2(SP)+ ;REPORT ERROR AND RETURN
10970 050126 162716 000010 SUB #10, (SP) ;MOVE SP TO NO ERROR
10971 050132 275$:
10972
10973 ;*****
10974 ;PERFORM THE REMAINING ERROR CHECKS ONLY FOR DATA TRANSFER COMMANDS
10975

```

I01

CZRMEBO RMO3/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 215
SECONDARY ERROR CHECK SUBROUTINE

SEQ 0215

10976	050132	022737	000030	051224	CMP	#SEARCH,515\$;WAS DATA TRANSFERRED??
10977	050140	103402			BLO	280\$;YES!!
10978	050142	000137	051170		JMP	355\$	
10979							

```

10980 ;THE FOLLOWING STATUS CHECKS ARE FOR DATA TRANSFER COMMANDS ONLY
10981 ;REPORT ERROR IF RMWC NOT ZERO AND TRE IS ZERO
10982 050146 013737 001332 001142 280$: MOV RMWCI,$BDDAT ;WORD COUNT ZERO??
10983 050154 001421 ;BEQ 285$ ;YES
10984 050156 032737 040000 001330 BIT #TRE,RMCSI1 ;TRANSFER ERROR DETECTED??
10985 050164 001015 ;BNE 285$ ;YES!!
10986 050166 062716 000004 ADD #4,(SP)
10987 050172 112776 000015 000000 MOVB #15,2(SP) ;ERROR NUMBER
10988 050200 005037 001140 CLR $GDDAT ;GOOD DATA FOR TYPEOUT
10989 050204 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
10990 050210 004736 JSR PC,2(SP)+ ;REPORT WORD COUNT NOT ZERO
10991 050212 162716 000010 SUB #10,(SP) ;RESTORE (SP)
10992 050216 000240 NOP
10993 ;REPORT ERROR IF RMBA IS NOT CORRECT
10994 050220 013737 001332 001140 285$: MOV RMWCI,$GDDAT ;NUMBER OF WORDS TRANSFERRED
10995 050226 163737 001402 001140 SUB RMWCO,$GDDAT
10996 050234 006337 001140 ASL $GDDAT
10997 050240 063737 001404 001140 ADD RMBA0,$GDDAT ;EXPECTED BUS ADDRESS
10998 050246 032737 000010 001340 BIT #BAI,RMCSI2 ;WAS BAI SET ??
10999 050254 001403 ;BEQ 290$ ;NO !!
11000 050256 013737 001404 001140 MOV RMBA0,$GDDAT ;ADDRESS SHOULD NOT HAVE CHANGED
11001 050264 023737 001140 001334 290$: CMP $GDDAT,RMBAI ;BUS ADDRESS OK??
11002 050272 001416 ;BEQ 295$ ;YES!!
11003 050274 013737 001334 001142 MOV RMBAI,$BDDAT ;BAD DATA FOR TYPEOUT
11004 050302 062716 000004 ADD #4,(SP)
11005 050306 112776 000016 000000 MOVB #16,2(SP) ;ERROR NUMBER
11006 050314 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
11007 050320 004736 JSR PC,2(SP)+ ;REPORT UNEXPECTED ADDRESS
11008 050322 162716 000010 SUB #10,(SP) ;RESTORE (SP)
11009 050326 000240 NOP
11010 ;COMPUTE NUMBER OF SECTORS TRANSFERRED
11011 050330 005046 295$: CLR -(SP) ;NUMBER OF SECTORS TRANSFERRED
11012 050332 013746 001332 MOV RMWCI,-(SP) ;NUMBER OF WORDS TRANSFERRED
11013 050336 163716 001402 SUB RMWCO,(SP)
11014 050342 012746 000400 MOV #256,-(SP) ;NUMBER OF WORDS PER SECTOR
11015 050346 032737 000002 001400 BIT #BIT1,RMCSI0 ;HEADER & DATA COMMAND ??
11016 050354 001402 ;BEQ 300$ ;NO !!
11017 050356 012716 000402 MOV #256,(SP) ;YES - CHANGE WORDS PER SECTOR
11018 050362 005266 000004 300$: INC 4(SP) ;INCREMENT SECTOR COUNT
11019 050366 161666 000002 SUB (SP),2(SP) ;SUBTRACT ONE SECTOR'S WORTH
11020 050372 003373 BGT 300$ ;CONTINUE IF NOT DONE
11021 050374 022626 CMP (SP)+,(SP)+ ;STRIP 2 FROM STACK
11022 ;COMPUTE EXPECTED SECTOR, TRACK AND CYLINDER ADDRESS FROM NUMBER OF SECTORS
11023 050376 013737 001406 051222 MOV RMDA0,510$ ;STORE ORIGINAL SECTOR
11024 050404 013737 001406 051220 MOV RMDA0,505$ ;STORE ORIGINAL TRACK
11025 050412 013737 001434 051216 MOV RMDCO,500$ ;STORE ORIGINAL CYLINDER
11026 050420 042737 177740 051222 BIC #C37,510$
11027 050426 000337 051220 SWAB 505$
11028 050432 042737 177770 051220 BIC #C7,505$
11029 050440 062637 051222 ADD (SP)+,510$
11030
11031 050444 023727 051222 000040 305$: CMP 510$,#32. ;SECTOR OVEFLOWED??
11032 050452 103420 ;BLO 315$ ;NO!!
11033 050454 023727 051222 000240 CMP 510$,#<5*32.> ;CYLINDERS WORTH??
11034 050462 103406 ;BLO 310$ ;NO!!
11035 050464 005237 051216 INC 500$ ;YES INCREMENT CYLINDER

```

```

11036 050470 162737 000240 051222      SUB      #(<5*32.),510$ ;ADJUST SECTOR
11037 050476 000762                BR      305$ ;TRY AGAIN
11038 050500 005237 051220 310$:    INC      505$ ;INCREMENT TRACK
11039 050504 162737 000040 051222      SUB      #32.,510$ ;ADJUST SECTOR
11040 050512 000754                BR      305$ ;TRY AGAIN
11041
11042 050514 023727 051220 000005 315$:    CMP      505$,#5 ;TRACK OVERFLOWED??
11043 050522 103406                BLO     320$ ;NO!!
11044 050524 005237 051216      INC      500$ ;INCREMENT CYLINDER
11045 050530 162737 000005 051220      SUB      #5,505$ ;ADJUST TRACK
11046 050536 000766                BR      315$ ;TRY AGAIN
11047      ;REPORT ERROR IF "LBT" IS NOT CORRECT
11048 050540 005037 001140 320$:    CLR      $GDDAT ;SET GOOD DATA FOR LBT=0
11049 050544 022737 001467 051216      CMP      #823.,500$ ;SHOULD LBT BE SET??
11050 050552 101007                BHI     325$ ;NO!!
11051 050554 032737 002000 001344      BIT      #IAE,RMER1I ;WAS IAE SET ??
11052 050562 001003                BNE     325$ ;YES - LBT SHOULD NOT BE SET
11053 050564 012737 002000 001140      MOV      #LBT,$GDDAT ;SET GOOD DATA FOR LBT=1
11054 050572 013737 001342 001142 325$:    MOV      #RMDA,$BDDAT ;BAD DATA FOR TYPEOUT
11055 050600 042737 175777 001142      BIC      #CLBT,$BDDAT
11056 050606 023737 001140 001142      CMP      $GDDAT,$BDDAT ;IS LBT CORRECT??
11057 050614 001413                BEQ     330$ ;YES!!
11058 050616 062716 000004      ADD      #4,(SP)
11059 050622 112776 000017 000000      MOVVB   #17,2(SP) ;ERROR NUMBER
11060 050630 162716 000002      SUB      #2,(SP) ;MOVE SP TO RETURN FOR ERROR
11061 050634 004736                JSR     PC,2(SP)+ ;REPORT LBT IS WRONG
11062 050636 162716 000010      SUB      #10,(SP) ;RESTORE (SP)
11063 050642 000240      NOP
11064      ;REPORT ERROR IF "AOE" IS INCORRECT
11065 050644 005037 001140 330$:    CLR      $GDDAT ;SET FOR AOE=0
11066 050650 032737 002000 001344      BIT      #IAE,RMER1I ;WAS "IAE" DETECTED??
11067 050656 001031                BNE     340$ ;YES-"AOE" SHOULD BE ZERO
11068 050660 022737 001467 051216      CMP      #823.,500$ ;SHOULD AOE BE SET??
11069 050666 101025                BHI     340$ ;NO!!
11070 050670 005737 051220      TST     505$ ;MAYBE
11071 050674 001012                BNE     335$ ;YES
11072 050676 005737 051222      TST     510$
11073 050702 001007                BNE     335$ ;YES !!
11074 050704 032737 000010 051224      BIT      #F2,515$ ;WAS THIS READ OR WRITE CHECK ??
11075 050712 001413                BEQ     340$ ;NO !!
11076 050714 005737 001332      TST     #RMWCI ;WAS ALL DATA TRANSFERRED ??
11077 050720 001410                BEQ     340$ ;YES !!
11078 050722 012737 001000 001140 335$:    MOV      #AOE,$GDDAT ;SET FOR AOE=1
11079 050730 005037 051220      CLR      505$ ;CLEAR EXPECTED TRACK
11080 050734 012737 000001 051222      MOV      #1,510$ ;EXPECT SECTOR=1
11081 050742 013737 001344 001142 340$:    MOV      #RMER1I,$BDDAT ;BAD DATA FOR TYPEOUT
11082 050750 042737 176777 001142      BIC      #CAOE,$BDDAT
11083 050756 023737 001140 001142      CMP      $GDDAT,$BDDAT ;IS AOE CORRECT??
11084 050764 001413                BEQ     345$ ;YES!!
11085 050766 062716 000004      ADD      #4,(SP)
11086 050772 112776 000020 000000      MOVVB   #20,2(SP) ;ERROR NUMBER
11087 051000 162716 000002      SUB      #2,(SP) ;MOVE SP TO RETURN FOR ERROR
11088 051004 004736                JSR     PC,2(SP)+ ;REPORT AOE IS WRONG
11089 051006 162716 000010      SUB      #10,(SP) ;RESTORE (SP)
11090 051012 000240      NOP
11091      ;REPORT ERROR IF RMDA IS NOT CORRECT

```

L01

CZRMEBO RM03/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 218
SECONDARY ERROR CHECK SUBROUTINE

SEQ 0218

11092	051014	032737	002000	001344	345\$:	BIT	#IAE,RMER11	; WAS THERE AN IAE ERROR ??
11093	051022	001062				BNE	355\$; YES - DONT CHECK RMDA,RMDC
11094	051024	013737	051220	001140		MOV	505\$,\$GDDAT	; SETUP EXPECTED DISK ADDRESS
11095	051032	000337	001140			SWAB	\$GDDAT	
11096	051036	113737	051222	001140		MOVB	510\$,\$GDDAT	
11097	051044	013737	001336	001142		MOV	RMDAT,\$BDDAT	; SETUP RECEIVED DISK ADDRESS
11098	051052	023737	001140	001142		CMP	\$GDDAT,\$BDDAT	; COMPARE EXPECTED & RECEIVED
11099	051060	001413				BEQ	350\$; BRANCH IF EQUAL
11100	051062	062716	000004			ADD	#4,(SP)	
11101	051066	112776	000021	000000		MOVB	#21,@(SP)	; ERROR NUMBER
11102	051074	162716	000002			SUB	#2,(SP)	; MOVE SP TO RETURN FOR ERROR
11103	051100	004736				JSR	PC,@(SP)+	; REPORT BAD DISK ADDRESS
11104	051102	162716	000010			SUB	#10,(SP)	; RESTORE (SP)
11105	051106	000240				NOP		
11106								; REPORT ERROR IF RMDC IS INCORRECT
11107	051110	013737	051216	001140	350\$:	MOV	500\$,\$GDDAT	; SETUP EXPECTED CYLINDER
11108	051116	042737	176000	001140		BIC	#1C177,\$GDDAT	
11109	051124	013737	001364	001142		MOV	RMDCI,\$BDDAT	; SETUP RECEIVED CYLINDER
11110	051132	023737	001140	001142		CMP	\$GDDAT,\$BDDAT	; COMPARE CYLINDERS
11111	051140	001413				BEQ	355\$; BRANCH IF EQUAL
11112	051142	062716	000004			ADD	#4,(SP)	
11113	051146	112776	000022	000000		MOVB	#22,@(SP)	; ERROR NUMBER
11114	051154	162716	000002			SUB	#2,(SP)	; MOVE SP TO RETURN FOR ERROR
11115	051160	004736				JSR	PC,@(SP)+	; REPORT BAD CYLINDER
11116	051162	162716	000010			SUB	#10,(SP)	; RESTORE (SP)
11117	051166	000240				NOP		

MO1

CZRMEBO RM03/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 219
SECONDARY ERROR CHECK SUBROUTINE

SEQ 0219

11118	051170	062716	000004	355\$:	ADD	#4,(SP)	;MOVE (SP) TO ERROR CALL
11119	051174	105776	000000		TSTB	2(SP)	;WAS ERROR FOUND??
11120	051200	001403			BEQ	360\$	
11121	051202	062716	000004		ADD	#4,(SP)	;MOVE (SP) TO ERROR RETURN
11122	051206	000402			BR	365\$	
11123	051210	162716	000004	360\$:	SUB	#4,(SP)	;MOVE (SP) TO NO ERROR RETURN
11124	051214	000207		365\$:	RTS	PC	
11125							
11126	051216	000000		500\$:	.WORD	0	;CYLINDER
11127	051220	000000		505\$:	.WORD	0	;TRACK
11128	051222	000000		510\$:	.WORD	0	;SECTOR
11129	051224	000000		515\$:	.WORD	0	;FUNCTION CODE
11130							
11131							

```

11132
11133
11134
11135
11136
11137
11138
11139
11140
11141
11142
11143 051226
11144
11145
11146 051226 062716 000004
11147 051232 105076 000000
11148 051236 162716 000004
11149
11150 051242 013746 000004
11151 051246 013746 000006
11152 051252 010046
11153 051254 010146
11154 051256 012737 051376 000004
11155 051264 012737 000300 000006
11156 051272 013700 001276
11157 051276 013701 001450
11158
11159
11160 051302 111160 000010
11161 051306 016037 000000 001176
11162 051314 016037 000010 001174
11163 051322 032737 010000 001174
11164 051330 001407
11165 051332 062766 000004 000010
11166 051340 112776 000111 000010
11167 051346 000422
11168 051350 032737 004000 001176 10$:
11169 051356 001021
11170 051360 062766 000004 000010
11171 051366 112776 000112 000010
11172 051374 000407
11173
11174 051376
11175
11176
11177 051376 022626
11178 051400 062766 000004 000010
11179 051406 112776 000113 000010
11180 051414 162766 000002 000010 30$:
11181
11182 051422
11183
11184
11185
11186 051422 012601
11187 051424 012600

```

```

.SBTTL DEVICE SELECT SUBROUTINE
; THIS SUBROUTINE SELECTS THE DEVICE, GETTING THE DEVICE NUMFPR FROM THE
; TEST QUEUE.
;CALL:
;(1) JSR PC,DEVSEL
;(2) BR ?? RETURN IF NO ERROR
;(3) NOP RETURN IF ERROR
;(4) ERROR ERROR DEFINED BY SUBROUTINE
DEVSEL:
;CLEAR USER'S ERROR CALL
ADD #4,(SP) ;MOVE SP TO USER'S ERROR
CLR8 2(SP) ;CLEAR LOW ORDER BYTE OF CALL
SJB #4,(SP) ;MOVE SP BACK
;SAVE USER'S INFORMATION AND SETUP REGISTERS
MOV ERRVEC-(SP) ;PUSH ERRVEC ON STACK
MOV ERRVEC+2,-(SP) ;PUSH ERRVEC+2 ON STACK
MOV RO,-(SP) ;PUSH RO ON STACK
MOV R1,-(SP) ;PUSH R1 ON STACK
MOV #20$,ERRVEC ;SETUP FOR BUS TIMEOUT
MOV #PR6,ERRVEC+2
MOV $BASE,RO ;RO = UNIBUS ADDRESS
MOV YSTQUE,R1 ;R1 POINTS TO DEVICE NUMBER
;SELECT DEVICE AND VERIFY THAT DEVICE IS AVAILABLE
MOVB (R1),RMCS2(RO) ;WRITE UNIT SELECT BITS
MOV RMCS1(RO),$TMP1 ;GET "DVA" STATUS
MOV RMCS2(RO),$TMP0 ;GET "NED" STATUS
BIT #NED,$TMP0 ;IS DEVICE NONEXISTENT??
BEQ 10$ ;NO!!
ADD #4,10(SP) ;MOVE SP TO USERS ERROR CALL
MOVB #111,210(SP) ;WRITE ERROR NUMBER
BR 30$
BIT #DVA,$TMP1 ;IS DEVICE AVAILABLE??
BNE 35$ ;YES!!
ADD #4,10(SP)
MOVB #112,210(SP)
BR 30$
20$:
;HANDLE BUS TIMEOUT
CMP (SP)+,(SP)+ ;ADJUST SP
ADD #4,10(SP)
MOVB #113,210(SP)
30$: SUB #2,10(SP) ;MOVE SP TO RETURN IF ERROR
35$:
;RESTORE USERS DATA AND RETURN TO ADDRESS ON STACK
MOV (SP)+,R1 ;POP STACK INTO R1
MOV (SP)+,RO ;POP STACK INTO RO

```


CZRMEBO RMO3/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 221
DEVICE SELECT SUBROUTINE

SEQ 0221

11188 051426 012637 000006
11189 051432 012637 000004
11190 051436 000207

MOV (SP)+,ERRVEC+2 ;:POP STACK INTO ERRVEC+2
MOV (SP)+,ERRVEC ;:POP STACK INTO ERRVEC
RTS PC ;EXIT

```

11191 .SBTTL SEEK STATUS CHECK SUBROUTINE
11192
11193 ;THIS SUBROUTINE VERIFIES THE RESULTS OF SEEK TESTS USING STATUS
11194 ;STORED IN THE GET BUFFER AND TEST PARAMETERS STORED IN THE PUT BUFFER.
11195
11196
11197 ;THE SUBROUTINE RETURNS TO THE CALLING ROUTINE IF AN ERROR IS DETECTED
11198 ;AFTER HAVING LOADED THE APPROPRIATE ERROR NUMBER IN THE "ERROR" TRAP
11199 ;OF THE CALLING ROUTINE. SEEK STATUS IS CHECKED AS FOLLOWS:
11200
11201 CALL:
11202 (1) JSR PC,SEKSTS
11203 BR ??? RETURN HERE IF NO ERROR
11204 NOP RETURN HERE TO REPORT AN ERROR
11205 ERROR ERROR NUMBER DEFINED BY SUB
11206 JSR PC,2(SP)+ GO BACK TO SUB FOR MORE ERROR CHECKS
11207 ??? RETURN HERE IF NO MORE ERRORS
11208
11209 051440 SEKSTS:
11210
11211 ;CLEAR USER'S ERROR CALL
11212 051440 000240 NOP
11213 051442 062716 000074 ADD #4,(SP) ;MOVE (SP) TO ERROR CALL
11214 051446 105076 000000 CLRB 2(SP) ;CLEAR ERROR NUMBER
11215 051452 162716 000004 SUB #4,(SP) ;MOVE (SP) TO NO ERROR RETURN
11216 051456 005037 052676 CLR 300$ ;CLEAR ERROR FLAGS
11217
11218 ;TEST FOR MASSBUS CONTROL BUS PARITY ERROR WHEN WRITING
11219 ;LOCAL REGISTERS I.E. "PAR" = 1 AND "DPE" = 0
11220 051462 032737 000010 001344 BIT #PAR,RMER1I ;WAS PARITY ERROR DETECTED??
11221 051470 001424 BEQ 1$ ;NO!!
11222 051472 032737 000010 001372 BIT #DPE,RMER2I ;WAS IT DUE TO CONTROL BUS??
11223 051500 001020 BNE 1$ ;NOT SURE!!
11224
11225 ;REPORT REGISTER PARITY ERROR VIA USER'S ERROR CALL
11226 051502 005037 001140 CLR $GDDAT ;EXPECTED STATUS
11227 051506 013737 001344 001142 MOV RMER1I,$BDDAT ;RECEIVED STATUS
11228 051514 062716 000004 ADD #4,(SP) ;MOVE STACK TO USER'S ERROR
11229 051520 112776 000050 000000 MOVB #50,2(SP) ;ERROR #50
11230 051526 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
11231 051532 004736 JSR PC,2(SP)+
11232 051534 162716 000010 SUB #10,(SP) ;RESTORE STACK
11233 051540 000437 BR 3$ ;IAE SHOULD BE ZERO
11234
11235
11236 ;DETERMINE THE VALUE OF "IAE" STATUS BASED ON TRACK, SECTOR AND
11237 ;CYLINDER ADDRESS USED DURING SEEK OPERATION
11238 051542 012737 002000 001140 1$: MOV #IAE,$GDDAT ;SET UP FOR IAE = 1
11239 051550 052737 040000 052676 BIS #SKI,300$ ;SET SKI ERROR FLAG
11240 051556 023727 001434 001466 CMP RMDCO,#822. ;CYLINDER > 822??
11241 051564 101025 BHI 3$ ;YES!!
11242 051566 042737 040000 052676 BIC #SKI,300$ ;CLEAR SKI ERROR FLAG
11243 051574 123727 001406 000037 CMPB RMDAO,#31. ;SECTOR > 31??
11244 051602 101016 BHI 3$ ;YES!!
11245 051604 123727 001406 000035 CMPB RMDAO,#29. ;SECTOR > 29 ??
11246 051612 101404 BLGS 2$ ;NO!!

```

```

11247 051614 032737 010000 001432          BIT      #FMT16,RMOFO    ;30 SECTOR FORMAT??
11248 051622 001406          BEQ      3$            ;YES!!
11249 051624 123727 001407 000004 2$:      CMPB    RMDAO+1,#4.    ;TRACK >4??
11250 051632 101002          BHI      3$            ;YES!!
11251 051634 005037 001140          CLR      $GDDAT        ;"IAE" SHOULD BE 0
11252
11253          ;COMPARE EXPECTED AND RECEIVED "IAE" STATUS
11254 051640 013737 001344 001142 3$:      MOV     RMER1I,$BDDAT  ;IS IAE OK??
11255 051646 042737 175777 001142          BIC     #+CIAE,$BDDAT
11256 051654 023737 001140 001142          CMP     $GDDAT,$BDDAT
11257 051662 001004          BNE     35$           ;IAE IN ERROR
11258 051664 042737 040000 052676          BIC     #SKI,300$     ;CLEAR SKI FLAG
11259 051672 000413          BR      5$            ;GO CHECK NEXT ERROR
11260 051674
11261          35$:
11262          ;REPORT INCORRECT "IAE" STATUS VIA USER'S ERROR CALL
11263 051674 062716 000004          ADD     #4,(SP)
11264 051700 112776 000051 000000          MOVB   #51,2(SP)     ;ERROR 51
11265 051706 162716 000002          SUB     #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
11266 051712 004736          JSR    PC,2(SP)+     ;REPORT INCORRECT IAE
11267 051714 162716 000010          SUB     #10,(SP)    ;RESTORE (SP)
11268 051720 000240          NOP
11269
11270          5$:
11271          ;REPORT ANY IVC ERROR AS
11272          ; IVC ERROR WITH VOLUME VALID ZERO
11273          ; ERRONEOUS IVC ERROR, VOLUME VALID IS SET
11274 051722 032737 010000 001372          BIT     #IVC,RMER2I   ;IVC ERROR??
11275 051730 001427          BEQ     52$           ;NO!!
11276 051732 005037 001140          CLR     $GDDAT        ;EXPECTED STATUS
11277 051736 013737 001372 001142          MOV     RMER2I,$BDDAT ;RECEIVED STATUS
11278 051744 062716 000004          ADD     #4,(SP)      ;MOVE SP TO USER'S ERROR
11279 051750 112776 000060 000000          MOVB   #60,2(SP)    ;ERROR 60 IF VV = 0
11280 051756 032737 000100 001342          BIT     #VV,RMDSI
11281 051764 001403          BEQ     51$           ;ERROR 61 IF VV = 1
11282 051766 112776 000061 000000          MOVB   #61,2(SP)    ;MOVE SP TO RETURN FOR ERROR
11283 051774 162716 000002 51$:      SUB     #2,(SP)      ;REPORT ERROR VIA USER
11284 052000 004736          JSR    PC,2(SP)+     ;RESTORE SP
11285 052002 162716 000010          SUB     #10,(SP)
11286 052006 000240          NOP
11287 052010 013737 001372 001142 52$:      MOV     RMER2I,$BDDAT ;RECEIVED STATUS
11288 052016 042737 137777 001142          BIC     #+CSKI,$BDDAT ;CLEAR DONT CARES
11289 052024 013737 052676 001140          MOV     300$,$GDDAT  ;GET EXPECTED SKI STATUS
11290 052032 042737 137777 001140          BIC     #+CSKI,$GDDAT ;CLEAR DONT CARES
11291 052040 001417          BEQ     53$           ;BRANCH IF 0 EXPECTED
11292
11293          ;REPORT ERROR IF SKI IS NOT SET (IAE WAS NOT DETECTED)
11294 052042 032737 040000 001142          BIT     #SKI,$BDDAT  ;WAS SKI DETECTED ??
11295 052050 001032          BNE     54$           ;YES !!
11296 052052 062716 000004          ADD     #4,(SP)      ;MOVE SP TO USERS ERROR CALL
11297 052056 112776 000267 000000          MOVB   #267,2(SP)   ;WRITE ERROR NUMBER
11298 052064 162716 000002          SUB     #2,(SP)      ;MOVE SP TO ERROR RETURN
11299 052070 004736          JSR    PC,2(SP)+     ;REPORT ERROR AND RETURN
11300 052072 162716 000010          SUB     #10,(SP)    ;MOVE SP TO NO ERROR
11301 052076 000443          BR      6$            ;GO TO NEXT ERROR CHECK
11302 052100          53$:

```

```

11303
11304 ;REPORT ERROR IF SKI IS SET
11305 052100 032737 040000 001142 BIT #SKI,$BDDAT ;IS SKI SET ??
11306 052106 001413 BEQ 54$ ;NO - SKI IS OK
11307 052110 062716 000004 ADD #4,(SP) ;MOVE (SP) TO ERROR
11308 052114 112776 000054 000000 MOVB #54,@(SP) ;LOAD ERROR NUMBER
11309 052122 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
11310 052126 004736 JSR PC,@(SP)+ ;REPORT SEEK ERROR
11311 052130 162716 000010 SUB #10,(SP) ;RESTORE (SP)
11312 052134 000240 NOP
11313
11314 ;REPORT ANY DEVICE CHECK
11315 052136 032737 000200 001372 54$: BIT #DVC,RMER2I ;WAS THERE DVC DURING SEEK??
11316 052144 001420 BEQ 6$ ;NO!!
11317 052146 005037 001140 CLR $GDDAT ;EXPECTED STATUS
11318 052152 013737 001372 001142 MOV RMER2I,$BDDAT ;RECEIVED STATUS
11319 052160 062716 000004 ADD #4,(SP)
11320 052164 112776 000055 000000 MOVB #55,@(SP) ;ERROR #55
11321 052172 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
11322 052176 004736 JSR PC,@(SP)+ ;REPORT ERROR VIA USER
11323 052200 162716 000010 SUB #10,(SP) ;RESTORE SP
11324 052204 000240 NOP
11325
11326 ;REPORT ANY "OPI" ERROR AS OPI WITH MOL =0, OR OPI
11327 ;BECHUSE ON CYLINDER LATCH DIDN'T RESET
11328 052206 032737 020000 001344 6$: BIT #OPI,RMER1I ;"OPI" ERROR??
11329 052214 001427 BEQ 8$ ;NO!!
11330 052216 005037 001140 CLR $GDDAT ;EXPECTED STATUS
11331 052222 013737 001344 001142 MOV RMER1I,$BDDAT ;RECEIVED STATUS
11332 052230 062716 000004 ADD #4,(SP) ;MOVE (SP) TO ERROR
11333 052234 112776 000052 000000 MOVB #52,@(SP) ;LOAD ERROR NUMBER
11334 052242 032737 010000 001342 BIT #MOL,RMDSI ;WAS MEDIUM ON LINE??
11335 052250 001403 BEQ 7$ ;NO!!
11336 052252 112776 000053 000000 MOVB #53,@(SP) ;YES - CHANGE ERROR NUMBER
11337 052260 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
11338 052264 004736 JSR PC,@(SP)+ ;REPORT "OPI" ERROR
11339 052266 162716 000010 SUB #10,(SP) ;RESTORE (SP)
11340 052272 000240 NOP
11341
11342 ;SEE IF "PIP" IS 0, AND "ATA","MOL"AND"VV" =1
11343 052274 013746 001342 8$: MOV RMDSI,-(SP)
11344 052300 042716 047677 BIC #<ATA!PIP!MOL!VV>,(SP)
11345 052304 022726 110100 CMP #ATA!MOL!VV,(SP)+
11346 052310 001002 BNE 9$ ;ERROR IN RMDS
11347 052312 000137 052646 JMP 14$ ;RMDS IS OK
11348
11349 ;REPORT ERROR IF MOL = 0 AND OPI = 0
11350 052316 032737 010000 001342 9$: BIT #MOL,RMDSI ;IS MOL RESET??
11351 052324 001030 BNE 10$ ;NO - MOL IS SET
11352 052326 032737 020000 001344 BIT #OPI,RMER1I ;WAS OPI SET
11353 052334 001024 BNE 10$ ;YES - DONT REPORT ERROR
11354 052336 013737 001342 001140 MOV RMDSI,$GDDAT ;EXPECTED STATUS
11355 052344 052737 010000 001140 BIS #MOL,$GDDAT
11356 052352 013737 001342 001142 MOV RMDSI,$BDDAT ;RECEIVED STATUS
11357 052360 062716 000004 ADD #4,(SP)
11358 052364 112776 000062 000000 MOVB #62,@(SP)

```

```

11359 052372 162716 000002          SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
11360 052376 004736          JSR      PC,@(SP)+      ;REPORT ERROR VIA USER
11361 052400 162716 000010          SUB      #10,(SP)
11362 052404 000240          NOP
11363
11364          ;REPORT AN ERROR IF "PIP" IS STIL SET AND SKI NOT SET
11365 052406 032737 020000 001342 10$: BIT      #PIP,RMDSI      ;IS "PIP" STILL SET??
11366 052414 001430          BEQ      11$            ;NO!!
11367 052416 032737 040000 001372  BIT      #SKI,RMER2I     ;WAS "SKI"SET??
11368 052424 001024          BNE      11$            ;YES-DONT REPORT PIP
11369 052426 013737 001342 001140  MOV      RMDSI,$GDDAT    ;EXPECTED STATUS
11370 052434 042737 020000 001142  BIC      #PIP,$BDDAT
11371 052442 013737 001342 001142  MOV      RMDSI,$BDDAT    ;RECEIVED STATUS
11372 052450 062716 000004          ADD      #4,(SP)        ;MOVE (SP) TO ERROR
11373 052454 112776 000056 000000  MOVB     #56,@(SP)      ;LOAD ERROR NUMBER
11374 052462 162716 000002          SUB      #2,(SP)        ;MOVE SP TO RETURN FOR ERROR
11375 052466 004736          JSR      PC,@(SP)+      ;REPORT "PIP" SET AFTER SEEK
11376 052470 162716 000010          SUB      #10,(SP)      ;RESTORE (SP)
11377 052474 000240          NOP
11378
11379          ;REPORT AN ERROR IF "ATA" IS NOT SET
11380 052476 032737 100000 001342 11$: BIT      -ATA,RMDSI    ;WAS "ATA" SET ??
11381 052504 001024          BNE      13$            ;YES!!
11382 052506 013737 001342 001140  MOV      RMDSI,$GDDAT    ;EXPECTED STATUS
11383 052514 052737 110600 001140  BIS      #ATA!MOL!DPR!DRY,$GDDAT
11384 052522 013737 001342 001142  MOV      RMDSI,$BDDAT    ;RECEIVED STATUS
11385 052530 062716 000004          ADD      #4,(SP)        ;MOVE (SP) TO ERROR
11386 052534 112776 000057 000000  MOVB     #57,@(SP)      ;LOAD ERROR NUMBER
11387 052542 162716 000002          SUB      #2,(SP)        ;MOVE SP TO RETURN FOR ERROR
11388 052546 004736          JSR      PC,@(SP)+      ;REPORT ATTENTION NOT SET DURING
11389          ;SEEK TEST
11390 052550 162716 000010          SUB      #10,(SP)      ;RESTORE (SP)
11391 052554 000240          NOP
11392
11393          ;REPORT ERROR IF VOLUME VALID IS RESET AND IVC IS ZERO
11394 052556 032737 000100 001342 13$: BIT      #VV,RMDSI      ;IS VV = 0 ??
11395 052564 001030          BNE      14$            ;NO!!
11396 052566 032737 010000 001372  BIT      #IVC,RMER2I     ;IS IVC ALSO 0 ??
11397 052574 001024          BNE      14$            ;NO - IVC IS SET
11398 052576 013737 001342 001140  MOV      RMDSI,$GDDAT    ;EXPECTED STATUS
11399 052604 052737 000100 001140  BIS      #VV,$GDDAT
11400 052612 013737 001342 001142  MOV      RMDSI,$BDDAT    ;RECEIVED STATUS
11401 052620 062716 000004          ADD      #4,(SP)
11402 052624 112776 000064 000000  MOVB     #64,@(SP)      ;ERROR #64
11403 052632 162716 000002          SUB      #2,(SP)        ;MOVE SP TO RETURN FOR ERROR
11404 052636 004736          JSR      PC,@(SP)+
11405 052640 162716 000010          SUB      #10,(SP)
11406 052644 000240          NOP
11407 052646
11408
11409          ;MODIFY THE RETURN ADDRESS IF AN ERROR WAS DETECTED
11410 052646 000240          NOP
11411 052650 062716 000004          ADD      #4,(SP)        ;MOVE (SP) TO ERROR CALL
11412 052654 105776 000000  TSTB     @(SP)          ;WAS ERROR CALLED??
11413 052660 001403          BEQ      15$            ;NO!!
11414 052662 062716 000004          ADC      #4,(SP)        ;MOVE TO ERROR RETURN

```

G02

CZRMEBO RMD3/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 226
SEEK STATUS CHECK SUBROUTINE

SEQ 0226

11415	052666	000402		BR	16\$	
11416						
11417	052670	162716	000004	15\$:	SUB	#4, (SP)
11418	052674	000207		16\$:	RTS	PC
11419						; MOVE (SP) TO NO ERROR RETURN
11420	052676	000000		300\$:	.WORD	0
11421						; ERROR FLAGS

```

11422
11423
11424
11425
11426
11427
11428
11429
11430
11431
11432
11433 052700
11434 052700 010046
11435 052702 010146
11436 052704 013746 000004
11437 052710 013746 000006
11438 052714 012737 052754 000004
11439 052722 012737 000300 000006
11440 052730 013700 001276
11441 052734 012760 000040 000010
11442 052742 013701 001450
11443 052746 111160 000010
11444 052752 000412
11445 052754 022626
11446 052756 062766 000004 000010
11447 052764 112776 000007 000010
11448 052772 162766 000002 000010
11449 053000
11450 053000 012637 000006
11451 053004 012637 000004
11452 053010 012601
11453 053012 012600
11454 053014 000207
11455

.SBTTL CONTROLLER CLEAR SUBROUTINE
; THIS SUBROUTINE CLEARS THE MASSBUS CONTROLLER, MASSBUS ADAPTERS,
; AND DRIVES, THEN SELECTS THE DRIVE.
:CALL: JSR PC,CNTCLR
:      BR ??? RETURN HERE IF NO ERROR FOUND
:      NOP RETURN HERE IF ANY ERROR FOUND
:      ERROR SUB DEFINES ERROR NUMBER
:      ???

CNTCLR:
MOV RO,-(SP) ;; PUSH RO ON STACK
MOV R1,-(SP) ;; PUSH R1 ON STACK
MOV ERRVEC,-(SP) ;; PUSH ERRVEC ON STACK
MOV ERRVEC+2,-(SP) ;; PUSH ERRVEC+2 ON STACK
MOV #10$,ERRVEC ; SETUP FOR BUS TIMEOUT
MOV #PR6,ERRVEC+2
MOV $BASE,RO ; RO=UNIBUS ADDRESS
MOV #CLR,AMCS2(RO) ; CLEAR MASSBUS
MOV TSTQUE,R1
MOVB (R1),RMCS2(RO) ; SELECT DEVICE
BR 20$
10$: CMP (SP)+,(SP)+ ; ADJUST STACK
ADD #4,10(SP) ; MOVE SP TO USER'S ERROR CALL
MOVB #7,210(SP) ; WRITE THE ERROR NUMBER
SUB #2,10(SP)
20$: MOV (SP)+,ERRVEC+2 ;; POP STACK INTO ERRVEC+2
MOV (SP)+,ERRVEC ;; POP STACK INTO ERRVEC
MOV (SP)+,R1 ;; POP STACK INTO R1
MOV (SP)+,RO ;; POP STACK INTO RO
PC RTS

```

11456
11457
11458
11459
11460
11461
11462
11463
11464
11465
11466
11467
11468
11469
11470
11471
11472
11473
11474
11475
11476
11477 053016
11478
11479
11480 053016 062716 000004
11481 053022 105076 000000
11482 053026 162716 000004
11483
11484 053032 013737 001330 001142
11485 053040 042737 100000 001142
11486 053046 012737 004200 001140
11487 053054 023737 001140 001142
11488 053062 001413
11489 053064 062716 000004
11490 053070 112776 000126 000000
11491 053076 162716 000002
11492 053102 004736
11493 053104 162716 000010
11494 053110 000240
11495
11496 053112 005037 001140
11497 053116 013737 001334 001142
11498 053124 001413
11499 053126 062716 000004
11500 053132 112776 000127 000000
11501 053140 162716 000002
11502 053144 004736
11503 053146 162716 000010
11504 053152 000240
11505
11506 053154 013737 001340 001142
11507 053162 010146
11508 053164 005046
11509 053166 013701 001450
11510 053172 111116
11511 053174 052716 000100

```

.SBTTL CONTROLLER CLEAR STATUS CHECK SUBROUTINE

;THIS SUBROUTINE VERIFIES THAT THE RMO3 SUBSYSTEM IS INITIALIZED BASED ON
;STATUS STORED IN THE GET BUFFER. THIS SUBROUTINE SHOULD ONLY BE
;USED FOLLOWING A CONTROLLER CLEAR OPERATION, I.E. WRITING A 1 IN BIT
;5 OF RMCS2, BECAUSE THE ERROR MESSAGES ARE BASED ON THAT CONDITION

;STATUS PERTINENT TO THE DEVICE IS NOT CHECKED. IN PARTICULAR, THE
;FOLLOWING STATUS BITS ARE NOT CHECKED:
:
:       ATA,ERR,PIP,MOL,WRL,LBT,PGM,VV,OM,UNS,SKI,DVC
:
:CALL:
:(1)   JSR      PC,CLRSTS
:       BR      ???          RETURN HERE IF NO ERROR
:       NOP
:       ERROR   RETURN HERE TO REPORT AN ERROR
:       JSR      PC,@(SP)+   ERROR NUMBER DEFINED BY SUB
:       ???          GO BACK TO SUB FOR MORE ERROR CHECKS
:                       RETURN HERE IF NO MORE ERRORS

CLRSTS:
:CLEAR USER'S ERROR CALL
:       ADD     #4,(SP)      ;MOVE SP TO ERROR
:       CLR    @2(SP)      ;CLEAR ERROR NUMBER
:       SUB    #4,(SP)      ;MOVE SP BACK TO NO ERROR

;REPORT ERROR IF RMCS1 NOT INITIALIZED
4$:   MOV     RMCS1I,$BDDAT ;VERIFY RMCS1
:       BIC    #5C,$BDDAT  ;IGNORE SPECIAL CONDITION
:       MOV    #DVA!RDY,$GDDAT ;EXPECT DVA & RDY
:       CMP    $GDDAT,$BDDAT ;COMPARE EXPECTED, RECEIVED
:       BEQ   7$           ;BRANCH IF EQUAL
:       ADD    #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
:       MOV    #126,@2(SP) ;WRITE ERROR NUMBER IN CALL
:       SUB    #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
:       JSR    PC,@(SP)+   ;REPORT ERROR VIA USER
:       SUB    #10,(SP)     ;MOVE SP BACK TO NO ERROR
:       NOP

;REPORT ERROR IF RMBA NOT RESET
5$:   CLR    $GDDAT        ;VERIFY RMBA IS ZERO
:       MOV    RMBAI,$BDDAT
:       BEQ   7$           ;BRANCH IF ZERO
:       ADD    #4,(SP)      ;MOVE SP TO USER'S ERROR CALL
:       MOV    #127,@2(SP) ;WRITE ERROR NUMBER IN CALL
:       SUB    #2,(SP)      ;MOVE SP TO RETURN FOR ERROR
:       JSR    PC,@(SP)+   ;REPORT ERROR VIA USER
:       SUB    #10,(SP)     ;MOVE SP BACK TO NO ERROR
:       NOP

;REPORT ERROR IF RMCS2 NOT INITIALIZED
7$:   MOV    RMCS2I,$BDDAT ;VERIFY RMCS2
:       MOV    R1,-(SP)     ;PUSH R1 ON STACK
:       CLR    -(SP)        ;EXPECT IR & UNIT NUMBER
:       MOV    TSTQUE,R1   ;R1 = ADDRESS OF TEST QUE
:       MOV    (R1),(SP)
:       BIS   #IR,(SP)
    
```



```

11512 053200 012637 001140      MOV      (SP)+,$GDDAT
11513 053204 012601      MOV      (SP)+,R1          ; POP STACK INTO R1
11514 053206 023737 001140 001142  CMP      $GDDAT,$BDDAT    ; COMPARE EXPECTED & RECEIVED
11515 053214 001413      BEQ      9$              ; BRANCH IF EQUAL
11516 053216 062716 000004      ADD      #4,(SP)          ; MOVE SP TO USER'S ERROR CALL
11517 053222 112776 000130 000000  MOVVB   #130,2(SP)        ; WITE ERROR NUMBER IN CALL
11518 053230 162716 000002      SUB      #2,(SP)          ; MOVE SP TO RETURN FOR ERROR
11519 053234 004736      JSR      PC,2(SP)+        ; REPORT ERROR VIA USER
11520 053236 162716 000010      SUB      #10,(SP)         ; MOVE SP BACK TO NO ERROR
11521 053242 000240      NOP
11522      ;REPORT ERROR IF RMER1 NOT RESET-IGNORE UNS
11523 053244 005037 001140 9$:      CLR      $GDDAT          ; VERIFY RMER1
11524 053250 013737 001344 001142  MOV      RMER1I,$BDDAT
11525 053256 042737 040000 001142  BIC      #UNS,$BDDAT      ; IGNORE UNSAFE
11526 053264 001413      BEQ      13$            ; BRANCH IF ZERO
11527 053266 062716 000004      ADD      #4,(SP)          ; MOVE SP TO USER'S ERROR CALL
11528 053272 112776 000131 000000  MOVVB   #131,2(SP)        ; WITE ERROR NUMBER IN CALL
11529 053300 162716 000002      SUB      #2,(SP)          ; MOVE SP TO RETURN FOR ERROR
11530 053304 004736      JSR      PC,2(SP)+        ; REPORT ERROR VIA USER
11531 053306 162716 000010      SUB      #10,(SP)         ; MOVE SP BACK TO NO ERROR
11532 053312 000240      NOP
11533      ;REPORT ERROR IF RMMR1 NOT INITIALIZED-IGNORE WC,LS,LST
11534 053314 013737 001354 001142 13$:    MOV      RMMR1I,$BDDAT    ; VERIFY RMMR
11535 053322 042737 000046 001142  BIC      #WC!LS!LST,$BDDAT ; IGNORE WORD CLOCK, SCT, TRK
11536 053330 012737 000010 001140  MOV      #MWD,$GDDAT      ; EXPECT WRITE DATA BIT
11537 053336 023737 001140 001142  CMP      $GDDAT,$BDDAT    ; COMPARE EXPECTED AND RECEIVED
11538 053344 001413      BEQ      17$            ; BRANCH IF 0
11539 053346 062716 000004      ADD      #4,(SP)          ; MOVE SP TO USER'S ERROR CALL
11540 053352 112776 000133 000000  MOVVB   #133,2(SP)        ; WITE ERROR NUMBER IN CALL
11541 053360 162716 000002      SUB      #2,(SP)          ; MOVE SP TO RETURN FOR ERROR
11542 053364 004736      JSR      PC,2(SP)+        ; REPORT ERROR VIA USER
11543 053366 162716 000010      SUB      #10,(SP)         ; MOVE SP BACK TO NO ERROR
11544 053372 000240      NOP
11545      ;REPORT AN ERROR IF RMEC2 IS NOT RESET
11546 053374 005037 001140 17$:    CLR      $GDDAT          ; EXPECT ZEROS
11547 053400 013737 001376 001142  MOV      RMEC2I,$BDDAT    ; VERIFY RMEC2=0
11548 053406 001413      BEQ      19$            ; BRANCH IF 0
11549 053410 062716 000004      ADD      #4,(SP)          ; MOVE SP TO USER'S ERROR CALL
11550 053414 112776 000135 000000  MOVVB   #135,2(SP)        ; WITE ERROR NUMBER IN CALL
11551 053422 162716 000002      SUB      #2,(SP)          ; MOVE SP TO RETURN FOR ERROR
11552 053426 004736      JSR      PC,2(SP)+        ; REPORT ERROR VIA USER
11553 053430 162716 000010      SUB      #10,(SP)         ; MOVE SP BACK TO NO ERROR
11554 053434 000240      NOP
11555      ;REPORT ERROR IF RMMR2 NOT INITIALIZED-IGNORE RQA,RQB
11556 053436 013737 001370 001142 19$:    MOV      RMMR2I,$BDDAT    ; VERIFY RMMR2
11557 053444 042737 140000 001142  BIC      #RQA!RQB,$BDDAT
11558 053452 012737 011777 001140  MOV      #TST!1777,$GDDAT ; EXPECT TEST BIT ON
11559 053460 023737 001140 001142  CMP      $GDDAT,$BDDAT
11560 053466 001413      BEQ      21$            ; BRANCH IF 0
11561 053470 062716 000004      ADD      #4,(SP)          ; MOVE SP TO USER'S ERROR CALL
11562 053474 112776 000136 000000  MOVVB   #136,2(SP)        ; WITE ERROR NUMBER IN CALL
11563 053502 162716 000002      SUB      #2,(SP)          ; MOVE SP TO RETURN FOR ERROR
11564 053506 004736      JSR      PC,2(SP)+        ; REPORT ERROR VIA USER
11565 053510 162716 000010      SUB      #10,(SP)         ; MOVE SP BACK TO NO ERROR
11566 053514 000240      NOP
11567      ;REPORT ERROR IF RMER2 NOT RESET-IGNORE SKI,DVC

```

```

11568 053516 005037 001140      21$: CLR      $GDDAT      ; EXPECT ALL ZEROS
11569 053522 013737 001372 001142  MOV      RMER2I,$BDDAT ; VERIFY RMER2
11570 053530 042737 040200 001142  BIC      #SKI!DVC,$BDDAT ; IGNORE DEVICE ERRORS
11571 053536 001413      215$ BEQ      #4,(SP)      ; BRANCH IF OTHER BITS 0
11572 053540 062716 000004      ADD      #4,(SP)      ; MOVE SP TO USER'S ERROR CALL
11573 053544 112776 000174 000000  MOVB    #174,2(SP)    ; WRITE ERROR NUMBER IN CALL
11574 053552 162716 000002      SUB      #2,(SP)      ; MOVE SP TO RETURN FOR ERROR
11575 053556 004736      JSR      PC,2(SP)+    ; REPORT ERROR VIA USER
11576 053560 162716 000010      SUB      #10,(SP)     ; MOVE SP BACK TO NO ERROR
11577 053564 000240      NOP
11578      ;REPORT ERROR IF RMD5 NOT INITIALIZED
11579 053566 013737 001342 001142 215$: MOV      RMD5I,$BDDAT ; TEST DRIVE STATUS REGISTER
11580 053574 042737 177177 001142  BIC      #1C(DRY!DPR),$BDDAT
11581 053602 012737 000600 001140  MOV      #DPR!DRY,$GDDAT ; EXPECTED DRIVE STATUS
11582 053610 023737 001140 001142  CMP      $GDDAT,$BDDAT ; COMPARE EXPECTED & RECEIVED
11583 053616 001413      22$ BEQ      #4,(SP)      ; BRANCH IF EQUAL
11584 053620 062716 000004      ADD      #4,(SP)      ; MOVE SP TO USER'S ERROR CALL
11585 053624 112776 000134 000000  MOVB    #134,2(SP)    ; WRITE ERROR NUMBER
11586 053632 162716 000002      SUB      #2,(SP)      ; MOVE SP TO RETURN FOR ERROR
11587 053636 004736      JSR      PC,2(SP)+    ; REPORT ERROR TO USER
11588 053640 162716 000010      SUB      #10,(SP)     ; MOVE SP BACK TO NO ERROR
11589 053644 000240      NOP
11590 053646 062716 000004      22$: ADD      #4,(SP)      ; MOVE SP TO ERROR CALL
11591 053652 105776 000000      TSTB    2(SP)        ; WAS AN ERROE DETECTED??
11592 053656 001403      BEQ      23$         ; NO!!
11593 053660 062716 000004      ADD      #4,(SP)      ; YES - MOVE TO ERROR RETURN
11594 053664 000402      BR      24$         ;
11595 053666 162716 000004      23$: SUB      #4,(SP)      ; MOVE SP TO NO ERROR RETURN
11596 053672 000240      24$: NOP
11597 053674 000207      RTS      PC

```

.SBTTL PACK ACKNOWLEDGE STATUS CHECK

; THIS SUBROUTINE CHECKS THE RESULTS OF A PACK ACKNOWLEDGE
; COMMAND USING THE STATUS STORED IN THE GET BUFFER. ERRORS ARE
; REPORTED TO THE USER VIA THE USER'S ERROR CALL.

; CALL:
; (1) JSR PC,ACKSTS
; BR ??? RETURN HERE IF NO ERROR
; NOP RETURN HERE TO REPORT AN ERROR
; ERROR ERROR NUMBER DEFINED BY SUB
; JSR PC,@(SP)+ GO BACK TO SUB FOR MORE ERROR CHECKS
; ??? RETURN HERE IF NO MORE ERRORS

ACKSTS:

; CLEAR USER'S ERROR CALL
ADD #4,(SP) ; MOVE SP TO ERROR CALL
CLRB @(SP) ; CLEAR LOW ORDER BYTE
SUB #4,(SP) ; MOVE SP BACK

; REPORT AN ERROR IF "VV" IS 0
BIT #VV,RMDSI ; IS VOLUME VALID SET??
BNE 1\$; YES!!
MOV RMDSI,\$GDDAT ; EXPECTED STATUS
BIS #VV,\$GDDAT
MOV RMDSI,\$BDDAT ; RECEIVED STATUS
ADD #4,(SP) ; MOVE SP TO ERROR CALL
MOVB #155,@(SP) ; WRITE NUMBER IN ERROR CALL
SUB #2,(SP) ; MOVE SP TO RETURN FOR ERROR
JSR PC,@(SP)+ ; REPORT THE ERROR
SUB #10,(SP) ; MOVE SP BACK TO BRANCH
NOP

1\$:

; REPORT AN ERROR IF "MOL" IS 0
BIT #MOL,RMDSI ; IS MOL SET??
BNE 2\$; YES!!
MOV RMDSI,\$GDDAT ; EXPECTED STATUS
BIS #MOL,\$GDDAT
MOV RMDSI,\$BDDAT ; RECEIVED STATUS
ADD #4,(SP) ; MOVE SP TO ERROR CALL
MOVB #41,@(SP) ; WRITE NUMBER OF ERROR IN CALL
SUB #2,(SP) ; MOVE SP TO RETURN FOR ERROR
JSR PC,@(SP)+ ; REPORT TH ERROR
SUB #10,(SP) ; MOVE SP TO BRANCH
NOP

2\$:

; SEE IF "UNS", "OPI", "RMR", "ILR", OR "ILF" IS SET
BIT #UNS!OPI!RMR!ILR!ILF,RMER11
BEQ 7\$

; REPORT AN ERROR IF "UNS" IS SET
BIT #UNS,RMER11 ; WAS UNS SET??
BEQ 3\$; NO!!

11598
11599
11600
11601
11602
11603
11604
11605
11606
11607
11608
11609
11610
11611
11612 053676
11613
11614
11615 053676 062716 000004
11616 053702 105076 000000
11617 053706 162716 000004
11618
11619
11620 053712 032737 000100 001342
11621 053720 001024
11622 053722 013737 001342 001140
11623 053730 052737 000100 001140
11624 053736 013737 001342 001142
11625 053744 062716 000004
11626 053750 112776 000155 000000
11627 053756 162716 000002
11628 053762 004736
11629 053764 162716 000010
11630 053770 000240
11631 053772
11632
11633
11634 053772 032737 010000 001342
11635 054000 001024
11636 054002 013737 001342 001140
11637 054010 052737 010000 001140
11638 054016 013737 001342 001142
11639 054024 062716 000004
11640 054030 112776 000041 000000
11641 054036 162716 000002
11642 054042 004736
11643 054044 162716 000010
11644 054050 000240
11645 054052
11646
11647
11648 054052 032737 060007 001344
11649 054060 001570
11650
11651
11652 054062 032737 040000 001344
11653 054070 001424

11654	054072	013737	001344	001142	MOV	RMER11,\$BDDAT	;RECEIVED STATUS
11655	054100	013737	001344	001140	MOV	RMER11,\$GDDAT	;EXPECTED STATUS
11656	054106	042737	040000	001140	BIC	#UNS,\$GDDAT	
11657	054114	062716	000004		ADD	#4,(SP)	;MOVE SP TO ERROR CALL
11658	054120	112776	000042	000000	MOVB	#42,2(SP)	;WRITE NUMBER OF ERROR IN CALL
11659	054126	162716	000002		SUB	#2,(SP)	;MOVE SP TO RETURN FOR ERROR
11660	054132	004736			JSR	PC,2(SP)+	;REPORT THE ERROR VIA USER
11661	054134	162716	000010		SUB	#10,(SP)	;MOVE SP TO NO ERROR RETURN
11662	054140	000240			NOP		
11663	054142						
11664							
11665							
11666	054142	032737	020000	001344	;REPORT ANY OPI ERROR		
11667	054150	001424			BIT	#OPI,RMER11	;WAS OPI SET ??
11668	054152	013737	001344	001142	BEQ	4\$;NO!!
11669	054160	013737	001344	001140	MOV	RMER11,\$BDDAT	;RECEIVED STATUS
11670	054166	042737	020000	001140	MOV	RMER11,\$GDDAT	;EXPECTED STATUS
11671	054174	062716	000004		BIC	#OPI,\$GDDAT	
11672	054200	112776	000043	000000	ADD	#4,(SP)	;MOVE SP TO ERROR CALL
11673	054206	162716	000002		MOVB	#43,2(SP)	;WRITE NUMBER OF ERROR IN CALL
11674	054212	004736			SUB	#2,(SP)	;MOVE SP TO RETURN FOR ERROR
11675	054214	162716	000010		JSR	PC,2(SP)+	;REPORT THE ERROR VIA USER
11676	054220	000240			SUB	#10,(SP)	;MOVE SP TO NO ERROR RETURN
11677	054222				NOP		
11678							
11679							
11680	054222	032737	000004	001344	;REPORT ANY RMR ERROR		
11681	054230	001424			BIT	#RMR,RMER11	;WAS RMR SET??
11682	054232	013737	001344	001142	BEQ	5\$;NO!!
11683	054240	013737	001344	001140	MOV	RMER11,\$BDDAT	;RECEIVED STATUS
11684	054246	042737	000004	001140	MOV	RMER11,\$GDDAT	;EXPECTED STATUS
11685	054254	062716	000004		BIC	#RMR,\$GDDAT	
11686	054260	112776	000044	000000	ADD	#4,(SP)	;MOVE SP TO ERROR CALL
11687	054266	162716	000002		MOVB	#44,2(SP)	;WRITE NUMBER OF ERROR IN CALL
11688	054272	004736			SUB	#2,(SP)	;MOVE SP TO RETURN FOR ERROR
11689	054274	162716	000010		JSR	PC,2(SP)+	;REPORT THE ERROR VIA USER
11690	054300	000240			SUB	#10,(SP)	;MOVE SP TO NO ERROR RETURN
11691	054302				NOP		
11692							
11693							
11694	054302	032737	000002	001344	;REPORT ANY ILR ERROR		
11695	054310	001424			BIT	#ILR,RMER11	;WAS ILR SET??
11696	054312	013737	001344	001142	BEQ	6\$;NO!!
11697	054320	013737	001344	001140	MOV	RMER11,\$BDDAT	;RECEIVED STATUS
11698	054326	042737	000002	001140	MOV	RMER11,\$GDDAT	;EXPECTED STATUS
11699	054334	062716	000004		BIC	#ILR,\$GDDAT	
11700	054340	112776	000045	000000	ADD	#4,(SP)	;MOVE SP TO ERROR CALL
11701	054346	162716	000002		MOVB	#45,2(SP)	;WRITE NUMBER OF ERROR IN CALL
11702	054352	004736			SUB	#2,(SP)	;MOVE SP TO RETURN FOR ERROR
11703	054354	162716	000010		JSR	PC,2(SP)+	;REPORT THE ERROR VIA USER
11704	054360	000240			SUB	#10,(SP)	;MOVE SP TO NO ERROR RETURN
11705	054362				NOP		
11706							
11707							
11708	054362	032737	000001	001344	;REPORT ANY ILF ERROR		
11709	054370	001424			BIT	#ILF,RMER11	;WAS ILF SET??
					BEQ	7\$;NO!!

11710	054372	013737	001344	001142	MOV	RMER11,\$BDDAT	;RECEIVED STATUS
11711	054400	013737	001344	001140	MOV	RMER11,\$GDDAT	;EXPECTED STATUS
11712	054406	042737	000001	001140	BIC	#1LF,\$GDDAT	
11713	054414	062716	000004		ADD	#4,(SP)	;MOVE SP TO ERROR CALL
11714	054420	112776	000046	000000	MOVB	#46,@(SP)	;WRITE NUMBER OF ERROR IN CALL
11715	054426	162716	000002		SUB	#2,(SP)	;MOVE SP TO RETURN FOR ERROR
11716	054432	004736			JSR	PC,@(SP)+	;REPORT THE ERROR VIA USER
11717	054434	162716	000010		SUB	#10,(SP)	;MOVE SP TO NO ERROR RETURN
11718	054440	000240			NOP		
11719	054442						
11720							
11721							
11722	054442	062716	000004				
11723	054446	105776	000000		ADD	#4,(SP)	;MOVE SP TO ERROR CALL
11724	054452	001403			TSTB	@(SP)	;WAS ERROR FOUND??
11725	054454	062716	000004		BEQ	8\$;NO!!
11726	054460	000402			ADD	#4,(SP)	;YES - MOVE TO ERROR RETURN
11727	054462	162716	000004		BR	9\$	
11728	054466	000240		8\$:	SUB	#4,(SP)	;MOVE SP TO NO ERROR RETURN
11729	054470	000207		9\$:	NOP		
11730					RTS	PC	

```

11731
11732
11733
11734
11735
11736
11737
11738
11739
11740
11741
11742
11743
11744
11745 054472
11746
11747
11748 054472 062716 000004
11749 054476 105076 000000
11750 054502 162716 000004
11751
11752
11753
11754 054506 032737 022011 001344
11755 054514 001553
11756
11757
11758
11759 054516 032737 000010 001344
11760 054524 001430
11761 054526 032737 000010 001372
11762 054534 001024
11763 054536 013737 001344 001140
11764 054544 042737 000010 001140
11765 054552 013737 001344 001142
11766 054560 062716 000004
11767 054564 112776 000050 000000
11768 054572 162716 000002
11769 054576 004736
11770 054600 162716 000010
11771 054604 000240
11772 054606
11773
11774
11775 054606 032737 000001 001344
11776 054614 001424
11777 054616 013737 001344 001140
11778 054624 042737 000001 001140
11779 054632 013737 001344 001142
11780 054640 062716 000004
11781 054644 112776 000071 000000
11782 054652 162716 000002
11783 054656 004736
11784 054660 162716 000010
11785 054664 000240
11786 054666

```

```

.SBTTL RECALIBRATE STATUS CHECK SUBROUTINE
;THIS SUBROUTINE CHECKS THE RESULTS OF A RECALIBRATE OPERATION
;USING THE STATUS STORED IN THE GET BUFFER.
;CALL:
(1) JSR PC,RCLSTS ;CALL SUBROUTINE
BR ??? ;RETURN HERE IF NO ERROR
NOP ;RETURN HERE TO REPORT AN ERROR
ERROR ;ERROR NUMBER DEFINED BY SUB
JSR PC,2(SP)+ ;GO BACK TO SUB FOR MORE ERROR CHECKS
??? ;RETURN HERE IF NO MORE ERRORS

RCLSTS:
;CLEAR USER'S ERROR NUMBER
ADD #4,(SP)
CLRB 2(SP) ;CLEAR USER'S ERROR CALL
SUB #4,(SP) ;MOVE SP BACK TO BRANCH

;SEE IF "PAR" OR "ILF" OR "OPI" OR "IAE" IS SET
BIT #OPI!PAR!ILF!IAE,RMER1I
BEQ 4$ ;NONE ARE SET - GO TO NEXT CHECK

;REPORT ANY MASSBUS CONTROL BUS PARITY ERROR, I.E.,
;"PAR" = 1 AND "DPE" = 0
BIT #PAR,RMER1I ;WAS "PAR" SET??
BEQ 1$ ;NO!!
BIT #DPE,RMER2I ;WAS "DPE" SET??
BNE 1$ ;YES - NOT A REGISTER ERROR
MOV RMER1I,$GDDAT ;EXPECTED STATUS
BIC #PAR,$GDDAT
MOV RMER1I,$BDDAT ;RECEIVED STATUS
ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
MOVB #50,2(SP) ;WRITE ERROR NUMBER IN CALL
SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
JSR PC,2(SP)+ ;GO REPORT ERROR
SUB #10,(SP) ;MOVE SP BACK TO BRANCH
NOP

1$:
;REPORT ANY "ILF" ERROR
BIT #ILF,RMER1I ;WAS "ILF" SET??
BEQ 2$ ;NO!!
MOV RMER1I,$GDDAT ;EXPECTED STATUS
BIC #ILF,$GDDAT
MOV RMER1I,$BDDAT ;RECEIVED STATUS
ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
MOVB #71,2(SP) ;WRITE ERROR NUMBER IN CALL
SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
JSR PC,2(SP)+ ;REPORT ERROR VIA USER
SUB #10,(SP) ;MOVE SP BACK TO BRANCH
NOP

2$:

```

```

11787
11788
11789
11790
11791 054666 032737 020000 001344
11792 054674 001433
11793 054676 013737 001344 001140
11794 054704 042737 020000 001140
11795 054712 013737 001344 001142
11796 054720 062716 000004
11797 054724 112776 000072 000000
11798 054732 032737 010000 001342
11799 054740 001403
11800 054742 112776 000073 000000
11801 054750 162716 000002
11802 054754 004736
11803 054756 162716 000010
11804 054762 000240
11805 054764
11806
11807
11808 054764 032737 002000 001344
11809 054772 001424
11810 054774 013737 001344 001140
11811 055002 042737 002000 001140
11812 055010 013737 001344 001142
11813 055016 062716 000004
11814 055022 112776 000070 000000
11815 055030 162716 000002
11816 055034 004736
11817 055036 162716 000010
11818 055042 000240
11819 055044
11820
11821
11822 055044 032737 050200 001372
11823 055052 001517
11824
11825
11826
11827
11828
11829 055054 032737 010000 001372
11830 055062 001433
11831 055064 013737 001372 001140
11832 055072 042737 010000 001140
11833 055100 013737 001372 001142
11834 055106 062716 000004
11835 055112 112776 000074 000000
11836 055120 032737 000100 001342
11837 055126 001403
11838 055130 112776 000075 000000
11839 055136 162716 000002
11840 055142 004736
11841 055144 162716 000010
11842 055150 000240

;REPORT ANY "OPI" ERROR AS
; . OPI DUE TO "MOL" = 0
; . OPI BECAUSE ON CYLINDER LATCH DIDN'T RESET
;
; BIT #OPI,RMER11 ;WAS OPI SET??
; BEQ 31$ ;NO!!
; MOV RMER11,$GDDAT ;EXPECTED STATUS
; BIC #OPI,$GDDAT
; MOV RMER11,$BDDAT ;RECEIVED STATUS
; ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
; MOVB #72,@(SP) ;WRITE ERROR NUMBER IN USER'S CALL
; BIT #MOL,RMDSI ;WAS "MOL" = 0??
; BEQ 3$ ;YES!!
; MOVB #73,@(SP) ;NO - CHANGE ERROR NUMBER
; SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
; JSR PC,@(SP)+ ;REPORT ERROR VIA USER
; SUB #10,(SP) ;MOVE SP BACK TO BRANCH
; NOP

3$:

31$:

;REPORT AN ERROR IF "IAE" IS SET
;
; BIT #IAE,RMER11 ;IS "IAE" SET??
; BEQ 4$ ;NO!!
; MOV RMER11,$GDDAT ;EXPECTED STATUS
; BIC #IAE,$GDDAT
; MOV RMER11,$BDDAT ;RECEIVED STATUS
; ADD #4,(SP) ;MOVE SP TO ERROR CALL
; MOVB #70,@(SP) ;WRITE ERROR NUMBER IN USER'S CALL
; SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
; JSR PC,@(SP)+ ;REPORT ERROR
; SUB #10,(SP) ;MOVE SP BACK TO NO ERROR RETURN
; NOP

4$:

;SEE IF "SKI" OR "IVC" OR "DVC" IS SET
;
; BIT #SKI!IVC!DVC,RMER2I
; BEQ 8$ ;NONE OF THE BITS ARE SET

8$:

;REPORT ANY "IVC" ERROR AS
; . IVC WITH VV = 0
; . ERRONEOUS IVC ERROR
;
; BIT #IVC,RMER2I ;WAS IVC SET??
; BEQ 6$ ;NO!!
; MOV RMER2I,$GDDAT ;EXPECTED STATUS
; BIC #IVC,$GDDAT
; MOV RMER2I,$BDDAT ;RECEIVED STATUS
; ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
; MOVB #74,@(SP) ;WRITE ERROR NUMBER IN CALL
; BIT #VV,RMDSI ;WAS VV = 0??
; BEQ 5$ ;YES!!
; MOVB #75,@(SP) ;NO - CHANGE ERROR NUMBER
; SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
; JSR PC,@(SP)+ ;REPORT ERROR VIA USER
; SUB #10,(SP) ;MOVE SP BACK TO BRANCH
; NOP

5$:

```

```

11843 055152          6$:
11844
11845          ;REPORT ANY "SKI" ERROR
11846 055152 032737 040000 001372          BIT      #SKI,RMER2I      ;WAS SKI SET??
11847 055152 001424          BEQ      7$              ;NO!!
11848 055152 013737 001372 001140          MOV      RMER2I,$GDDAT  ;EXPECTED STATUS
11849 055152 042737 040000 001140          BIC      #SKI,$GDDAT
11850 055176 013737 001372 001142          MOV      RMER2I,$BDDAT  ;RECEIVED STATUS
11851 055204 062716 000004          ADD      #4,(SP)        ;MOVE SP TO USER'S ERROR CALL
11852 055210 112776 000076 000000          MOVB    #76,2(SP)      ;WRITE ERROR NUMBER
11853 055216 162716 000002          SUB      #2,(SP)        ;MOVE SP TO RETURN FOR ERROR
11854 055222 004736          JSR      PC,2(SP)+      ;REPORT ERROR VIA USER
11855 055224 162716 000010          SUB      #10,(SP)      ;MOVE SP TO BRANCH
11856 055230 000240          NOP
11857 055232          7$:
11858
11859          ;REPORT ANY "DVC" ERROR
11860 055232 032737 000200 001372          BIT      #DVC,RMER2I    ;WAS "DVC" SET??
11861 055240 001424          BEQ      8$              ;NO!!
11862 055242 013737 001372 001140          MOV      RMER2I,$GDDAT  ;EXPECTED STATUS
11863 055250 042737 000200 001140          BIC      #DVC,$GDDAT
11864 055256 013737 001372 001142          MOV      RMER2I,$BDDAT  ;RECEIVED STATUS
11865 055264 062716 000004          ADD      #4,(SP)        ;MOVE SP TO USER'S ERROR CALL
11866 055270 112776 000077 000000          MOVB    #77,2(SP)      ;WRITE ERROR NUMBER
11867 055276 162716 000002          SUB      #2,(SP)        ;MOVE SP TO RETURN FOR ERROR
11868 055302 004736          JSR      PC,2(SP)+      ;REPORT ERROR VIA USER
11869 055304 162716 000010          SUB      #10,(SP)      ;MOVE SP TO USER'S BRANCH
11870 055310 000240          NOP
11871 055312          8$:
11872
11873          ;SEE IF "PIP" AND "OM" ARE 0, AND "ATA" "MOL" AND "VV" ARE 1
11874 055312 013746 001342          MOV      RMDSI, -(SP)    ;PUT RMDS ON STACK
11875 055316 042716 047676          BIC      #1<PIP!MOL!VV!OM!ATA>,(SP)
11876 055322 022726 110100          CMP      #ATA!MOL!VV,(SP)+
11877 055326 001002          BNE     85$
11878 055330 000137 055744          JMP     13$
11879 055334          85$:
11880
11881          ;REPORT AN ERROR IF MOL = 0 AND OPI = 0, I.E., MEDIUM WENT OFF
11882          ;LINE AFTER RECALIBRATE WAS INITIATED
11883 055334 032737 010000 001342          BIT      #MOL,RMDSI     ;DID MOL DROP??
11884 055342 001030          BNE     9$              ;NO!!
11885 055344 032737 020000 001344          BIT      #OPI,RMER1I    ;WAS OPI ERROR REPORTED??
11886 055352 001024          BNE     9$              ;YES - DON'T REPORT MOL=0
11887 055354 013737 001342 001140          MOV      RMDSI,$GDDAT  ;EXPECTED STATUS
11888 055362 052737 010000 001140          BIS      #MOL,$GDDAT
11889 055370 013737 001342 001142          MOV      RMDSI,$BDDAT  ;RECEIVED STATUS
11890 055376 062716 000004          ADD      #4,(SP)        ;MOVE SP TO USER'S ERROR CALL
11891 055402 112776 000100 000000          MOVB    #100,2(SP)     ;WRITE ERROR NUMBER
11892 055410 162716 000002          SUB      #2,(SP)        ;MOVE SP TO RETURN FOR ERROR
11893 055414 004736          JSR      PC,2(SP)+      ;REPORT ERROR VIA USER
11894 055416 162716 000010          SUB      #10,(SP)      ;MOVE SP BACK TO USER'S BRANCH
11895 055422 000240          NOP
11896 055424          9$:
11897
11898          ;REPORT AN ERROR IF "VV" = 0 AND "IVC" = 0

```


E03

CZRMEBO RM03/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 237
RECALIBRATE STATUS CHECK SUBROUTINE

SEQ 0237

11899	055424	032737	000100	001342	BIT	#VV,RMDSI	;DID "VV" DROP??
11900	055432	001030			BNE	10\$;NO!!
11901	055434	032737	010000	001372	BIT	#IVC,RMER2I	;WAS THERE A IVC ERROR??
11902	055442	001024			BNE	10\$;YES - DONT REPORT VV=0
11903	055444	013737	001342	001140	MOV	RMDSI,\$GDDAT	;EXPECTED STATUS
11904	055452	013737	001342	001142	MOV	RMDSI,\$BDDAT	;RECEIVED STATUS
11905	055460	052737	000100	001140	BIS	#VV,\$GDDAT	
11906	055466	062716	000004		ADD	#4,(SP)	;MOVE SP TO USER'S ERROR CALL
11907	055472	112776	000101	000000	MOVB	#101,2(SP)	;WRITE ERROR NUMBER IN CALL
11908	055500	162716	000002		SUB	#2,(SP)	;MOVE SP TO RETURN FOR ERROR
11909	055504	004736			JSR	PC,2(SP)+	
11910	055506	162716	000010		SUB	#10,(SP)	;MOVE SP BACK TO USER'S BRANCH
11911	055512	000240			NOP		
11912	055514						
11913							
11914							
11915	055514	032737	100000	001342			;REPORT AN ERROR IF ATA IS NOT SET
11916	055522	001024			BIT	#ATA,RMDSI	;WAS ATA SET DURING RECALIBRATE??
11917	055524	013737	001342	001140	BNE	11\$;YES!!
11918	055532	052737	100000	001140	MOV	RMDSI,\$GDDAT	;EXPECTED STATUS
11919	055540	013737	001342	001142	BIS	#ATA,\$GDDAT	
11920	055546	062716	000004		MOV	RMDSI,\$BDDAT	;RECEIVED STATUS
11921	055552	112776	000102	000000	ADD	#4,(SP)	;MOVE SP TO USER'S ERROR CALL
11922	055560	162716	000002		MOVB	#102,2(SP)	;WRITE ERROR NUMBER IN CALL
11923	055564	004736			SUB	#2,(SP)	
11924	055566	162716	000010		JSR	PC,2(SP)+	
11925	055572	000240			SUB	#10,(SP)	;MOVE SP TO USER'S BRANCH
11926					NOP		
11927	055574						
11928							
11929							
11930							
11931	055574	032737	000001	001342			;REPORT AN ERROR IF "OM" IS NOT ZERO BECAUSE RECALIBRATE SHOULD
11932	055602	001424			;ALWAYS		CLEAR OFFSET MODE
11933	055604	013737	001342	001140	BIT	#OM,RMDSI	;WAS "OM" RESET??
11934	055612	042737	000001	001140	BEQ	12\$;YES!!
11935	055620	013737	001342	001142	MOV	RMDSI,\$GDDAT	;EXPECTED STATUS
11936	055626	062716	000004		BIC	#OM,\$GDDAT	
11937	055632	112776	000103	000000	MOV	RMDSI,\$BDDAT	;RECEIVED STATUS
11938	055640	162716	000002		ADD	#4,(SP)	;MOVE SP TO USER'S ERROR CALL
11939	055644	004736			MOVB	#103,2(SP)	;WRITE ERROR NUMBER
11940	055646	162716	000010		SUB	#2,(SP)	;MOVE SP TO RETURN FOR ERROR
11941	055652	000240			JSR	PC,2(SP)+	;REPORT ERROR VIA USER
11942	055654				SUB	#10,(SP)	;MOVE SP TO USER'S BRANCH
11943					NOP		
11944							
11945							
11946	055654	032737	020000	001342			;REPORT AN ERROR IF "PIP" IS STIL ON, I.E., DRIVE NOT ON
11947	055662	001430			;CYLINDER		
11948	055664	032737	040000	001372	BIT	#PIP,RMDSI	;IS DRIVE OFF CYLINDER??
11949	055672	001024			BEQ	13\$;NO!!
11950	055674	013737	001342	001140	BIT	#SKI,RMER2I	;WAS "SKI" DETECTED??
11951	055702	042737	020000	001140	BNE	13\$;YES-DONT REPORT "PIP"
11952	055710	013737	001342	001142	MOV	RMDSI,\$GDDAT	;EXPECTED STATUS
11953	055716	062716	000004		BIC	#PIP,\$GDDAT	
11954	055722	112776	000104	000000	MOV	RMDSI,\$BDDAT	;RECEIVED STATUS
					ADD	#4,(SP)	;MOVE SP TO USER'S ERROR CALL
					MOVB	#104,2(SP)	;WRITE ERROR NUMBER

```

11955 055730 162716 000002          SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
11956 055734 004736                JSR      PC,@(SP)+
11957 055736 162716 000010          SUB      #10,(SP)         ;MOVE SP BACK TO USER'S BRANCH
11958 055742 000240                NOP
11959 055744
11960
11961                                13$:
11962 055744 032737 040006 001344      ;SEE IF "ILR" OR "RMR" OR "UNS" IS SET
11963 055752 001514                BIT      #ILR!RMR!UNS,RMER1I
11964                                BEQ      16$
11965                                ;REPORT AN ERROR IF "ILR" IS SET
11966 055754 032737 000002 001344      BIT      #ILR,RMER1I      ;WAS ILR SET DURING RECALIBRATE??
11967 055762 001424                BEQ      14$              ;NO!!
11968 055764 013737 001344 001140      MOV      RMER1I,$GDDAT    ;EXPECTED STATUS
11969 055772 042737 000002 001140      BIC      #ILR,$GDDAT
11970 056000 013737 001344 001142      MOV      RMER1I,$BDDAT    ;RECEIVED STATUS
11971 056006 062716 000004                ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR CALL
11972 056012 112776 000105 000000      MOV      #105,@(SP)      ;WRITE ERROR NUMBER IN CALL
11973 056020 162716 000002          SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
11974 056024 004736                JSR      PC,@(SP)+
11975 056026 162716 000010          SUB      #10,(SP)         ;MOVE SP TO USER'S BRANCH
11976 056032 000240                NOP
11977 056034
11978
11979                                14$:
11980 056034 032737 000004 001344      ;REPORT AN ERROR IF "RMR" IS SET
11981 056042 001424                BIT      #RMR,RMER1I      ;WAS RMR SET??
11982 056044 013737 001344 001140      BEQ      15$              ;NO!!
11983 056052 042737 000004 001140      MOV      RMER1I,$GDDAT    ;EXPECTED STATUS
11984 056060 013737 001344 001142      BIC      #RMR,$GDDAT
11985 056066 062716 000004                MOV      RMER1I,$BDDAT    ;RECEIVED STATUS
11986 056072 112776 000106 000000      ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR CALL
11987 056100 162716 000002          MOV      #106,@(SP)      ;WRITE ERROR NUMBER IN USER'S CALL
11988 056104 004736                SJB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
11989 056106 162716 000010          JSR      PC,@(SP)+        ;REPORT ERROR VIA USER
11990 056112 000240                SUB      #10,(SP)         ;MOVE SP TO USER'S BRANCH
11991 056114
11992
11993                                15$:
11994 056114 032737 040000 001344      ;REPORT AN ERROR IF "UNS" IS SET AND "DVC" IS 0
11995 056122 001430                BIT      #UNS,RMER1I      ;WAS UNSAFE ON??
11996 056124 032737 000200 001372      BEQ      16$              ;NO!!
11997 056132 001024                BIT      #DVC,RMER2I      ;WAS THERE A DEVICE CHECK??
11998 056134 013737 001344 001140      BNE      16$              ;YES - DON'T REPORT UNSAFE
11999 056142 042737 040000 001140      MOV      RMER1I,$GDDAT    ;EXPECTED STATUS
12000 056150 013737 001344 001142      BIC      #UNS,$GDDAT
12001 056156 062716 000004                MOV      RMER1I,$BDDAT    ;RECEIVED STATUS
12002 056162 112776 000107 000000      ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR CALL
12003 056170 162716 000002          MOV      #107,@(SP)      ;WRITE ERROR NUMBER
12004 056174 004736                SUB      #2,(SP)          ;MOVE SP TO RETURN FOR ERROR
12005 056176 162716 000010          JSR      PC,@(SP)+        ;REPORT ERROR VIA USER
12006 056202 000240                SUB      #10,(SP)         ;MOVE SP BACK TO USER'S BRANCH
12007 056204
12008                                16$:
12009
12010 056204 062716 000004          ;AUGMENT THE RETURN ADDRESS IF ANY ERROR WAS DETECTED
                                ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR CALL

```

G03

CZRMEBO RM03/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 239
RECALIBRATE STATUS CHECK SUBROUTINE

SEQ 0239

12011	056210	105776	000000	TSTB	2(SP)	; WAS AN ERROR REPORTED??
12012	056214	001403		BEQ	17\$; NO!!
12013	056216	062716	000004	ADD	4(SP)	; YES - AUGMENT SP RETURN
12014	056222	000402		BR	18\$	
12015	056224	162716	000004	17\$: SUB	4, (SP)	; NO ERROR - RETURN SP TO BRANCH
12016	056230	000240		18\$: NOP		
12017	056232	000207		RTS	PC	; STATUS CECK IS COMPLETE
12018						

```

12019 .SBTTL DRIVE CLEAR STATUS CHECK SUBROUTINE
12020 BR ??? RETURN HERE IF NO ERROR
12021 NOP RETURN HERE TO REPORT AN ERROR
12022 ERROR ERROR NUMBER DEFINED BY SUB
12023 JSR PC,2(SP)+ GO BACK TO SUB FOR MORE ERROR CHECKS
12024 ??? RETURN HERE IF NO MORE ERRORS
12025
12026 056234 DRVSTS:
12027
12028 ;CLEAR USER'S ERROR CALL
12029 056234 062716 000004 ADD #4,(SP) ;MOVE SP TO ERROR CALL
12030 056240 105076 000000 CLRB 2(SP) ;CLEAR ERROR CALL
12031 056244 162716 000004 SUB #4,(SP) ;MOVE SP TO USER'S BRANCH
12032
12033 056250 013737 001330 001142 ;REPORT ERROR IF RMCS1 NOT INITIALIZED
12034 056256 042737 173700 001142 4S: MOV RMCS1,$BDDAT ;CHECK RMCS1
12035 056264 012737 004010 001140 BIC #1C<DVA!FNCMSK>,$BDDAT ;CLEAR DONT CARES
12036 056272 023737 001140 001142 MOV #DVA!DRVCLR,$GDDAT ;EXPECT DVA
12037 056300 001443 BEQ $S ;COMPARE EXPECTED & RECEIVED
12038 056302 062716 000004 ADD #4,(SP) ;BRANCH IF EQUAL
12039 056306 112776 000141 000000 MOV #141,2(SP) ;MOVE SP TO ERROR CALL
12040 056314 162716 000002 SUB #2,(SP) ;WRITE NUMBER OF ERROR IN CALL
12041 056320 004736 JSR PC,2(SP)+ ;MOVE SP TO RETURN FOR ERROR
12042 056322 162716 000010 SUB #10,(SP) ;REPORT THE ERROR VIA USER
12043 056326 000240 NOP ;MOVE SP TO NO ERROR RETURN
12044
12045 056330 013737 001342 001142 ;REPORT ERROR IF RMDS NOT INITIALIZED
12046 056336 042737 021101 001142 5S: MOV RMDS,$BDDAT ;CHECK RMDS
12047 056344 012737 010600 001140 BIC #PGM!OM!VV!PIP,$BDDAT ;CLEAR DONT CARES
12048 056352 023737 001140 001142 MOV #MOL!DPR!DRY,$GDDAT ;EXPECT DRY & DPR
12049 056360 001413 BEQ $S ;COMPARE EXPECTED & RECLIVED
12050 056362 062716 000004 ADD #4,(SP) ;BRANCH IF EQUAL
12051 056366 112776 000142 000000 MOV #142,2(SP) ;MOVE SP TO ERROR CALL
12052 056374 162716 000002 SUB #2,(SP) ;WRITE NUMBER OF ERROR IN CALL
12053 056400 004736 JSR PC,2(SP)+ ;MOVE SP TO RETURN FOR ERROR
12054 056402 162716 000010 SUB #10,(SP) ;REPORT THE ERROR VIA USER
12055 056406 000240 NOP ;MOVE SP TO NO ERROR RETURN
12056
12057 056410 005037 001140 001142 ;REPORT ERROR IF RMER1 NOT INITIALIZED
12058 056414 013737 001344 001142 6S: CLR $GDDAT ;EXPECT 0'S
12059 056422 001413 BEQ $S ;CHECK RMER1
12060 056424 062716 000004 ADD #4,(SP) ;BRANCH IF EQUAL
12061 056430 112776 000143 000000 MOV #143,2(SP) ;MOVE SP TO ERROR CALL
12062 056436 162716 000002 SUB #2,(SP) ;WRITE NUMBER OF ERROR IN CALL
12063 056442 004736 JSR PC,2(SP)+ ;MOVE SP TO RETURN FOR ERROR
12064 056444 162716 000010 SUB #10,(SP) ;REPORT THE ERROR VIA USER
12065 056450 000240 NOP ;MOVE SP TO NO ERROR RETURN
12066
12067 056452 013737 001346 001142 ;REPORT ERROR IF ATA NOT INITIALIZED
12068 056460 010146 001142 001142 7S: MOV RMAS1,$BDDAT ;CHECK ATTENTION BIT
12069 056462 010246 MOV R1,-(SP) ;PUSH R1 ON STACK
12070 056464 013701 001450 MOV R2,-(SP) ;PUSH R2 ON STACK
12071 056470 116102 000001 MOV TSTQUE,R1
12072 056474 042702 177400 MOVB 1(R1),R2
12073 056500 005102 BIC #1CATNMSK,R2
12074 056502 040237 001142 COM R2
BIC R2,$BDDAT

```

```

12075 056506 012602          MOV      (SP)+,R2          ;; POP STACK INTO R2
12076 056510 012601          MOV      (SP)+,R1          ;; POP STACK INTO R1
12077 056512 005737 001142      TST      $BDDAT          ;; IS ATTENTION CLEARED??
12078 056516 001413          BEQ      9$              ;; BRANCH IF ATTENTION CLEARED
12079 056520 062716 000004      ADD      #4,(SP)          ;; MOVE SP TO ERROR CALL
12080 056524 112776 000144 000000      MOVVB   #14,2(SP)        ;; WRITE NUMBER OF ERROR IN CALL
12081 056532 162716 000002          SUB      #2,(SP)          ;; MOVE SP TO RETURN FOR ERROR
12082 056536 004736          JSR      PC,2(SP)+        ;; REPORT THE ERROR VIA USER
12083 056540 162716 000010          SUB      #10,(SP)        ;; MOVE SP TO NO ERROR RETURN
12084 056544 000240          NOP
12085          ;REPORT ERROR IF RMMR1 NOT INITIALIZED
12086 056546 013737 001354 001142 9$: MOV      RMMR1,$BDDAT      ;; CHECK RMMR
12087 056554 042737 000046 001142      BIC      #WC!LS!LST,$BDDAT ;; CLEAR DONT CARES
12088 056562 012737 000010 001140      MOV      #MWD,$GDDAT      ;; EXPECT WRITE DATA ON
12089 056570 023737 001140 001142      CMP      $GDDAT,$BDDAT    ;; COMPARE EXPECTED AND RECEIVED
12090 056576 001413          BEQ      11$             ;; BRANCH IF ZERO
12091 056600 062716 000004          ADD      #4,(SP)          ;; MOVE SP TO ERROR CALL
12092 056604 112776 000145 000000      MOVVB   #145,2(SP)        ;; WRITE NUMBER OF ERROR IN CALL
12093 056612 162716 000002          SUB      #2,(SP)          ;; MOVE SP TO RETURN FOR ERROR
12094 056616 004736          JSR      PC,2(SP)+        ;; REPORT THE ERROR VIA USER
12095 056620 162716 000010          SUB      #10,(SP)        ;; MOVE SP TO NO ERROR RETURN
12096 056624 000240          NOP
12097          ;REPORT ERROR IF RMMR2 NOT INITIALIZED
12098 056626 013737 001370 001142 11$: MOV      RMMR2,$BDDAT      ;; CHECK RMMR2
12099 056634 042737 140000 001142      BIC      #RQA!ROB,$BDDAT   ;; CLEAR RQA, ROQB
12100 056642 012737 011777 001140      MOV      #TST!1777,$GDDAT ;; EXPECT TEST BIT ON
12101 056650 023737 001140 001142      CMP      $GDDAT,$BDDAT    ;; COMPARE EXPECTED & RECEIVED
12102 056656 001413          BEQ      15$             ;; BRANCH IF EQUAL
12103 056660 062716 000004          ADD      #4,(SP)          ;; MOVE SP TO ERROR CALL
12104 056664 112776 000146 000000      MOVVB   #146,2(SP)        ;; WRITE NUMBER OF ERROR IN CALL
12105 056672 162716 000002          SUB      #2,(SP)          ;; MOVE SP TO RETURN FOR ERROR
12106 056676 004736          JSR      PC,2(SP)+        ;; REPORT THE ERROR VIA USER
12107 056700 162716 000010          SUB      #10,(SP)        ;; MOVE SP TO NO ERROR RETURN
12108 056704 000240          NOP
12109 056706 005037 001140          15$: CLR      $GDDAT          ;; EXPECT ZEROS
12110          ;REPORT ERROR IF RMEC2 NOT RESET
12111 056712 013737 001376 001142 ;REPORT MOV      RMEC2I,$BDDAT ;; CHECK RMEC2
12112 056720 001413          BEQ      17$             ;; BRANCH IF 0
12113 056722 062716 000004          ADD      #4,(SP)          ;; MOVE SP TO ERROR CALL
12114 056726 112776 000150 000000      MOVVB   #150,2(SP)        ;; WRITE NUMBER OF ERROR IN CALL
12115 056734 162716 000002          SUB      #2,(SP)          ;; MOVE SP TO RETURN FOR ERROR
12116 056740 004736          JSR      PC,2(SP)+        ;; REPORT THE ERROR VIA USER
12117 056742 162716 000010          SUB      #10,(SP)        ;; MOVE SP TO NO ERROR RETURN
12118 056746 000240          NOP
12119          ;REPORT ERROR IF RMER2 NOT RESET
12120 056750 013737 001372 001142 17$: MOV      RMER2I,$BDDAT ;; CHECK RMER2
12121 056756 001413          BEQ      18$             ;; BRANCH IF NO ERROR
12122 056760 062716 000004          ADD      #4,(SP)          ;; MOVE SP TO ERROR CALL
12123 056764 112776 000147 000000      MOVVB   #147,2(SP)        ;; WRITE NUMBER OF ERROR IN CALL
12124 056772 162716 000002          SUB      #2,(SP)          ;; MOVE SP TO RETURN FOR ERROR
12125 056776 004736          JSR      PC,2(SP)+        ;; REPORT THE ERROR VIA USER
12126 057000 162716 000010          SUB      #10,(SP)        ;; MOVE SP TO NO ERROR RETURN
12127 057004 000240          NOP
12128 057006          18$:
12129          19$:
12130 057006

```

J03

CZRMEBO RMO3 2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 242
DRIVE CLEAR STATUS CHECK SUBROUTINE

SEQ 0242

12131			
12132			
12133	057006	062716	000004
12134	057012	105776	000000
12135	057016	001403	
12136	057020	062716	000004
12137	057024	000402	
12138	057026	162716	000004
12139	057032	000240	
12140	057034	000207	

```
; AUGMENT RETURN ADDRESS IF ANY ERROR WAS FOUND
      ADD      #4, (SP)      ; MOVE SP TO ERROR CALL
      TSTB    @ (SP)        ; WAS AN ERROR DETECTED??
      BEQ     21$,          ; NO!!
      ADD     #4, (SP)        ; YES - MOVE SP 0 ERROR RETURN
      BR      23$,
21$:  SUB     #4, (SP) ; MOVE SP BACK TO NO ERROR RETURN
23$:  NOP
      RTS      PC           ; RETURN TO USER
```

K03

CZRMEBO RM03/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 243
DATA TRANSFER COMMAND STATUS CHECK SUBROUTINE

SEQ 0243

```

12141
12142
12143
12144
12145
12146
12147
12148
12149
12150
12151
12152
12153
12154
12155
12156
12157 057036
12158
12159
12160 057036 062716 000004
12161 057042 105076 000000
12162 057046 162716 000004
12163 057052 005037 062432
12164
12165
12166 057056 032737 020000 001330
12167 057064 001422
12168 057066 013737 001330 001140
12169 057074 042737 020000 001140
12170 057102 013737 001330 001142
12171 057110 062716 000004
12172 057114 112776 000013 000000
12173 057122 162716 000002
12174 057126 004736
12175 057130 000466
12176 057132
12177
12178
12179
12180 057132 032737 000010 001344
12181 057140 001435
12182 057142 032737 000010 001372
12183 057150 001031
12184 057152 013737 001344 001140
12185 057160 042737 000010 001140
12186 057166 013737 001344 001142
12187 057174 062716 000004
12188 057200 112776 000050 000000
12189 057206 032737 001000 001330
12190 057214 001003
12191 057216 112776 000274 000000
12192 057224 162716 000002
12193 057230 004736
12194 057232 000425
12195
12196 057234

```

```

.SBTTL DATA TRANSFER COMMAND STATUS CHECK SUBROUTINE
; THIS SUBROUTINE VERIFIES THE RESULTS OF ALL DATA TRANSFER COMMANDS
; USING STATUS STORED IN THE GET BUFFER AND TEST PARAMETERS
; STORED IN THE PUT BUFFER. ERRORS ARE REPORTED BY WRITING
; THE ERROR NUMBER IN THE USER'S ERROR CALL.
; USER'S SUBROUTINE CALL:
; (1) JSR PC,DTASTS
; (2) BR ??
; (3) NOP
; (4) ERROR
; (5) JSR PC,@(SP)+
; (6) ??
; RETURN HERE IF NO DATA ERRORS
; RETURN HERE TO REPORT AN ERROR
; SUB WRITES ERROR NUMBER
; USER RETURNS FOR MORE CHECKS
; SUB RETURNS HERE AFTER ALL
; ERRORS ARE REPORTED
DTASTS:
; CLEAR USER'S ERROR CALL AND ERROR FLAGS
ADD #4,(SP) ; MOVE SP TO USER'S ERROR
CLRB @(SP) ; CLEAR LOW ORDER BYTE OF TRAP
SUB #4,(SP) ; RESTORE SP TO NO ERROR
CLR 500$ ; CLEAR ERROR FLAGS
; REPORT ANY CONTROL BUS PARITY ERROR WHILE READING REMOTE REGISTERS, I.E.. MCPE = 1
BIT #MCPE,RMCS1I ; WAS THERE A PARITY ERROR??
BEQ 10$ ; NO!!
MOV RMCS1I,$GDDAT ; EXPECTED STATUS
BIC #MCPE,$GDDAT
MOV RMCS1I,$BDDAT ; RECEIVED STATUS
ADD #4,(SP) ; MOVE SP TO USER'S ERROR CALL
MOVB #13,@(SP) ; WRITE ERROR NUMBER
SUB #2,(SP) ; MOVE SP TO RETURN IF ERROR
JSR PC,@(SP)+ ; REPORT ERROR AND RETURN
BR 30$
10$:
; REPORT ANY CONTROL BUS PARITY ERROR WHILE WRING REMOTE REGISTERS, I.E.,
; PAR=1 AND DPE = 0
BIT #PAR,RMER1I ; WAS THERE A PARITY ERROR??
BEQ 20$ ; NO!!
BIT #DPE,RMER2I ; WAS IT DUE TO CONTROL BUS??
BNE 20$ ; NO!!
MOV RMER1I,$GDDAT ; EXPECTED STATUS
BIC #PAR,$GDDAT
MOV RMER1I,$BDDAT ; RECEIVED STATUS
ADD #4,(SP) ; MOVE SP TO USER'S ERROR
MOVB #50,@(SP) ; WRITE ERROR NUMBER
BIT #MXF,RMCS1I ; DID MXF GET SET??
BNE 15$ ; YES!!
MOVB #274,@(SP) ; NO - CHANGE ERROR NUMBER
SUB #2,(SP) ; MOVE SP TO RETURN IF ERROR
JSR PC,@(SP)+ ; REPORT ERROR AND RETURN
BR 30$
15$:
20$:

```

```

12197
12198
12199
12200
12201
12202
12203 057234 032737 001000 001340
12204 057242 001425
12205 057244 013737 001340 001140
12206 057252 042737 001000 001140
12207 057260 013737 001340 001142
12208 057266 062716 000004
12209 057272 112776 000275 000000
12210 057300 162716 000002
12211 057304 004736
12212 057306
12213
12214
12215 057306 162716 000010
12216 057312 000137 062404
12217
12218 057316
12219
12220
12221
12222
12223 057316 032737 020000 001344
12224 057324 001447
12225 057326 013737 001344 001140
12226 057334 042737 020000 001140
12227 057342 013737 001344 001142
12228 057350 032737 010000 001342
12229 057356 001404
12230 057360 032737 000100 001342
12231 057366 001013
12232 057370 062716 000004
12233 057374 112776 000276 000000
12234 057402 162716 000002
12235 057406 004736
12236 057410 162716 000010
12237 057414 000413
12238 057416
12239
12240
12241
12242 057416 062716 000004
12243 057422 112776 000277 000000
12244 057430 162716 000002
12245 057434 004736
12246 057436 162716 000010
12247 057442 000240
12248 057444
12249
12250
12251 057444 032737 010000 001372
12252 057452 001432

```

```

;LOOK FOR ANY ERRORS WHICH MAY HAVE OCCURRED DURING COMMAND INITIATION OR
;MECHANICAL POSITIONING

;FIRST TEST MXF WHICH WOULD INDICATE COMPOSITE ERROR SET WHEN FUNCTION
;CODE AND GO BIT WERE LOADED
      BIT      #MXF,RMCS2I      ;WAS "MISSED TRANSFER" SET??
      BEQ      40$              ;NO!!
      MOV      RMCS2I,$GDDAT    ;EXPECTED STATUS
      BIC      #MXF,$GDDAT
      MOV      RMCS2I,$BDDAT    ;RECEIVED STATUS
      ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR CALL
      MOVB     #275,@(SP)       ;WRITE ERROR NUMBER
      SUB      #2,(SP)          ;MOVE SP TO RETURN IF ERROR
      JSR      PC,@(SP)+        ;REPORT ERROR AND RETURN

30$:
;RESTORE SP TO NO ERROR RETURN AND BYPASS FURHTER STATUS CHECKING
      SUB      #10,(SP)         ;MOVE SP TO NO ERROR
      JMP      380$            ;SKIP TO END OF SUB

40$:
;REPORT AN ERROR IF "OPI" ERROR OCCURRED DUE TO "MOL" = 0, OR IF "OPI"
;AND "MOL" ARE SET, BUT "VV" IS RESET, INDICATING AN INTERMITTENT
;"MOL"
      BIT      #OPI,RMER1I      ;IS "OPI" SET??
      BEQ      60$              ;NO!!
      MOV      RMER1I,$GDDAT    ;EXPECTED STATUS
      BIC      #OPI,$GDDAT
      MOV      RMER1I,$BDDAT    ;RECEIVED STATUS
      BIT      #MOL,RMDSI      ;WAS MEDIUM OFF LINE??
      BEQ      45$              ;YES!!
      BIT      #VV,RMDSI        ;WAS "MOL" INTERMITTENT??
      BNE      50$              ;NO!!
      ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR CALL
      MOVB     #276,@(SP)       ;WRITE ERROR NUMBER IN CALL
      SUB      #2,(SP)          ;MOVE SP TO RETURN IF ERROR
      JSR      PC,@(SP)+        ;REPORT ERROR AND RETURN
      SUB      #10,(SP)         ;RESTORE SP TO NO ERROR
      BR       60$

50$:
;REPORT "OPI" ERROR, WHICH IS DUE TO "ON CYLINDER" NOT DROPPING OR
;"RUN" TIMEOUT (20 MS) OR SEARCH TIMEOUT (50 MS)
      ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR CALL
      MOVB     #277,@(SP)       ;WRITE ERROR NUMBER IN CALL
      SUB      #2,(SP)          ;MOVE SP TO RETURN IF ERROR
      JSR      PC,@(SP)+        ;REPORT ERROR AND RETURN
      SUB      #10,(SP)         ;RESTORE SP TO NO ERROR
      NOP

60$:
;LOOK FOR "IVC" ERROR DURING COMMAND INITIATION
      BIT      #IVC,RMER2I      ;WAS THERE AN "IVC" ERROR??
      BEQ      70$              ;NO!!

```


M03

CZRMEBO RM03/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 245
DATA TRANSFER COMMAND STATUS CHECK SUBROUTINE

SEQ 0245

```

12253 ;REPORT "IVC" ERROR DUE TO "VV" = 0, OR REPORT ERRONEOUS "IVC" ERROR
12254 057454 013737 001372 001140 MOV RMER2I,$GDDAT ;EXPECTED STATUS
12255 057462 042737 010000 001140 BIC #IVC,$GDDAT
12256 057470 013737 001372 001142 MOV RMER2I,$BDDAT ;RECEIVED STATUS
12257 057476 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR
12258 057502 112776 000300 000000 MOVB #300,2(SP) ;WRITE ERROR NUMBER IN CALL
12259 057510 032737 000100 001342 BIT #VV,RMSDI ;WAS VOLUME VALID??
12260 057516 001403 BEQ 65$ ;NO!!
12261 057520 112776 000301 000000 MOVB #301,2(SP) ;CHANGE ERROR NUMBER
12262 057526 162716 000002 65$: SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
12263 057532 004736 JSR PC,2(SP)+ ;REPORT "IVC" ERROR AND RETURN
12264 057534 162716 000010 SUB #10,(SP) ;RESTORE SP TO NO ERROR
12265 057540 70$:
12266
12267 ;SEE IF "ILF" OR "RMR" IS SET
12268 057540 032737 000007 001344 BIT #ILF!ILF!RMR,RMER1I
12269 057546 001510 BEQ 100$ ;NO ERRORS DETECTED
12270
12271 ;REPORT AN ERROR IF "ILR" IS SET
12272 057550 032737 000002 001344 BIT #ILR,RMER1I ;WAS "ILR" DETECTED??
12273 057556 001424 BEQ 80$ ;NO!!
12274 057560 013737 001344 001140 MOV RMER1I,$GDDAT ;EXPECTED STATUS
12275 057566 042737 000002 001140 BIC #ILR,$GDDAT
12276 057574 013737 001344 001142 MOV RMER1I,$BDDAT ;RECEIVED STATUS
12277 057602 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
12278 057606 112776 000302 000000 MOVB #302,2(SP) ;WRITE ERROR NUMBER IN CALL
12279 057614 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
12280 057620 004736 JSR PC,2(SP)+ ;REPORT ERROR AND RETURN
12281 057622 162716 000010 SUB #10,(SP) ;RESTORE SP TO NO ERROR
12282 057630 80$:
12283
12284 ;REPORT AN ERROR IF "ILF" IS SET
12285 057630 032737 000001 001344 BIT #ILF,RMER1I ;WAS "ILF" DETECTED??
12286 057636 001424 BEQ 90$ ;NO!!
12287 057640 013737 001344 001140 MOV RMER1I,$GDDAT ;EXPECTED STATUS
12288 057646 042737 000001 001140 BIC #ILF,$GDDAT
12289 057654 013737 001344 001142 MOV RMER1I,$BDDAT ;RECEIVED STATUS
12290 057662 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
12291 057666 112776 000303 000000 MOVB #303,2(SP) ;WRITE ERROR NUMBER IN CALL
12292 057674 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
12293 057700 004736 JSR PC,2(SP)+ ;REPORT ERROR AND RETURN
12294 057702 162716 000010 SUB #10,(SP) ;RESTORE SP TO NO ERROR
12295 057706 000240 90$:
12296 057710
12297
12298 ;REPORT AN ERROR IF "RMR" IS SET
12299 057710 032737 000004 001344 BIT #RMR,RMER1I ;WAS "RMR" DETECTED??
12300 057716 001424 BEQ 100$ ;NO!!
12301 057720 013737 001344 001140 MOV RMER1I,$GDDAT ;EXPECTED STATUS
12302 057726 042737 000004 001140 BIC #RMR,$GDDAT
12303 057734 013737 001344 001142 MOV RMER1I,$BDDAT ;RECEIVED STATUS
12304 057742 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
12305 057746 112776 000304 000000 MOVB #304,2(SP) ;WRITE ERROR NUMBER IN CALL
12306 057754 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
12307 057762 004736 JSR PC,2(SP)+ ;REPORT ERROR AND RETURN
12308 057762 162716 000010 SUB #10,(SP) ;RESTORE SP TO NO ERROR

```



```

12365 060300 090240          NOP
12366 060302          150$:
12367
12368
12369 060302 032737 006200 C01372 ;LOOK FOR "LSC" OR "LBC" OR "DVC" IN ERROR REGISTER #2
12370 060310 001512          BIT      #LSC!LBC!DVC,RMER2I
12371          BEQ      180$          ;NO ERRORS SET
12372
12373 060312 032737 000200 001372 ;REPORT ANY DEVICE FAULT I.E., "DVC" = 1
12374 060320 001424          BIT      #DVC,RMER2I          ;IS "DVC" = 1??
12375 060322 013737 001372 001140 BEQ      160$          ;NO!!
12376 060330 042737 000200 001140 MOV      RMER2I,$GDDAT          ;EXPECTED STATUS
12377 060336 013737 001372 001142 BIC      #DVC,$GDDAT
12378 060344 062716 000004          MOV      RMER2I,$BDDAT          ;RECEIVED STATUS
12379 060350 112776 000310 000000 ADD      #4,(SP)          ;MOVE SP TO USERS ERROR
12380 060356 162716 000002          MOVVB   #310,2(SP)          ;WRITE ERROR NUMBER IN CALL
12381 060362 004736          SUB      #2,(SP)          ;MOVE SP TO RETURN IF ERROR
12382 060364 162716 000010          JSR     PC,2(SP)+          ;REPORT ERROR AND RETURN
12383 060370 000240          SUB      #10,(SP)          ;RESTORE SP TO NO ERROR
12384 060372          NOP
12385
12386          160$:
12387 060372 032737 002000 001372 ;REPORT LOSS OF BIT CLOCK I.E.; "LBC" = 1 IF "MOL" = 1
12388 060400 001430          BIT      #LBC,RMER2I          ;IS LBC SET??
12389 060402 032737 010000 001342 BEQ      170$          ;NO!!
12390 060410 001424          BIT      #MOL,RMDSI          ;WAS LBC ERROR BY MOL = 0
12391 060412 013737 001372 001140 BEQ      170$          ;YES!!
12392 060420 042737 002000 001140 MOV      RMER2I,$GDDAT          ;EXPECTED STATUS
12393 060426 013737 001372 001142 BIC      #LBC,$GDDAT
12394 060434 062716 000004          MOV      RMER2I,$BDDAT          ;RECEIVED STATUS
12395 060440 112776 000311 000000 ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR CALL
12396 060446 162716 000002          MOVVB   #311,2(SP)          ;WRITE ERROR NUMBER IN CALL
12397 060452 004736          SUB      #2,(SP)          ;MOVE SP TO RETURN IF ERROR
12398 060454 162716 000010          JSR     PC,2(SP)+          ;REPORT ERROR AND RETURN
12399 060460 000240          SUB      #10,(SP)          ;RESTORE SP TO NO ERROR
12400 060462          NOP
12401
12402          170$:
12403 060462 032737 004000 001372 ;REPORT LOS OF SYSTEM CLOCK, I.E. "LSC" = 1
12404 060470 001422          BIT      #LSC,RMER2I          ;IS "LSC" = 1??
12405 060472 013737 001372 001140 BEQ      180$          ;NO!!
12406 060500 042737 004000 001140 MOV      RMER2I,$GDDAT          ;EXPECTED STATUS
12407 060506 013737 001372 001142 BIC      #LSC,$GDDAT
12408 060514 062716 000004          MOV      RMER2I,$BDDAT          ;RECEIVED STATUS
12409 060520 112776 000312 000000 ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR CALL
12410 060526 004736          MOVVB   #312,2(SP)          ;WRITE ERROR NUMBER
12411 060530 162716 000010          JSR     PC,2(SP)+          ;REPORT ERROR AND RETURN
12412 060534 000240          SUB      #10,(SP)          ;RESTORE SP TO NO ERROR
12413 060536          NOP
12414
12415          180$:
12416 060536 032737 054000 001344 ;LOOK FOR "UNS" OR "DTE" OR "WLE" IN ERROR REGISTER #1
12417 060544 001527          BIT      #UNS!DTE!WLE,RMER1I
12418          BEQ      220$          ;NO BITS SET
12419 060546 032737 040000 001344 ;REPORT "UNS" ERROR IF "DVC" = 0
12420 060554 001427          BIT      #UNS,RMER1I          ;IS "UNS" SET??
          BEQ      190$          ;NO!!

```

DATA TRANSFER COMMAND STATUS CHECK SUBROUTINE

```

12421 060556 032737 000200 001372 BIT #DVC,RMER2I ;WAS "UNS" CAUSED BY "DVC"??
12422 060564 001023 BNE 190$ ;YES!!
12423 060566 013737 001344 001140 MOV RMER1I,$GDDAT ;EXPECTED STATUS
12424 060574 042737 040000 001140 BIC #UNS,$GDDAT
12425 060602 013737 001344 001142 MOV RMER1I,$BDDAT ;RECEIVED STATUS
12426 060610 062716 000004 ADD #4,(SP) ;MOVE SP TO USERS ERROR CALL
12427 060614 112776 000313 000000 MCVB #313,2(SP) ;WRITE ERROR NUMBER
12428 060622 162716 000002 SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
12429 060626 004736 JSR PC,2(SP)+ ;REPORT ERROR AND RETURN
12430 060630 162716 000010 SUB #10,(SP) ;RESTORE SP TO NO ERROR
12431 060634
12432
12433
12434 060634 032737 010000 001344 ;REPORT ANY DRIVE TIMING ERROR, I.E. "DTE" = 1
12435 060642 001423 BIT #DTE,RMER1I ;IS DTE SET??
12436 060644 013737 001344 001140 BEQ 200$ ;NO!!
12437 060652 042737 010000 001140 MOV RMER1I,$GDDAT ;EXPECTED STATUS
12438 060660 013737 001344 001142 BIC #DTE,$GDDAT
12439 060666 062716 000004 ADD #4,(SP) ;RECEIVED STATUS
12440 060672 112776 000314 000000 MOVB #314,2(SP) ;MOVE SP TO USER'S ERROR CALL
12441 060700 162716 000002 SUB #2,(SP) ;WRITE ERROR NUMBER IN CALL
12442 060704 004736 JSR PC,2(SP)+ ;MOVE SP TO RETURN IF ERROR
12443 060706 162716 000010 SUB #10,(SP) ;REPORT ERROR AND RETURN
12444 060712
12445
12446
12447 ;REPORT AN ERROR IF WRITE LOCK ERROR IS SET. SEE IF DRIVE IS NOT
12448 060712 032737 004000 001344 ;WRITE PROTECTED, OR IF FUNCTION WAS NOT A WRITE
12449 060720 001441 BIT #WLE,RMER1I ;WAS "WLE" SET??
12450 060722 013737 001344 001142 BEQ 220$ ;NO!!
12451 060730 013737 001344 001140 MOV RMER1I,$BDDAT ;RECEIVED STATUS
12452 060736 052737 004000 001140 MOV RMER1I,$GDDAT ;EXPECTED STATUS
12453 060744 062716 000004 BIS #WLE,$GDDAT
12454 060750 112776 000315 000000 ADD #4,(SP) ;MOVE SP TO USERS ERROR CALL
12455 060756 032737 004000 001342 MOVB #315,2(SP) ;WRITE ERROR NUMBER IN CALL
12456 060764 001404 BIT #WRL,RMDSI ;WAS DRIVE WRITE PROTECTED??
12457 060766 032737 000010 001400 BEQ 205$ ;NO!!
12458 060774 001406 BIT #BIT3,RMCS10 ;WAS COMMAND A WRITE??
12459 060776 112776 000316 000000 BEQ 210$ ;YES!!
12460 061004 042737 004000 001140 MOVB #316,2(SP) ;CHANGE ERROR NUMBER
12461 061012 162716 000002 BIC #WLE,$GDDAT
12462 061016 004736 SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
12463 061020 162716 000010 JSR PC,2(SP)+ ;REPORT ERROR AND RETURN
12464
12465 061024 SUB #10,(SP) ;MOVE SP TO NO ERROR
12466
12467
12468 ;OMIT DATA ERROR CHECKS IF ANY PREVIOUS ERRORS HAVE BEEN DETECTED
12469 061024 062716 000004 ADD #4,(SP) ;MOVE SP TO USER'S ERROR
12470 061030 105776 000000 TSTB 2(SP) ;WAS ERROR DETECTED??
12471 061034 001404 BEQ 225$ ;NO - DO DATA CHECKS
12472 061036 162716 000004 SUB #4,(SP) ;RESTORE SP
12473 061042 000137 062044 JMP 340$ ;SKIP DATA CHECKS
12474 061046 162716 000004 SUB #4,(SP) ;RESTORE SP
12475
12476 ;CHECK HEADER ERRORS IF FUNCTION WAS NOT WRITE HEADER AND DATA, AND
;IF HEADER COMPARE IS NOT INHIBITED

```

```

12477 061052 013737 001400 062434      MOV      RMCS10,510$      ;STRIP AND STORE FUNCTION CODE
12478 061060 042737 177700 062434      BIC      #CFNCHSK,510$
12479 061066 022737 000063 062434      CMP      #WH!GO,510$      ;WAS FUNCTION WRITE HEADER & DATA??
12480 061074 001512                BEQ      250$              ;YES - SKIP HEADER CHECKS
12481 061076 032737 002000 001362      BIT      #HCI,RMOFI      ;WAS HCI SET??
12482 061104 001106                BNE      250$              ;YES - SKIP HEADER CHECKS
12483
12484
12485 061106 032737 000620 001344      ;SEE IF ANY HEADER ERRORS ARE SET I.E., "FER" OR "HCRC" OR "HCE"
12486 061114 001533                BIT      #HCRC!FER!HCE,RMER11
12487
12488
12489 061116 032737 000400 001344      ;REPORT HEADER CRC ERROR IF SET
12490 061124 001422                BEQ      230$              ;WAS HCRC SET??
12491 061126 013737 001344 001140      MOV      RMER11,$GDDAT    ;NO!!
12492 061134 042737 000400 001140      BIC      #HCRC,$GDDAT    ;EXPECTED STATUS
12493 061142 013737 001344 001142      MOV      RMER11,$BDDAT    ;RECEIVED STATUS
12494 061150 062716 000004                ADD      #4,(SP)          ;MOVE SP TO USERS ERROR
12495 061154 112776 000317 000000      MOVVB   #317,2(SP)       ;WRITE ERROR NUMBER
12496 061162 162716 000002                SUB      #2,(SP)         ;MOVE SP TO RETURN IF ERROR
12497 061166 004736                JSR      PC,2(SP)+       ;REPORT ERROR AND RETURN
12498 061170 000501                BR       260$
12499 061172
12500
12501
12502 061172 032737 000020 001344      ;REPORT FORMAT ERROR IF SET
12503 061200 001422                BIT      #FER,RMER11      ;WAS "FER" SET??
12504 061202 013737 001344 001140      BEQ      240$              ;NO!!
12505 061210 042737 000020 001140      MOV      RMER11,$GDDAT    ;EXPECTED STATUS
12506 061216 013737 001344 001142      BIC      #FER,$GDDAT
12507 061224 062716 000004                MOV      RMER11,$BDDAT    ;RECEIVED STATUS
12508 061230 112776 000320 000000      ADD      #4,(SP)          ;MOVE SP TO USERS ERROR
12509 061236 162716 000002                MOVVB   #320,2(SP)       ;WRITE ERROR NUMBER
12510 061242 004736                SUB      #2,(SP)         ;MOVE SP TO RETURN IF ERROR
12511 061244 000453                JSR      PC,2(SP)+       ;REPORT ERROR AND RETURN
12512 061246
12513
12514
12515 061246 032737 000200 001344      ;REPORT HEADER COMPARE ERROR IF SET
12516 061254 001453                BIT      #HCE,RMER11      ;WAS "HCE" SET??
12517 061256 013737 001344 001140      BEQ      270$              ;NO!!
12518 061264 042737 000200 001140      MOV      RMER11,$GDDAT    ;EXPECTED STATUS
12519 061272 013737 001344 001142      BIC      #HCE,$GDDAT
12520 061300 062716 000004                MOV      RMER11,$BDDAT    ;RECEIVED STATUS
12521 061304 112776 000321 000000      ADD      #4,(SP)          ;MOVE SP TO USER'S ERROR
12522 061312 162716 000002                MOVVB   #321,2(SP)       ;WRITE ERROR NUMBER
12523 061316 004736                SUB      #2,(SP)         ;MOVE SP TO RETURN IF ERROR
12524 061320 000425                JSR      PC,2(SP)+       ;REPORT ERROR AND RETURN
12525
12526
12527
12528 061322 032737 000620 001344      ;THERE SHOULD BE NO HEADER ERRORS BECAUSE
12529 061330 001425                ;.COMMAND WAS WRITE HEADER AND DATA, OR
12530 061332 013737 001344 001140      ;.HEADER COMPARE INHIBIT WAS SET
12531 061340 042737 000620 001140      250$: BIT      #HCE!FER!HCRC,RMER11
12532 061346 013737 001344 001142      BEQ      270$              ;NO ERRORS WERE SET
12533
12534
12535 061332 013737 001344 001140      MOV      RMER11,$GDDAT    ;EXPECTED STATUS
12536 061340 042737 000620 001140      BIC      #HCE!FER!HCRC,$GDDAT
12537 061346 013737 001344 001142      MOV      RMER11,$BDDAT    ;RECEIVED STATUS

```

E04

CZRMEBO RM03/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 250
DATA TRANSFER COMMAND STATUS CHECK SUBROUTINE

SEQ 0250

12533	061354	062716	000004		ADD	#4,(SP)	; MOVE SP TO USER'S ERROR CALL
12534	061360	112776	000322	000000	MOVB	#322,2(SP)	; WRITE ERROR NUMBER
12535	061366	162716	000002		SUB	#2,(SP)	; MOVE SP TO RETURN IF ERROR
12536	061372	004736			JSR	PC,2(SP)+	; REPORT ERROR AND RETURN
12537	061374	162716	000010	260\$:	SUB	#10,(SP)	; MOVE SP TO NO ERROR
12538	061400	000137	062044		JMP	340\$; OMIT FURTHER DATA CHECKS
12539							
12540	061404						270\$:
12541							
12542							; IF COMMAND WAS A WRITE COMMAND, GO DO WRITE ERROR CHECKS, OTHERWISE
12543							; DO READ ERROR CHECKS
12544	061404	032737	000010	062434	BIT	#BIT3,510\$; WAS THIS A WRITE COMMAND?
12545	061412	001002			BNE	275\$; NO!!
12546	061414	000137	061632		JMP	310\$; GO DO WRITE STATUS CHECK
12547	061420						275\$:
12548							
12549							; REPORT DATA CHECK IF SET
12550	061420	032737	100000	001344	BIT	#DCK,RMER11	; DATA CHECK ERROR??
12551	061426	00450			BEQ	290\$; NO!!
12552	061430	013737	001344	001140	MOV	RMER11,\$GDDAT	; EXPECTED STATUS
12553	061436	042737	100000	001140	BIC	#DCK,\$GDDAT	
12554	061444	013737	001344	001142	MOV	RMER11,\$BDDAT	; RECEIVED STATUS
12555	061452	062716	000004		ADD	#4,(SP)	; MOVE SP TO USER'S ERROR
12556	061456	112776	000323	000000	MOVB	#323,2(SP)	; WRITE ERROR NUMBER
12557	061464	032737	004000	001362	BIT	#ECI,RMOFI	; WAS ECC CORRECTION DISABLED??
12558	061472	001021			BNE	280\$; YES!!
12559	061474	112776	000324	000000	MOVB	#324,2(SP)	; CHANGE TO RECOVERABLE ERROR
12560	061502	032737	000100	001344	BIT	#ECH,RMER11	; IS ERROR RECOVERABLE??
12561	061510	001007			BNE	276\$; NO !!
12562							; DO NOT REPORT RECOVERABLE ERROR IF READ COMMAND
12563	061512	032737	000020	062434	BIT	#BIT4,510\$; WAS THIS A READ COMMAND ??
12564	061520	001406			BEQ	280\$; NO !!
12565	061522	162716	000004		SUB	#4,(SP)	; RESTORE SP
12566	061526	000410			BR	290\$; SKIP ERROR - DATA WILL BE CORRECTED
12567	061530	112776	000325	000000	MCVB	#325,2(SP)	; CHANGE TO NON RECOVERABLE
12568	061536	162716	000002	276\$:	SUB	#2,(SP)	; MOVE SP TO RETURN IF ERROR
12569	061542	004736		280\$:	JSR	PC,2(SP)+	; REPORT ERROR AND RETURN
12570	061544	162716	000010		SUB	#10,(SP)	; RESTORE SP TO NO ERROR
12571							
12572	061550						290\$:
12573							
12574							; REPORT DATA BUS PARITY ERROR IF SET, I.E. MDPE = 1
12575	061550	032737	000400	001340	BIT	#MDPE,RMCS21	; PARITY ERROR SET??
12576	061556	001423			BEQ	300\$; NO!!
12577	061560	013737	001340	001140	MOV	RMCS21,\$GDDAT	; EXPECTED STATUS
12578	061566	042737	000400	001140	BIC	#MDPE,\$GDDAT	
12579	061574	013737	001340	001142	MOV	RMCS21,\$BDDAT	; RECEIVED STATUS
12580	061602	062716	000004		ADD	#4,(SP)	; MOVE SP TO USER'S ERROR
12581	061606	112776	000326	000000	MOVB	#326,2(SP)	; WRITE ERROR NUMBER
12582	061614	162716	000002		SUB	#2,(SP)	; MOVE SP TO RETURN IF ERROR
12583	061620	004736			JSR	PC,2(SP)+	; REPORT ERROR AND RETURN
12584	061622	162716	000010		SUB	#10,(SP)	; MOVE SP TO NO ERROR
12585	061626	000137	062044	300\$:	JMP	340\$; SKIP WRITE STATUS CHECK
12586							
12587	061632						310\$:
12588							

F04

CZRMEBO RM03/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 251
DATA TRANSFER COMMAND STATUS CHECK SUBROUTINE

SEG 0251

```

12589          ;TEST TO SEE THAT OFFSET MODE WAS RESET; REPORT ERROR IF "OM" = 1
12590 061632 032737 000001 001342          BIT      #OM,RMSI          ;IS OFFSET ON??
12591 061640 001423                SEQ      320$          ;NO
12592 061642 013737 001342 001140          MOV      RMSI,$GDDAT    ;EXPECTED STATUS
12593 061650 042737 000001 001140          BIC      #OM,$GDDAT
12594 061656 013737 001342 001142          MOV      RMSI,$BDDAT    ;RECEIVED STATUS
12595 061664 062716 000004                ADD      #4,(SP)        ;MOVE SP TO USER'S ERROR CALL
12596 061670 112776 000327 000000          MOVVB   #327,2(SP)     ;WRITE ERROR NUMBER IN CALL
12597 061676 162716 000002                SUB      #2,(SP)        ;MOVE SP TO RETURN IF ERROR
12598 061702 004736                JSR      PC,2(SP)+      ;REPORT ERROR AND RETURN
12599 061704 162716 000010                SUB      #10,(SP)      ;MOVE SP TO NO ERROR
12600 061710
12601
12602          ;TEST FOR DATA BUS PARITY ERROR; REPORT ERROR IF "DPE" = 1
12603 061710 032737 000010 001372          BIT      #DPE,RMER2I   ;DATA PARITY ERROR??
12604 061716 001423                BEQ      330$          ;NO!!
12605 061720 013737 001372 001140          MOV      RMER2I,$GDDAT ;EXPECTED STATUS
12606 061726 042737 000010 001140          BIC      #DPE,$GDDAT
12607 061734 013737 001372 001142          MOV      RMER2I,$BDDAT ;RECEIVED STATUS
12608 061742 062716 000004                ADD      #4,(SP)        ;MOVE SP TO USER'S ERROR CALL
12609 061746 112776 000330 000000          MOVVB   #330,2(SP)     ;WRITE ERROR NUMBER
12610 061754 162716 000002                SUB      #2,(SP)        ;MOVE SP TO RETURN IF ERROR
12611 061760 004736                JSR      PC,2(SP)+      ;REPORT ERROR AND RETURN
12612 061762 162716 000010                SUB      #10,(SP)      ;MOVE SP TO NO ERROR
12613 061766
12614
12615          ;TEST FOR WRITE CLOCK FAILURE; REPORT ERROR IF "WCF" = 1
12616 061766 032737 000040 001344          BIT      #WCF,RMER1I   ;IS "WCF" SET??
12617 061774 001423                BEQ      340$          ;NO!!
12618 061776 013737 001344 001140          MOV      RMER1I,$GDDAT ;EXPECTED STATUS
12619 062004 042737 000040 001140          BIC      #WCF,$GDDAT
12620 062012 013737 001344 001142          MOV      RMER1I,$BDDAT ;RECEIVED STATUS
12621 062020 062716 000004                ADD      #4,(SP)        ;MOVE SP TO USERS ERROR CALL
12622 062024 112776 000331 000000          MOVVB   #331,2(SP)     ;WRITE ERROR NUMBER
12623 062032 162716 000002                SUB      #2,(SP)        ;MOVE SP TO RETURN IF ERROR
12624 062036 004736                JSR      PC,2(SP)+      ;REPORT ERROR AND RETURN
12625 062040 162716 000010                SUB      #10,(SP)      ;MOVE SP TO NO ERROR
12626 062044
12627
12628          ;REPORT "DATA LATE" ERROR IF "DLT" = 1
12629 062044 032737 100000 001340          BIT      #DLT,RMCS2I   ;IS "DLT" SET??
12630 062052 001423                BEQ      350$          ;NO!!
12631 062054 013737 001340 001140          MOV      RMCS2I,$GDDAT ;EXPECTED STATUS
12632 062062 042737 100000 001140          BIC      #DLT,$GDDAT
12633 062070 013737 001340 001142          MOV      RMCS2I,$BDDAT ;RECEIVED STATUS
12634 062076 062716 000004                ADD      #4,(SP)        ;MOVE SP TO USERS ERROR CALL
12635 062102 112776 000332 000000          MOVVB   #332,2(SP)     ;WRITE ERROR NUMBER
12636 062110 162716 000002                SUB      #2,(SP)        ;MOVE SP TO RETURN IF ERROR
12637 062114 004736                JSR      PC,2(SP)+      ;REPORT ERROR AND RETURN
12638 062116 162716 000010                SUB      #10,(SP)      ;MOVE SP TO NO ERROR
12639 062122
12640
12641          ;LOOK FOR UNEXPECTED CHANGES IN DRIVE STATUS
12641 062122 013746 001342          MOV      RMSI,-(SP)    ;STACK DRIVE STATUS
12642 062126 042716 147677          BIC      #C<PIP!MOL!VV>,(SP) ;CLEAR DONT CARES
12643 062132 022726 010100          CMP      #MOL!VV,(SP)+ ;IS DRIVE STATUS OK??
12644 062136 001522                BEQ      380$          ;YES!!

```

```

12645
12646
12647
12648 062140 032737 020000 001342
12649 062146 001430
12650 062150 032737 040000 001372
12651 062156 001024
12652 062160 013737 001342 001140
12653 062166 042737 020000 001140
12654 062174 013737 001342 001142
12655 062202 062716 000004
12656 062206 112776 000333 000000
12657 062214 162716 000002
12658 062220 004736
12659 062222 162716 000010
12660 062226 000240
12661 062230
12662
12663
12664
12665 062230 032737 010000 001342
12666 062236 001027
12667 062240 032737 020000 001344
12668 062246 001023
12669 062250 013737 001342 001140
12670 062256 052737 010000 001140
12671 062264 013737 001342 001142
12672 062272 062716 000004
12673 062276 112776 000334 000000
12674 062304 162716 000002
12675 062310 004736
12676 062312 162716 000010
12677 062316
12678
12679
12680
12681 062316 032737 000100 001342
12682 062324 001027
12683 062326 032737 010000 001372
12684 062334 001033
12685 062336 013737 001342 001140
12686 062344 052737 000100 001140
12687 062352 013737 001342 001142
12688 062360 062716 000004
12689 062364 112776 000335 000000
12690 062372 162716 000002
12691 062376 004736
12692 062400 162716 000010
12693 062404
12694
12695
12696 062404 062716 000004
12697 062410 105776 000000
12698 062414 001403
12699 062416 062716 000004
:2700 062422 000402

;REPORT ERROR IF POSITIONING IN PROGRESS AND NO SEEK INCOMPLETE ERROR,
;I.E. PIP = 1 AND SKI = 0
BIT #PIP,RMDSI ;IS "PIP" SET??
BEQ 360$ ;NO!!
BIT #SKI,RMER2I ;WAS "SKI" ERROR REPORTED??
BNE 360$ ;YES-DONT REPORT PIP
MOV RMDSI,$GDDAT ;EXPECTED STATUS
BIC #PIP,$GDDAT
MOV RMDSI,$BDDAT ;RECEIVED STATUS
ADD #4,(SP) ;MOVE SP TO USERS ERROR CALL
MOVB #333,a(SP) ;WRITE ERROR NUMBER
SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
JSR PC,a(SP)+ ;REPORT ERROR AND RETURN
SUB #10,(SP) ;MOVE SP TO NO ERROR
NOP

360$:
;REPORT ERROR IF MEDIUM IS NOT ON LINE AND OPI ERROR WAS NOT
;REPORTED, I.E., MOL = OPI = 0
BIT #MOL,RMDSI ;IS MEDIUM ON LINE??
BNE 370$ ;YES!!
BIT #OPI,RMER1I ;WAS OPI ERROR REPORTED??
BNE 370$ ;YES!!
MOV RMDSI,$GDDAT ;EXPECTED STATUS
BIS #MOL,$GDDAT
MOV RMDSI,$BDDAT ;RECEIVED STATUS
ADD #4,(SP) ;MOVE SP TO USER'S ERROR
MOVB #334,a(SP) ;WRITE ERROR NUMBER
SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
JSR PC,a(SP)+ ;REPORT ERROR AND RETURN
SUB #10,(SP) ;MOVE SP TO NO ERROR

370$:
;REPORT ERROR IF VOLUME IS NOT VALID AND "IVC" ERROR WAS NOT
;REPORTED, I.E., VV = IVC = 0
BIT #VV,RMDSI ;IS VOLUME VALID??
BNE 380$ ;YES!!
BIT #IVC,RMER2I ;WAS IVC ERROR REPORTED??
BNE 390$ ;YES!!
MOV RMDSI,$GDDAT ;EXPECTED STATUS
BIS #VV,$GDDAT
MOV RMDSI,$BDDAT ;RECEIVED STATUS
ADD #4,(SP) ;MOVE SP TO USERS ERROR CALL
MOVB #335,a(SP) ;WRITE ERROR NUMBER
SUB #2,(SP) ;MOVE SP TO RETURN IF ERROR
JSR PC,a(SP)+ ;REPORT ERROR AND RETURN
SUB #10,(SP) ;MOVE SP TO NO ERROR

380$:
;AUGMENT THE RETURN ADDRESS IF ANY ERROR WAS FOUND
ADD #4,(SP) ;MOVE SP TO ERROR CALL
TSTB a(SP) ;ANY ERROR??
BEQ 390$ ;NO!!
ADD #4,(SP) ;YES - MOVE SP TO ERROR RETURN
BR 400$

```


H04

CZMEBO RM03.2 FCTNL TST 3
CZMEB.P11 23-NOV-77 12:23

MACY11 30(1045) 23-NOV-77 12:49 PAGE 253
DATA TRANSFER COMMAND STATUS CHECK SUBROUTINE

SEQ 0253

12701 062424 162716 000004
12702
12703 062430 000207
12704
12705 062432 000000
12706 062434 000000

390\$: SUB #4,(SP) ;MOVE SP TO NO ERROR RETURN
400\$: RTS PC ;RETURN TO USER
500\$: .WORD ;ERROR FLAGS
510\$: .WORD ;TEMPORARY STORAGE

12707
12708
12709
12710
12711
12712
12713
12714
12715
12716
12717
12718
12719
12720
12721
12722
12723
12724
12725
12726
12727
12728
12729
12730
12731
12732
12733
12734
12735
12736
12737
12738
12739
12740
12741
12742
12743
12744
12745
12746
12747
12748
12749
12750
12751
12752
12753
12754
12755
12756
12757

062436

062436	062716	000004	
062442	105076	000000	
062446	162716	000004	
062452	013746	001342	
062456	042716	147677	
062462	022726	010100	
062466	001524		
062470	032737	010000	001342
062476	001030		
062500	032737	020000	001344
062506	001024		
062510	013737	001342	001140
062516	052737	010000	001140
062524	013737	001342	001142
062532	062716	000004	
062536	112776	000207	000000
062544	162716	000002	
062550	004736		
062552	162716	000010	

```

.SBTTL  STATIC DRIVE STATUS CHECK SUBROUTINE
;THIS SUBROUTINE LOOKS FOR UNEXPECTED CHANGES IN DRIVE
;STATUS, SUCH AS THE DRIVE LOSING VOLUME VALID.  THE SUBROUTINE
;CAN BE USED BY HOUSEKEEPING AND OTHER COMMANDS DURING WHICH THERE
;SHOULD NOT BE ANY DRIVE ERRORS OR CHANGES IN STATE.
;THE FOLLOWING CONDITIONS ARE TESTED AND REPORTED AS ERRORS
;IF TRUE:
;
; .MOL = 0, INDICATES DRIVE WENT OFFLINE, NOTE
; THAT MOL IS ASSUMED TO HAVE BEEN SET
; .VV = 0, INDICATES THE DRIVE LOST VOLUME VALID
; .PIP = 1, INDICATES THAT THE DRIVE IS OFF CYLINDER
; .SKI = 1, INDICATES THE DRIVE HAS AN UNEXPECTED SKI ERROR
; .DVC = 1, INDICATES AN UNEXPECTED DEVICE FAULT
;THE SUBROUTINE IS CALLED AFTER STORING STATUS IN THE GET BUFFER.
(1)  JSR      PC,STCDRVSTS
;
;      B?      ???          RETURN HERE IF NO ERROR
;      PIP     ERROR       RETURN HERE TO REPORT AN ERROR
;      R      PC,2(SP)+    ERROR NUMBER DEFINED BY SUB
;      ?      ?          GO BACK TO SUB FOR MORE ERROR CHECKS
;                          RETURN HERE IF NO MORE ERRORS
STCDRVSTS:
;CLEAR USER'S ERROR CALL
ADD     #4,(SP) ;MOVE SP TO USER'S ERROR CALL
CLRB   2(SP)   ;CLEAR ERROR NUMBER
SUB    #4,(SP) ;MOVE SP BACK TO NO ERROR RETURN
;SEE IF "MOL" = "VV" = 1, AND "PIP" = 0
MOV    RMDSI,-(SP) ;PUT DRIVE STATUS ON STACK
BIC    #1C<PIP!MOL!VV>,(SP)
CMP    #MOL!VV,(SP)+ ;ARE MOL,VV AND PIP O.K.??
BEQ    30$     ;YES!!
;REPORT AN ERROR IF MOL = 0 AND "OPI" = 0
BIT    #MOL,RMDSI ;IS MOL ON ??
BNE    10$     ;YES!!
BIT    #OPI,RMER11 ;WAS "OPI" SET??
BNE    10$     ;YES-DONT REPORT "MOL" = 0
MOV    RMDSI,$GDDAT ;EXPECTED STATUS
BIS    #MOL,$GDDAT
MOV    RMDSI,$BDDAT ;RECEIVED STATUS
ADD    #4,(SP) ;MOVE SP TO USER'S ERROR CALL
MOVB  #207,2(SP) ;WRITE ERROR NUMBER IN CALL
SUB   #2,(SP) ;MOVE SP TO RETURN FOR ERROR
JSR   PC,2(SP)+ ;REPORT ERROR VIA USER
SUB   #10,(SP) ;MOVE SP BACK TO NO ERROR RETURN

```

```

12758 062556 000240
12759 062560
12760
12761
12762 062560 032737 000100 001342
12763 062566 001030
12764 062570 032737 010000 001372
12765 062576 001024
12766 062600 013737 001342 001140
12767 062606 052737 000100 001342
12768 062614 013737 001342 001142
12769 062622 062716 000004
12770 062626 112776 000210 000000
12771 062634 162716 000002
12772 062640 004736
12773 062642 162716 000010
12774 062646 000240
12775 062650
12776
12777
12778 062650 032737 020000 001342
12779 062656 001430
12780 062660 032737 040000 001372
12781 062666 001024
12782 062670 013737 001342 001140
12783 062676 042737 020000 001140
12784 062704 013737 001342 001142
12785 062712 062716 000004
12786 062716 112776 000211 000000
12787 062724 162716 000002
12788 062730 004736
12789 062732 162716 000010
12790 062736 000240
12791 062740
12792
12793
12794 062740 013746 001372
12795 062744 042726 137577
12796 062750 001460
12797 062752
12798
12799
12800 062752 032737 000200 001372
12801 062760 001424
12802 062762 013737 001372 001140
12803 062770 042737 000200 001140
12804 062776 013737 001372 001142
12805 063004 062716 000004
12806 063010 112776 000212 000000
12807 063016 162716 000002
12808 063022 004736
12809 063024 162716 000010
12810 063030 000240
12811 063032
12812
12813

NOP
10$:
;REPORT AN ERROR IF VOLUME VALID IS NOW ZERO AND "IVC" = 0
BIT #VV,RMDSI ;IS "VV" = 0??
BNE 20$ ;NO!!
BIT #IVC,RMER2I ;WAS "IVC" SET??
BNE 20$ ;YES-DONT REPORT "VV" = 0
MOV RMDSI,$GDDAT ;EXPECTED STATUS
BIS #VV,RMDSI ;RECEIVED STATUS
MOV RMDSI,$BDDAT ;RECEIVED STATUS
ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
MOVB #210,a(SP) ;WRITE ERROR NUMBER IN CALL
SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
JSR PC,a(SP)+ ;REPORT ERROR VIA USER
SUB #10,(SP) ;MOVE SP BACK TO NO ERROR
NOP
20$:
;REPORT AN ERROR IF DRIVE IS OFF CYLINDER AND "SKI" = 0
BIT #PIP,RMDSI ;IS DRIVE OFF CYLINDER??
BEQ 30$ ;NO!!
BIT #SKI,RMER2I ;WAS "SKI" SET??
BNE 30$ ;YES-DONT REPORT "PIP" = 1
MOV RMDSI,$GDDAT ;EXPECTED STATUS
BIC #PIP,$GDDAT
MOV RMDSI,$BDDAT ;RECEIVED STATUS
ADD #4,(SP) ;MOVE SP TO USER'S ERROR CALL
MOVB #211,a(SP) ;WRITE ERROR NUMBER IN USER'S CALL
SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
JSR PC,a(SP)+ ;REPORT ERROR VIA USER
SUB #10,(SP) ;MOVE SP TO NO ERROR RETURN
NOP
30$:
;SEE IF "SKI" = "DVC" = 0
MOV RMER2I,-(SP) ;PUT ERROR REG 2 ON STACK
BIC #C<SKI!DVC>,(SP)+
BEQ 60$ ;BRANCH IF NO ERROR
40$:
;REPORT AN ERROR IF THERE IS A DEVICE FAULT
BIT #DVC,RMER2I ;ANY DEVICE FAULT??
BEQ 50$ ;NO!!
MOV RMER2I,$GDDAT ;EXPECTED STATUS
BIC #DVC,$GDDAT
MOV RMER2I,$BDDAT ;RECEIVED STATUS
ADD #4,(SP) ;MOVE SP TO USER'S CALL
MOVB #212,a(SP) ;WRITE NUMBER OF ERROR IN CALL
SUB #2,(SP) ;MOVE SP TO RETURN FOR ERROR
JSR PC,a(SP)+ ;REPORT ERROR VIA USER
SUB #10,(SP) ;MOVE SP BACK TO NO ERROR
NOP
50$:
;REPORT AN ERROR IF "SKI" = 1

```

K04

CZRMEBO RMO3/2 FC*NL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 256
STATIC DRIVE STATUS CHECK SUBROUTINE

SEQ 0256

12814	063032	032737	040000	UD1372	BIT	#SKI,RMER2I	; IS THERE A SEEK INCOMPLETE ERROR
12815	063040	001424			BEQ	60\$; NO!!
12816	063042	013737	001372	001140	MOV	RMER2I,\$GDDAT	; EXPECTED STATUS
12817	063050	042737	040000	001140	BIC	#SKI,\$GDDAT	
12818	063056	013737	001372	001142	MOV	RMER2I,\$BDDAT	; RECEIVED STATUS
12819	063064	062716	000004		ADD	#4,(SP)	; MOVE SP TO USER'S ERROR CALL
12820	063070	112776	000213	000000	MOVB	#213,a(SP)	; WRITE ERROR NUMBER IN USER'S ERROR CALL
12821	063076	162716	000002		SUB	#2,(SP)	; MOVE SP TO RETURN FOR ERROR
12822	063102	004736			JSR	PC,a(SP)+	; REPORT ERROR VIA USER
12823	063104	162716	000010		SUB	#10,(SP)	; MOVE SP BACK TO NO ERROR
12824	063110	000240			NOP		
12825	063112						
12826							
12827							
12828	063112	062716	000004		ADD	#4,(SP)	; AUGMENT THE RETURN ADDRESS IF ANY ERROR WAS DETECTED
12829	063116	105776	000000		TSTB	a(SP)	; MOVE SP TO USER'S ERROR CALL
12830	063122	001403			BEQ	70\$; WAS AN ERROR DETECTED??
12831	063124	062716	000004		ADD	#4,(SP)	; NO!!
12832	063130	000402			BR	80\$; YES - MOVE SP TO USER'S ERROR RETURN
12833	063132	162716	000004		SUB	#4,(SP)	; NO - MOVE SP TO NO ERROR RETURN
12834	063136	000240			NOP		
12835	063140	000207			RTS	PC	; RETURN TO USER

.SBTTL COMPOSITE ERROR CHECK SUBROUTINE

; THIS SUBROUTINE CHECKS THE STORED CONTENTS OF RMER1 AND
; RMER2 AFTER MASKING EACH REGISTER WORD WITH THE USER'S STATUS
; MASKS AND REPORTS AN ERROR IF ANY BITS ARE LEFT ON AFTER
; THE MASKS ARE APPLIED.

; CALL:
; (1) JSR PC, CMPERRSTS ; MASK FOR ERROR REGISTER 1
; .WORD ; MASK FOR ERROR REGISTER 2
; .WORD ; RETURN HERE IF NO ERROR
; BR ??? ; RETURN HERE TO REPORT AN ERROR
; NOP ; ERROR NUMBER DEFINED BY SUB
; ERROR ; GO BACK TO SUB FOR MOPE ERROR CHECKS
; JSR PC, @ (SP)+ ; RETURN HERE IF NO MORE ERRORS
; ???

; NOTE: BITS TO BE MASKED SHOULD BE ONE; BITS TO BE TESTED SHOULD
; BE ZERO

CMPERRSTS:

; MASK AND STORE THE CONTENTS OF RMER1 AND RMER2
; STORE RMER1 AT TEMP STORAGE
MOV RMER1I, \$TMP1 ; MASK RMER1
BIC @ (SP), \$TMP1 ; MOVE SP TO NEXT MASK
ADD #2, (SP) ; STORE RMER2 AT TEMP STORAGE
MOV RMER2I, \$TMP2 ; MASK RMER2
BIC @ (SP), \$TMP2

; CLEAR USER'S ERROR CALL
; MOVE SP TO USER'S ERROR CALL
ADD #6, (SP) ; CLEAR ERROR NUMBER
CLRB @ (SP) ; LEAVE SP AT NO ERROR RETURN
SUB #4, (SP)

; SEE IF THERE WERE ANY ERRORS IN RMER1, I.E. \$TMP1
; ANY ERRORS TO REPORT??
TST \$TMP1 ; NO !!
BEQ SS ; RECEIVED STATUS FOR TYPEOUT
MOV \$TMP1, \$BDDAT ; EXPECTED STATUS FOR TYPEOUT
CLR \$GDDAT ; MOVE SP TO USER'S ERROR CALL
ADD #4, (SP) ; CORRECTABLE DATA CHECK ERROR
MOV #6, @ (SP) ; MOVE SP TO RETURN FOR ERROR
SUB #2, (SP) ; REPORT ERROR VIA USER
JSR PC, @ (SP)+ ; MOVE SP BACK TO BRANCH
SUB #10, (SP)
NOP

SS:

; SEE IF THERE ARE ANY ERRORS TO REPORT IN RMER2 (\$TMP2)
; ANY ERRORS IN RMER2?
TST \$TMP2 ; NO !!
BEQ IOS ; RECEIVED STATUS FOR TYPEOUT
MOV \$TMP2, \$BDDAT ; EXPECTED STATUS FOR TYPEOUT
CLR \$GDDAT ; MOVE SP TO USER'S ERROR CALL
ADD #4, (SP)

12836
12837
12838
12839
12840
12841
12842
12843
12844
12845
12846
12847
12848
12849
12850
12851
12852
12853
12854
12855
12856 063142
12857
12858
12859 063142 013737 001344 001176
12860 063150 047637 000000 001176
12861 063156 062716 000002
12862 063162 013737 001372 001200
12863 063170 047637 000000 001200
12864
12865
12866
12867 063176 062716 000006
12868 063202 105076 000000
12869 063206 162716 000004
12870
12871
12872 063212 005737 001176
12873 063216 001420
12874 063220 013737 001176 001142
12875 063226 005037 001140
12876 063232 062716 000004
12877 063236 112776 000066 000000
12878 063244 162716 000002
12879 063250 004736
12880 063252 162716 000010
12881 063256 000240
12882 063260
12883
12884
12885
12886 063260 005737 001200
12887 063264 001420
12888
12889 063266 013737 001200 001142
12890 063274 005037 001140
12891 063300 062716 000004

M04

CZRMEBO RM03/2 FCTNL TST 3
 CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 258
 COMPOSITE ERROR CHECK SUBROUTINE

SEQ 0258

12892	063304	112776	000067	000000	MOVB	#67, 2(SP)	;WRITE ERROR NUMBER IN USER'S CALL
12893	063312	162716	000002		SUB	#2, (SP)	;MOVE SP TO RETURN FOR ERROR
12894	063316	004736			JSR	PC, 2(SP)+	;REPORT ERROR VIA USER
12895	063320	162716	000010		SUB	#10, (SP)	;MOVE SP TO NO ERROR RETURN
12896	063324	000240			NOP		
12897	063326						
12898							
12899							
12900	063326	062716	000004				
12901	063332	105776	000000		ADD	#4, (SP)	;MOVE SP TO USER'S ERROR CALL
12902	063336	001403			TSTB	2(SP)	;WAS THERE AN ERROR CALLED??
12903	063340	062716	000004		BEQ	20\$;NO!!
12904	063344	000402			ADD	#4, (SP)	;YES - MOVE SP TO ERROR RETURN
12905	063346	162716	000004		BR	30\$	
12906	063352	000207		20\$:	SUB	#4, (SP)	;MOVE SP TO NO ERROR RETURN
12907				30\$:	RTS	PC	;RETURN TO USER
12908							

```

12909          .SBTTL  STOP AND SHUTDOWN SUBROUTINES
12910
12911 063354      STOP:
12912
12913          ;DROP PRIORITY TO ALLOW CONSOLE INTERRUPT
12914 063354 012746 000140      MOV    #PR3,-(SP)      ;;PUT NEW PS ON STACK
12915 063360 012746 063366      MOV    #64$,-(SP)     ;;PUT NEW PC ON STACK
12916 063364 000002          RTI          ;;POP NEW PC AND PS
12917 063366
12918 063366 000240      64$:      NOP
12919          ;RAISE PRIORITY TO INHIBIT INTERRUPT
12920 063370 012746 000300      MOV    #PR6,-(SP)     ;;PUT NEW PS ON STACK
12921 063374 012746 063402      MOV    #65$,-(SP)     ;;PUT NEW PC ON STACK
12922 063400 000002          RTI          ;;POP NEW PC AND PS
12923 063402
12924
12925 063402 000207      10$:      RTS      PC          ;CONTINUE
12926
12927
12928 063404
12929 063404 005737 001326      SHUT:  TST      @#CTLFG      ;HALT ?
12930 063410 001002          .+6          ;BRANCH IF SO
12931 063412 000137 007100      JMP      READY        ;OTHERWISE LOOP
12932 063416 104401 063440      TYPE    100$        ;TYPE THE HALT MESSAGE
12933 063422 005737 000042      TST      42          ;RUNNING STANDALONE ??
12934 063426 001402          BEQ      10$         ;YES !!
12935 063430 000137 037006      JMP      $EOP        ;NO - GO TO END OF PASS
12936 063434
12937 063434 000137 005324      10$:      JMP      START        ;GO TO START
12938 063440 042524 052123 053440      100$:    .ASCIZ  @TEST WAS HALTED@<CR><LF><LF>
12939 063446 051501 044040 046101
12940 063454 042524 006504 005012
12941 063462 000
12942 063464          .EVEN

```

12943 063464 012737 177777 001326 SHUT2: MOV # -1,CTLFG
 12944 063472 000002 RTI ;SET CONTROL C FLAG
 12945
 12946
 12947
 12948
 12949
 12950
 12951
 12952
 12953
 12954
 12955
 12956
 12957
 12958
 12959
 12960
 12961
 12962 063474
 12963 063474 010046
 12964 063476 010146
 12965 063500 010246
 12966 063502 010346
 12967 063504 010446
 12968 063506 010546
 12969 063510 016646 000022
 12970 063514 016646 000022
 12971 063520 016646 000022
 12972 063524 016646 000022
 12973 063530 000002
 12974
 12975
 12976
 12977
 12978 063532
 12979 063532 012666 000022
 12980 063536 012666 000022
 12981 063542 012666 000022
 12982 063546 012666 000022
 12983 063552 012605
 12984 063554 012604
 12985 063556 012603
 12986 063560 012602
 12987 063562 012601
 12988 063564 012600
 12989 063566 000002
 12990
 12991
 12992
 12993
 12994
 12995
 12996
 12997
 12998

```

SHUT2: MOV # -1,CTLFG
        RTI ;SET CONTROL C FLAG
.SBTTL SAVE AND RESTORE RO-RS ROUTINES

;*****
;SAVE RO-RS
;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 RO, -(SP) ;: PUSH RO 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 R2(SP), -(SP) ;: SAVE PS OF MAIN FLOW
        MOV R2(SP), -(SP) ;: SAVE PC OF MAIN FLOW
        MOV R2(SP), -(SP) ;: SAVE PS OF CALL
        MOV R2(SP), -(SP) ;: SAVE PC OF CALL
        RTI

;RESTORE RO-RS
;CALL:
; RESREG
;$RESREG:
        MOV (SP)+, R2(SP) ;: RESTORE PC OF CALL
        MOV (SP)+, R2(SP) ;: RESTORE PS OF CALL
        MOV (SP)+, R2(SP) ;: RESTORE PC OF MAIN FLOW
        MOV (SP)+, R2(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
        RTI

.SBTTL BINARY TO ASCII AND TYPE ROUTINE

;*****
;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 16-BIT
;BINARY-ASCII NUMBER AND TYPE IT.
;CALL:
; MOV NUMBER, -(SP) ;: NUMBER TO BE TYPED
; TYPBN ;: TYPE IT

```


C05

CZRMEBO RMO3-2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 261
BINARY TO ASCII AND TYPE ROUTINE

SEQ 0261

```

12999 063570 010146 $TYPBN: MOV R1,-(SP) ;;SAVE R1 ON THE STACK
13000 063572 016601 000006 MOV 6(SP),R1 ;;GET THE INPUT NUMBER
13001 063576 000261 SEC ;;SET "C" SO CAN KEEP TRACK OF THE NUMBER OF BITS
13002 063600 112737 000060 063642 1$: MOVVB #'0,$BIN ;;SET CHARACTER TO AN ASCII "0".
13003 063606 006101 ROL R1 ;;GET THIS BIT
13004 063610 001406 BEQ 2$ ;;DONE?
13005 063612 105537 063642 ADCB $BIN ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
13006 063616 104401 063642 TYPE , $BIN ;;GO TYPE THIS BIT
13007 063622 000241 CLC ;;CLEAR "C" SO CAN KEEP TRACK OF BITS
13008 063624 000765 BR 1$ ;;GO DO THE NEXT BIT
13009 063626 012601 2$: MOV (SP)+,R1 ;;POP THE STACK INTO R1
13010 063630 016666 000002 000004 MOV 2(SP),4(SP) ;;ADJUST THE STACK
13011 063636 012616 MOV (SP)+,(SP)
13012 063640 000002 RTI ;;RETURN TO USER
13013 063642 000 000 $BIN: .BYTE 0,0 ;;STORAGE FOR ASCII CHAR. AND TERMINATOR
13014 .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
13015
13016 ;;*****
13017 ;;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
13018 ;;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
13019 ;;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
13020 ;;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
13021 ;;*REPLACED WITH SPACES.
13022 ;;*CALL:
13023 ;;* MOV NUM,-(SP) ;;PUT THE BINARY NUMBER ON THE STACK
13024 ;;* TYPDS ;;GO TO THE ROUTINE
13025
13026 $TYPDS:
13027 063644 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
13028 063646 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
13029 063650 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
13030 063652 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
13031 063654 010546 MOV R5,-(SP) ;;PUSH R5 ON STACK
13032 063656 012746 020200 MOV #20200,-(SP) ;;SET BLANK SWITCH AND SIGN
13033 063662 016605 000020 MOV 20(SP),R5 ;;GET THE INPUT NUMBER
13034 063666 100004 BPL 1$ ;;BR IF INPUT IS POS.
13035 063670 005405 NEG R5 ;;MAKE THE BINARY NUMBER POS.
13036 063672 112766 000055 000001 MOVVB #'-,1(SP) ;;MAKE THE ASCII NUMBER NEG.
13037 063700 005000 1$: CLR R0 ;;ZERO THE CONSTANTS INDEX
13038 063702 012703 064060 MOV #SDBLK,R3 ;;SETUP THE OUTPUT POINTER
13039 063706 112723 000040 MOVVB #' ,(R3)+ ;;SET THE FIRST CHARACTER TO A BLANK
13040 063712 005002 2$: CLR R2 ;;CLEAR THE BCD NUMBER
13041 063714 016001 064050 MOV $DTBL(R0),R1 ;;GET THE CONSTANT
13042 063720 160105 3$: SUB R1,R5 ;;FORM THIS BCD DIGIT
13043 063722 002402 BLT 4$ ;;BR IF DONE
13044 063724 005202 INC R2 ;;INCREASE THE BCD DIGIT BY 1
13045 063726 000774 BR 3$
13046 063730 060105 4$: ADD R1,R5 ;;ADD BACK THE CONSTANT
13047 063732 005702 TST R2 ;;CHECK IF BCD DIGIT=0
13048 063734 001002 BNE 5$ ;;FALL THROUGH IF 0
13049 063736 105716 TSTB (SP) ;;STILL DOING LEADING 0'S?
13050 063740 100407 BMI 7$ ;;BR IF YES
13051 063742 106316 5$: ASLB (SP) ;;MSD?
13052 063744 103003 BCC 6$ ;;BR IF NO
13053 063746 116663 000001 177777 MOVB 1(SP),-1(R3) ;;YES--SET THE SIGN
13054 063754 052702 000060 6$: BIS #'0,R2 ;;MAKE THE BCD DIGIT ASCII

```

D05

CZRMEBO RMO3/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MA(Y11 30(1046) 23-NOV-77 12:49 PAGE 262
CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

SEQ 0262

13055	063760	052702	000040	
13056	063764	110223		
13057	063766	005720		
13058	063770	020027	000010	
13059	063774	002746		
13060	063776	003002		
13061	064000	010502		
13062	064002	000764		
13063	064004	105726		
13064	064006	100003		
13065	064010	116663	177777	177776
13066	064016	105013		
13067	064020	012605		
13068	064022	012603		
13069	064024	012602		
13070	064026	012601		
13071	064030	012600		
13072	064032	104401	064060	
13073	064036	016666	000002	000004
13074	064044	012616		
13075	064046	000002		
13076	064050	023420		
13077	064052	001750		
13078	064054	000144		
13079	064056	000012		
13080	064060	000004		
13081				
13082				
13083				
13084				
13085				
13086				
13087				
13088				
13089				
13090				
13091				
13092				
13093				
13094				
13095				
13096				
13097				
13098				
13099				
13100				
13101				
13102				
13103				
13104				
13105				
13106	064070	017646	000000	
13107	064074	116537	000001	064313
13108	064102	112637	064315	
13109	064106	062716	000002	
13110	064112	000406		

```

7$:  BIS      #' R2          ;; MAKE IT A SPACE IF NOT ALREADY A DIGIT
      MOVVB   R2,(R3)+      ;; PUT THIS CHARACTER IN THE OUTPUT BUFFER
      TST     (R0)+         ;; JUST INCREMENTING
      CMP     R0,#10        ;; CHECK THE TABLE INDEX
      BLT    2$            ;; GO DO THE NEXT DIGIT
      BGT    8$            ;; GO TO EXIT
      MOV     RS,R2         ;; GET THE LSD
      BR     6$            ;; GO CHANGE TO ASCII
8$:  TSTB    (SP)+         ;; WAS THE LSD THE FIRST NON-ZERO?
      BPL    9$            ;; BR IF NO
      MOVB   -1(SP),-2(R3)  ;; YES--SET THE SIGN FOR TYPING
9$:  CLRB    (R3)          ;; SET THE TERMINATOR
      MOV    (SP)+,R5       ;; POP STACK INTO R5
      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
      TYPE   $OBLK         ;; NOW TYPE THE NUMBER
      MOV    2(SP),4(SP)   ;; ADJUST THE STACK
      MOV    (SP)+,(SP)
      RTI                    ;; RETURN TO USER

$DTBL: 10000.
        1000.
        100.
        10.

$OBLK: .BLKW 4
.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   N             ;; 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   N             ;; CALL FOR TYPEOUT
; $TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
; CALL:
;     MOV     NUM,-(SP)      ;; NUMBER TO BE TYPED
;     TYPOC   N             ;; CALL FOR TYPEOUT

$TYPOS: MOV    2(SP),-(SP)   ;; PICKUP THE MODE
        MOVB   1(SP),$OFILL  ;; LOAD ZERO FILL SWITCH
        MOVB   (SP)+,$OMCDE+1 ;; NUMBER OF DIGITS TO TYPE
        ADD    #2,(SP)      ;; ADJUST RETURN ADDRESS
        BR     $TYPON

```

E05

CZRMEBO RMO3/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 263
BINARY TO OCTAL (ASCII) AND TYPE

SEQ 0263

13111	064114	112737	000001	064313	\$TYPOC: MOV	#1,\$OFILL	:: SET THE ZERO FILL SWITCH
13112	064122	112737	000006	064315	MOV	#6,\$SOMODE+1	:: SET FOR SIX(6) DIGITS
13113	064130	112737	000005	064312	\$TYPON: MOV	#5,\$SOCNT	:: SET THE ITERATION COUNT
13114	064136	010346			MOV	R3,-(SP)	:: SAVE R3
13115	064140	010446			MOV	R4,-(SP)	:: SAVE R4
13116	064142	010546			MOV	R5,-(SP)	:: SAVE R5
13117	064144	113704	064315		MOV	\$SOMODE+1,R4	:: GET THE NUMBER OF DIGITS TO TYPE
13118	064150	005404			NEG	R4	
13119	064152	062704	000006		ADD	#6,R4	:: SUBTRACT IT FOR MAX. ALLOWED
13120	064156	110437	064314		MOV	R4,\$SOMODE	:: SAVE IT FOR USE
13121	064162	113704	064313		MOV	\$OFILL,R4	:: GET THE ZERO FILL SWITCH
13122	064166	016605	000012		MOV	12(SP),R5	:: PICKUP THE INPUT NUMBER
13123	064172	005003			CLR	R3	:: CLEAR THE OUTPUT WORD
13124	064174	006105		1\$:	ROL	R5	:: ROTATE MSB INTO "C"
13125	064176	000404			BR	3\$:: GO DO MSB
13126	064200	006105		2\$:	ROL	R5	:: FORM THIS DIGIT
13127	064202	006105			ROL	R5	
13128	064204	006105			ROL	R5	
13129	064206	010503			MOV	R5,R3	
13130	064210	006103		3\$:	ROL	R3	:: GET LSB OF THIS DIGIT
13131	064212	105337	064314		DECB	\$SOMODE	:: TYPE THIS DIGIT?
13132	064216	100016			BPL	7\$:: BR IF NO
13133	064220	042703	177770		BIC	#177770,R3	:: GET RID OF JUNK
13134	064224	001002			BNE	4\$:: TEST FOR 0
13135	064226	005704			TST	R4	:: SUPPRESS THIS 0?
13136	064230	001403			BEQ	5\$:: BR IF YES
13137	064232	005204		4\$:	INC	R4	:: DON'T SUPPRESS ANYMORE 0'S
13138	064234	052703	000060		BIS	#'0,R3	:: MAKE THIS DIGIT ASCII
13139	064240	052703	000040		BIS	#'1,R3	:: MAKE ASCII IF NOT ALREADY
13140	064244	110337	064310		MOV	R3,8\$:: SAVE FOR TYPING
13141	064250	104401	064310		TYPE	8\$:: GO TYPE THIS DIGIT
13142	064254	105337	064312		DECB	\$SOCNT	:: COUNT BY 1
13143	064260	003347			BGT	2\$:: BR IF MORE TO DO
13144	064262	002402			BLT	6\$:: BR IF DONE
13145	064264	005204			INC	R4	:: INSURE LAST DIGIT ISN'T A BLANK
13146	064266	000744			BR	2\$:: GO DO THE LAST DIGIT
13147	064270	012605		6\$:	MOV	(SP)+,R5	:: RESTORE R5
13148	064272	012604			MOV	(SP)+,R4	:: RESTORE R4
13149	064274	012603			MOV	(SP)+,R3	:: RESTORE R3
13150	064276	016666	000002 000004		MOV	2(SP),4(SP)	:: SET THE STACK FOR RETURNING
13151	064304	012616			MOV	(SP)+,(SP)	
13152	064306	000002			RTI		:: RETURN
13153	064310	000		8\$:	.BYTE	0	:: STORAGE FOR ASCII DIGIT
13154	064311	000			.BYTE	0	:: TERMINATOR FOR TYPE ROUTINE
13155	064312	000		\$SOCNT:	.BYTE	0	:: OCTAL DIGIT COUNTER
13156	064313	000		\$OFILL:	.BYTE	0	:: ZERO FILL SWITCH
13157	064314	000000		\$SOMODE:	.WORD	0	:: NUMBER OF DIGITS TO TYPE
13158				.SBTTL	TYPE ROUTINE		

13159
13160
13161
13162
13163
13164
13165
13166

```

*****
*ROUTINE TO TYPE ASCII 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.
*

```

```

13167 ;*CALL:
13168 ;*1) USING A TRAP INSTRUCTION
13169 ;* TYPE ,MESADR ; ;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
13170 ;*OR
13171 ;* TYPE
13172 ;* MESADR
13173 ;*
13174 ;*
13175 064316 105737 001173 $TYPE: TSTB $TFPLG ; ; IS THERE A TERMINAL?
13176 064322 100002 BPL 1$ ; ; BR IF YES
13177 064324 000000 HALT ; ; HALT HERE IF NO TERMINAL
13178 064326 000430 BR 3$ ; ; LEAVE
13179 064330 010046 1$: MOV RD, -(SP) ; ; SAVE RD
13180 064332 017600 000002 MOV 02(SP), RD ; ; GET ADDRESS OF ASCIZ STRING
13181 064336 122737 000001 001242 CMPB #APTENV, $ENV ; ; RUNNING IN APT MODE
13182 064344 001011 BNE 62$ ; ; NO GO CHECK FOR APT CONSOLE
13183 064346 132737 000100 001243 BITB #APTPOOL, $ENVM ; ; SPOOL MESSAGE TO APT
13184 064354 001405 BEQ 62$ ; ; NO GO CHECK FOR CONSOLE
13185 064356 010037 064366 MOV RD, 61$ ; ; SETUP MESSAGE ADDRESS FOR APT
13186 064362 004737 067226 JSR PC, $A*Y3 ; ; SPOOL MESSAGE TO APT
13187 064366 000000 61$: .WORD 0 ; ; MESSAGE ADDRESS
13188 064370 132737 000040 001243 62$: BITB #APTCSUP, $ENVM ; ; APT CONSOLE SUPPRESSED
13189 064376 001003 BNE 60$ ; ; YES, SKIP TYPE OUT
13190 064400 112046 2$: MOVB (RD)+, -(SP) ; ; PUSH CHARACTER TO BE TYPED ONTO STACK
13191 064402 001005 BNE 4$ ; ; BR IF IT ISN'T THE TERMINATOR
13192 064404 005726 TST (SP)+ ; ; IF TERMINATOR POP IT OFF THE STACK
13193 064406 012600 60$: MOV (SP)+, RD ; ; RESTORE RD
13194 064410 062716 000002 3$: ADD #2, (SP) ; ; ADJUST RETURN PC
13195 064414 000002 RTI ; ; RETURN
13196 064416 122716 000011 4$: CMPB #HT, (SP) ; ; BRANCH IF <HT>
13197 064422 001430 BEQ 8$ ; ;
13198 064424 122716 000200 CMPB #CRLF, (SP) ; ; BRANCH IF NOT <CRLF>
13199 064430 001006 BNE 5$ ; ;
13200 064432 005726 TST (SP)+ ; ; POP <CR><LF> EQUIV
13201 064434 104401 TYPE ; ; TYPE A CR AND LF
13202 064436 001217 $CRLF
13203 064440 105037 064574 CLRB $CHARCNT ; ; CLEAR CHARACTER COUNT
13204 064444 000755 BR 2$ ; ; GET NEXT CHARACTER
13205 064446 004737 064530 5$: JSR PC, $TYPEC ; ; GO TYPE THIS CHARACTER
13206 064452 123726 001172 6$: CMPB $FILLC, (SP)+ ; ; IS IT TIME FOR FILLER CHARS.?
13207 064456 001350 BNE 2$ ; ; IF NO GO GET NEXT CHAR.
13208 064460 013746 001170 MOV $NULL, -(SP) ; ; GET # OF FILLER CHARS. NEEDED
13209 ; ; AND THE NULL CHAR.
13210 064464 105366 000001 7$: DECB 1(SP) ; ; DOES A NULL NEED TO BE TYPED?
13211 064470 002770 BLT 6$ ; ; BR IF NO--GO POP THE NULL OFF OF STACK
13212 064472 004737 064530 JSR PC, $TYPEC ; ; GO TYPE A NULL
13213 064476 105337 064574 DECB $CHARCNT ; ; DO NOT COUNT AS A COUNT
13214 064502 000770 BR 7$ ; ; LOOP
13215
13216 ;HORIZONTAL TAB PROCESSOR
13217
13218 064504 112716 000040 8$: MOVB #' (SP) ; ; REPLACE TAB WITH SPACE
13219 064510 004737 064530 9$: JSR PC, $TYPEC ; ; TYPE A SPACE
13220 064514 132737 000007 064574 BITB #7, $CHARCNT ; ; BRANCH IF NOT AT
13221 064522 001372 BNE 9$ ; ; TAB STOP
13222 064524 005726 TST (SP)+ ; ; POP SPACE OFF STACK

```

```

13223 064526 000724          BR      2$          ;;GET NEXT CHARACTER
13224 064530 105777 114430  $TYPEC: TSTB  2$TSPS      ;;WAIT UNTIL PRINTER IS READY
13225 064534 100375          BPL      $TYPEC
13226 064536 116677 000002 114422  MOVB    2(SP),2$TPB      ;;LOAD CHAR TO BE TYPED INTO DATA REG.
13227 064544 122766 000015 000002  CMPB    #CR,2(SP)      ;;IS CHARACTER A CARRIAGE RETURN?
13228 064552 001003          BNE     1$          ;;BRANCH IF NO
13229 064554 105037 064574  CLR     $CHARCNT      ;;YES--CLEAR CHARACTER COUNT
13230 064560 000406          BR      $TYPEX      ;;EXIT
13231 064562 122766 000012 000002 1$: CMPB    #LF,2(SP)      ;;IS CHARACTER A LINE FEED?
13232 064570 001402          BEQ     $TYPEX      ;;BRANCH IF YES
13233 064572 105227          INCB   (PC)+        ;;COUNT THE CHARACTER
13234 064574 000000  $CHARCNT: .WORD 0    ;;CHARACTER COUNT STORAGE
13235 064576 000207  $TYPEX: RTS      PC
13236
13237
13238 .SBTTL SCOPE HANDLER ROUTINE
13239
13240 ;;*****
13241 ;;THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
13242 ;;AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG. (DISPLAY<7:0>)
13243 ;;AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
13244 ;;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
13245 ;;*SW14=1 LOOP ON TEST
13246 ;;*SW11=1 INHIBIT ITERATIONS
13247 ;;*SW09=1 LOOP ON ERROR
13248 ;;*SW08=1 LOOP ON TEST IN SWR<7:0>
13249 ;;CALL
13250 ;;* SCOPE ;;SCOPE=IOT
13251
13251 064600  $SCOPE:
13252 064600 104410          CKSWR
13253 064602 004737 063354          JSR     PC,STOP      ;;TEST FOR CHANGE IN SOFT-SWR
13254 064606 032777 040000 114340 1$: BIT     #BIT14,2$SWR  ;;LOOP ON PRESENT TEST?
13255 064614 001131          BNE     $OVER      ;;YES IF SW14=1
13256
13257 064616 000416  $XTSTR: BR     6$      ;;*****START OF CODE FOR THE XOR TESTER*****
13258
13259 064620 013746 000004          MOV     2$ERRVEC, -(SP) ;;IF RUNNING ON THE "XOR" TESTER CHANGE
13260 064624 012737 064644 000004  MOV     #27,2$ERRVEC  ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
13261 064632 005737 177060          TST    2$177060      ;;SAVE THE CONTENTS OF THE ERROR VECTOR
13262 064636 012637 000004          MOV     (SP)+,2$ERRVEC ;;SET FOR TIMEOUT
13263 064642 000500          BR     $$VLAD        ;;TIME OUT ON XOR?
13264 064644 022626          $S: CMP   (SP)+,(SP)+  ;;RESTORE THE ERROR VECTOR
13265 064646 012637 000004          MOV   (SP)+,2$ERRVEC ;;GO TO THE NEXT TEST
13266 064652 000440          BR     7$          ;;CLEAR THE STACK AFTER A TIME OUT
13267 064654          $B: ;;*****END OF CODE FOR THE XOR TESTER*****
13268 064654 032777 000400 114272  BIT     #BIT08,2$SWR  ;;LOOP ON SPEC. TEST?
13269 064662 001421          BEQ     2$          ;;BR IF NO
13270 064664 005046          CLR    -(SP)        ;;CLEAR A TEMP. LOCATION
13271 064666 117716 114262          MOVB   2$SWR,(SP)    ;;PICKUP THE DESIRED TEST NUMBER
13272 064672 001414          BEQ     8$          ;;BRANCH IF BAD TEST NUMBER IN SWR
13273 064674 022716 000027          CMP   #27,(SP)     ;;CHECK THE NUMBER IN THE SWR
13274 064700 002411          BLT   8$          ;;BRANCH IF TEST NUMBER IS OUT OF RANGE
13275 064702 011637 001116          MOV   (SP),2$TSTNM  ;;UPDATE THE TEST NUMBER
13276 064706 005316          DEC   (SP)         ;;BACKUP BY ONE
13277 064710 006316          ASL   (SP)         ;;SCALE THE TEST NUMBER AS AN INDEX
13278 064712 062716 065116          ADD   #2$SW08TBL,(SP) ;;FORM THE ADDRESS OF TEST POINTER

```

13279	064716	013637	001122		MOV	2(SP)+, \$LPADR	;; SET LOOP ADDRESS TO DESIRED TEST
13280	064722	000466			BR	\$OVER	;; GO LOOP ON THE TEST
13281	064724	005726		8\$:	TST	(SP)+	;; CLEAN THE BAD TEST NUMBER OFF OF THE STACK
13282	064726	105737	001117	2\$:	TSTB	\$ERFLG	;; HAS AN ERROR OCCURRED?
13283	064732	001421			BEQ	3\$;; BR IF NO
13284	064734	123737	001131	001117	CMPB	\$ERMAX, \$ERFLG	;; MAX. ERRORS FOR THIS TEST OCCURRED?
13285	064742	101015			BHI	3\$;; BR IF NO
13286	064744	032777	001000	114202	BIT	#BIT09, 2SWR	;; LOOP ON ERROR?
13287	064752	001404			BEQ	4\$;; BR IF NO
13288	064754	013737	001124	001122	7\$:	MOV	\$LPERR, \$LPADR
13289	064762	000446			BR	\$OVER	;; SET LOOP ADDRESS TO LAST SCOPE
13290	064764	105037	001117		4\$:	CLRB	\$ERFLG
13291	064770	005037	001206		CLR	\$TIMES	;; ZERO THE ERROR FLAG
13292	064774	000415			BR	1\$;; CLEAR THE NUMBER OF ITERATIONS TO MAKE
13293	064776	032777	004000	114150	3\$:	BIT	#BIT11, 2SWR
13294	065004	001011			BNE	1\$;; ESCAPE TO THE NEXT TEST
13295	065006	005737	001230		TST	\$PASS	;; INHIBIT ITERATIONS?
13296	065012	001406			BEQ	1\$;; BR IF YES
13297	065014	005237	001120		INC	\$ICNT	;; IF FIRST PASS OF PROGRAM
13298	065020	023737	001206	001120	CMP	\$TIMES, \$ICNT	;; INHIBIT ITERATIONS
13299	065026	002024			BGE	\$OVER	;; INCREMENT ITERATION COUNT
13300	065030	012737	000001	001120	1\$:	MOV	#1, \$ICNT
13301	065036	013737	065114	001206	MOV	\$MXCNT, \$TIMES	;; CHECK THE NUMBER OF ITERATIONS MADE
13302	065044	105237	001116		SSVLAD:	INCB	\$STNM
13303	065050	113737	001116	001226	MOV	\$STNM, \$STNM	;; BR IF MORE ITERATION REQUIRED
13304	065056	011637	001122		MOV	(SP), \$LPADR	;; REINITIALIZE THE ITERATION COUNTER
13305	065062	011637	001124		MOV	(SP), \$LPERR	;; SET NUMBER OF ITERATIONS TO DO
13306	065066	005037	001210		CLR	\$ESCAPE	;; COUNT TEST NUMBERS
13307	065072	112737	000001	001131	MOV	#1, \$ERMAX	;; SET TEST NUMBER IN APT MAILBOX
13308	065100	013777	001116	114050	SOVER:	MOV	\$STNM, 2DISPLAY
13309	065106	013716	001122		MOV	\$LPADR, (SP)	;; SAVE SCOPE LOOP ADDRESS
13310	065112	000002			RTI		;; SAVE ERROR LOOP ADDRESS
13311	065114	000012			\$MXCNT:	10.	;; CLEAR THE ESCAPE FROM ERROR ADDRESS
13312	065116				SSWOBTBL:		;; ONLY ALLOW ONE(1) ERROR ON NEXT TEST
13313	065116	007140			.WORD	TST1+2	;; DISPLAY TEST NUMBER
13314	065120	007336			.WORD	TST2+2	;; FUDGE RETURN ADDRESS
13315	065122	007522			.WORD	TST3+2	;; FIXES PS
13316	065124	007704			.WORD	TST4+2	;; MAX. NUMBER OF ITERATIONS
13317	065126	010726			.WORD	TST5+2	;; STARTING ADDRESS OF TEST 1
13318	065130	011720			.WORD	TST6+2	;; STARTING ADDRESS OF TEST 2
13319	065132	013160			.WORD	TST7+2	;; STARTING ADDRESS OF TEST 3
13320	065134	014202			.WORD	TST10+2	;; STARTING ADDRESS OF TEST 4
13321	065136	015174			.WORD	TST11+2	;; STARTING ADDRESS OF TEST 5
13322	065140	016436			.WORD	TST12+2	;; STARTING ADDRESS OF TEST 6
13323	065142	017456			.WORD	TST13+2	;; STARTING ADDRESS OF TEST 7
13324	065144	021004			.WORD	TST14+2	;; STARTING ADDRESS OF TEST 10
13325	065146	022024			.WORD	TST15+2	;; STARTING ADDRESS OF TEST 11
13326	065150	023044			.WORD	TST16+2	;; STARTING ADDRESS OF TEST 12
13327	065152	024742			.WORD	TST17+2	;; STARTING ADDRESS OF TEST 13
13328	065154	026416			.WORD	TST20+2	;; STARTING ADDRESS OF TEST 14
13329	065156	027524			.WORD	TST21+2	;; STARTING ADDRESS OF TEST 15
13330	065160	030472			.WORD	TST22+2	;; STARTING ADDRESS OF TEST 16
13331	065162	031550			.WORD	TST23+2	;; STARTING ADDRESS OF TEST 17
13332	065164	032636			.WORD	TST24+2	;; STARTING ADDRESS OF TEST 20
13333	065166	033830			.WORD	TST25+2	;; STARTING ADDRESS OF TEST 21
13334	065170	034652			.WORD	TST26+2	;; STARTING ADDRESS OF TEST 22

```

13335 065172 035674 .WORD TST27+2 ;;STARTING ADDRESS OF TEST 27
13336 .SBTTL ERROR HANDLER ROUTINE
13337
13338 ;*****
13339 ;THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
13340 ;SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
13341 ;AND GO TO ERRTP ON ERROR
13342 ;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
13343 ;SW15=1 HALT ON ERROR
13344 ;SW13=1 INHIBIT ERROR TYPEOUTS
13345 ;SW10=1 BELL ON ERROR
13346 ;SW09=1 LOOP ON ERROR
13347 ;CALL
13348 ;* ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER
13349
13350 $ERROR:
13351 065174 104410
13352 065176 105237 001117 7$: CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
13353 065202 001775 INCB $ERFLG ;;SET THE ERROR FLAG
13354 065204 013777 001116 113744 BEQ 7$ ;;DON'T LET THE FLAG GO TO ZERO
13355 065212 032777 002000 113734 MOV $TSTNM,$DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
13356 065220 001402 BIT #BIT10,$SWR ;;BELL ON ERROR?
13357 065222 104401 001212 BEQ 1$ ;;NO - SKIP
13358 065226 005237 001126 1$: INC $BELL ;;RING BELL
13359 065232 011637 001132 INC $ERTTL ;;COUNT THE NUMBER OF ERRORS
13360 065236 162737 000002 001132 MOV (SP),$ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
13361 065244 117737 113662 001130 SUB #2,$ERRPC
13362 065252 032777 020000 113674 MOVB $ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
13363 065260 001004 BNE 20$ ;;SKIP TYPEOUT IF SET
13364 065262 004737 037216 JSR PC,ERRTP ;;SKIP TYPEOUTS
13365 065266 104401 001217 TYPE ,SCLRF ;;GO TO USER ERROR ROUTINE
13366 065272
13367 065272 122737 000001 001242 20$: CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
13368 065300 001007 BNE 2$ ;;NO SKIP APT ERROR REPORT
13369 065302 113737 001130 065314 MOVB $ITEMB,21$ ;;SET ITEM NUMBER AS ERROR NUMBER
13370 065310 004737 057236 JSR PC,SATY4 ;;REPORT FATAL ERROR TO APT
13371 065314 000
13372 065315 000 21$: .BYTE 0
13373 065316 000777 .BYTE 0
13374 065320 005777 113630 22$: BR 22$ ;;APT ERROR LOOP
13375 065324 100002 TST $SWR ;;HALT ON ERROR
13376 065326 000000 BPL 3$ ;;SKIP IF CONTINUE
13377 065330 104410 HALT ;;HALT ON ERROR!
13378 065332 032777 001000 113614 3$: BIT #BIT09,$SWR ;;TEST FOR CHANGE IN SOFT-SWR
13379 065340 001402 BEQ 4$ ;;LOOP ON ERROR SWITCH SET?
13380 065342 013716 001124 MOV $LPERR,(SP) ;;BR IF NO
13381 065346 005737 001210 4$: TST $ESCAPE ;;FUDGE RETURN FOR LOOPING
13382 065352 001402 BEQ 5$ ;;CHECK FOR AN ESCAPE ADDRESS
13383 065354 013716 001210 MOV $ESCAPE,(SP) ;;BR IF NONE
13384 065360
13385 065360 022737 037176 000042 5$: CMP #SENDAD,$42 ;;FUDGE RETURN ADDRESS FOR ESCAPE
13386 065366 001001 BNE 6$ ;;ACT-11 AUTO-ACCEPT?
13387 065370 000000 HALT ;;BRANCH IF NO
13388 065372
13389 065372 000002 6$: RTI ;;YES
13390 .SBTTL TTY INPUT ROUTINE ;;RETURN

```

```

13391
13392
13393
13394 065374 000000
13395 065376 000000
13396 065400 000000
13397 065402 000001
13398 065403
13399 065404
13400
13401
13402
13403
13404
13405
13406
13407
13408
13409 065404 005037 065374
13410 065410 012737 065402 065376
13411 065416 013737 065376 065400
13412 065424 012737 065454 000060
13413 065432 012737 000200 000062
13414 065440 005777 113516
13415 065444 012777 000100 113506
13416 065452 000207
13417
13418
13419
13420
13421
13422
13423
13424
13425 065454 117746 113502
13426 065460 042716 177600
13427 065464 021627 000003
13428 065470 001007
13429 065472 104401 066570
13430 065476 004737 065404
13431 065502 005726
13432 065504 000137 063464
13433 065510 021627 000007
13434 065514 001004
13435 065516 022737 000176 001154
13436 065524 001500
13437
13438 065526
13439 065526 022737 000001 065374
13440 065534 001004
13441 065536 104401 001212
13442 065542 005726
13443 065544 000451
13444 065546 021627 000023
13445 065552 001021
13446 065554 005077 113400

;*****
;ENABL LSB
$TKCNT: .WORD 0 ;: NUMBER OF ITEMS IN QUEUE
$TKQIN: .WORD 0 ;: INPUT POINTER
$TKQOUT: .WORD 0 ;: OUTPUT POINTER
$TKQSRV: .BLKB 1 ;: TTY KEYBOARD QUEUE
$TKQEND=.
.EVEN

; *TK INITIALIZE ROUTINE
; *THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
; *SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
; *CALL:
; * JSR PC,$TKINT
; * RETURN
$TKINT: CLR $TKCNT ;: CLEAR COUNT OF ITEMS IN QUEUE
MOV $TKQSRV,$TKQIN ;: MOVE THE STARTING ADDRESS OF THE
MOV $TKQIN,$TKQOUT ;: QUEUE INTO THE INPUT & OUTPUT POINTERS.
MOV $TKSRV,$TKVEC ;: INITIALIZE THE KEYBOARD VECTOR
MOV #200,$TKVEC+2 ;: "BR" LEVEL 4
TST $TKB ;: CLEAR DONE FLAG
MOV #100,$TKS ;: ENABLE TTY KEYBOARD INTERRUPT
RTS PC ;: RETURN TO CALLER

; *TK SERVICE ROUTINE
; *THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
; *BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
; *IT IN THE QUEUE.
; *IF THE CHARACTER IS A "CONTROL-C" (↑C) $TKINT IS CALLED AND
; *UPON RETURN EXIT IS MADE TO THE "CONTROL-C" RESTART ADDRESS (SHUT2)
$TKSRV: MOVB $TKB,-(SP) ;: PICKUP THE CHARACTER
BIC #↑C177,(SP) ;: STRIP THE JUNK
CMP (SP),#3 ;: IS IT A CONTROL C?
BNE IS ;: BRANCH IF NO
TYPE $CNTLC ;: TYPE A CONTROL-C (↑C)
JSR PC,$TKINT ;: INIT THE KEYBOARD
TST (SP)+ ;: CLEAN UP STACK
JMP SHUT2 ;: CONTROL C RESTART
IS: CMP (SP),#7 ;: IS IT A CONTROL G?
BNE 2$ ;: BRANCH IF NO
CMP $SWREG,$SWR ;: IS SOFT-SWR SELECTED?
BEQ 6$ ;: GO TO SWR CHANGE

2$: CMP #1,$TKCNT ;: IS THE QUEUE FULL?
BNE 3$ ;: BRANCH IF NO
TYPE $BELL ;: RING THE TTY BELL
TST (SP)+ ;: CLEAN CHARACTER OFF OF STACK
BR 5$ ;: EXIT
3$: CMP (SP),#23 ;: IS IT A CONTROL-S?
BNE 32$ ;: BRANCH IF NO
CLR $TKS ;: DISABLE TTY KEYBOARD INTERRUPTS

```



```

13447 065560 005726
13448 065562 105777 113372
13449 065566 100375
13450 065570 117746 113366
13451 065574 042716 177600
13452 065600 022627 000021
13453 065604 001366
13454 065606 012777 000100 113344
13455 065614 000002
13456 065616 005237 065374
13457 065622 021627 000140
13458 065626 002405
13459 065630 021627 000175
13460 065634 003002
13461 065636 042716 000040
13462 065642 112677 177530
13463 065646 005237 065376
13464 065652 023727 065403
13465 065660 001003
13466 065662 012737 065402 065376
13467 065670 000002
13468
13469
13470
13471
13472
13473
13474 065672 022737 000176 001154
13475 065700 001124
13476 065702 105777 113252
13477 065706 100121
13478 065710 117746 113246
13479 065714 042716 177600
13480 065720 021627 000007
13481 065724 001300
13482
13483
13484
13485
13486
13487
13488 065726 123727 001150 000001
13489 065734 001674
13490 065736 005726
13491 065740 004737 065404
13492 065744 005077 113210
13493 065750 112737 000001 001151
13494
13495 065756 104401 066602
13496 065762 104401 066607
13497 065766 013746 000176
13498 065772 104402
13499 065774 104401 066620
13500 066000 005046
13501 066002 005046
13502 066004 105777 113150

31$: TST (SP)+ ; CLEAN CHAR OFF STACK
TSTB @STKS ; WAIT FOR A CHAR
BPL 31$ ; LOOP UNTIL ITS THERE
MOVB @STKB, -(SP) ; GET THE CHARACTER
BIC #1C177, (SP) ; MAKE IT 7-BIT ASCII
CMP (SP)+, #21 ; IS IT A CONTROL-Q?
BNE 31$ ; BRANCH IF NO
MOV #100, @STKS ; REENABLE TTY KEYBOARD INTERRUPTS
RTI ; RETURN

32$: INC $TKCNT ; COUNT THIS CHARACTER
CMP (SP), #140 ; IS IT UPPER CASE?
BLT 4$ ; BRANCH IF YES
CMP (SP), #175 ; IS IT A SPECIAL CHAR?
BGT 4$ ; BRANCH IF YES
BIC #40, (SP) ; MAKE IT UPPER CASE
MOVB (SP)+, @STKQIN ; AND PUT IT IN QUEUE
INC $TKQIN ; UPDATE THE POINTER
CMP $TKQIN, $STKQEND ; GO OFF THE END?
BNE 5$ ; BRANCH IF NO
MOV #STKGSRT, $TKQIN ; RESET THE POINTER
RTI ; RETURN

*****
; *SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
; *ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
; *SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
; *CALL WHEN OPERATING IN TTY INTERRUPT MODE.
$CKSWR: CMP #SWREG, SWR ; IS THE SOFT-SWR SELECTED
BNE 15$ ; EXIT IF NOT
TSTB @STKS ; IS A CHAR WAITING?
BPL 15$ ; IF NOT, EXIT
MOVB @STKB, -(SP) ; YES
BIC #1C177, (SP) ; MAKE IT 7-BIT ASCII
CMP (SP), #7 ; IS IT A CONTROL-G?
BNE 2$ ; IF NOT, PUT IT IN THE TTY QUEUE
; AND EXIT

*****
; *CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
; *ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
; *CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
6$: CMPB $AUTOB, #1 ; ARE WE RUNNING IN AUTO-MODE?
BEQ 2$ ; BRANCH IF YES
TST (SP)+ ; CLEAR CONTROL-G OFF STACK
JSR PC, $TKINT ; FLUSH THE TTY INPUT QUEUE
CLR @STKS ; DISABLE TTY KEYBOARD INTERRUPTS
MOVB #1, $INTAG ; SET INTERRUPT MODE INDICATOR

; *ECHO THE CONTROL-G (+G)
; *TYPE CURRENT CONTENTS
; *SAVE SWREG FOR TYPEOUT
; *GO TYPE--OCTAL ASCII(ALL DIGITS)
; *PROMPT FOR NEW SWR
$GTSWR: TYPE , $CNTLG
TYPE , $MSWR
MOV SWREG, -(SP)
TYPOC
TYPE , $MNEW
19$: CLR -(SP) ; CLEAR COUNTER
CLR -(SP) ; THE NEW SWR
7$: TSTB @STKS ; CHAR THERE?

```

```

13503 066010 100375          BPL      7$          ;; IF NOT TRY AGAIN
13504
13505 066012 117746 113144    MOVB     @STKB, -(SP)    ;; PICK UP CHAR
13506 066016 042716 177600    BIC     #1C177, (SP)   ;; MAKE IT 7-BIT ASCII
13507
13508 066022 021627 000003    CMP     (SP), #3       ;; IS IT A CONTROL-C?
13509 066026 001015          BNE     9$             ;; BRANCH IF NOT
13510 066030 104401 066570    TYPE   $CNTLC         ;; YES, ECHO CONTROL-C (↑C)
13511 066034 062706 000006    ADD     #6, SP        ;; CLEAN UP STACK
13512 066040 123727 001151 000001  CMPB   $INTAG, #1     ;; REENABLE TTY KEYBOARD INTERRUPTS?
13513 066046 001003          BNE     8$             ;; BRANCH IF NO
13514 066050 012777 000100 113102  MOV    #100, @STKS    ;; ALLOW TTY KEYBOARD INTERRUPTS
13515 066056 000137 063464    8$:    JMP     SHUT2        ;; CONTROL-C RESTART
13516
13517
13518 066062 021627 000025    9$:    CMP     (SP), #25    ;; IS IT A CONTROL-U?
13519 066066 001005          BNE     10$           ;; BRANCH IF NOT
13520 066070 104401 066575    TYPE   $CNTLU         ;; YES, ECHO CONTROL-U (↑U)
13521 066074 062706 000006    20$:   ADD     #6, SP        ;; IGNORE PREVIOUS INPUT
13522 066100 000737          BR     19$           ;; LET'S TRY IT AGAIN
13523
13524
13525 066102 021627 000015    10$:   CMP     (SP), #15    ;; IS IT A <CR>?
13526 066106 001022          BNE     16$           ;; BRANCH IF NO
13527 066110 005766 000004    TST    4(SP)          ;; YES, IS IT THE FIRST CHAR?
13528 066114 001403          BEQ    11$           ;; BRANCH IF YES
13529 066116 016677 000002 113030  MOV    2(SP), @SWR    ;; SAVE NEW SWR
13530 066124 062706 000006    11$:   ADD     #6, SP        ;; CLEAN UP STACK
13531 066130 104401 001217    14$:   TYPE   $CRLF         ;; ECHO <CR> AND <LF>
13532 066134 123727 001151 000001  CMPB   $INTAG, #1     ;; RE-ENABLE TTY KBD INTERRUPTS?
13533 066142 001003          BNE     15$           ;; BRANCH IF NOT
13534 066144 012777 000100 113006  MOV    #100, @STKS    ;; RE-ENABLE TTY KBD INTERRUPTS
13535 066152 000002          RTI                    ;; RETURN
13536 066154 004737 064530    16$:   JSR    PC, $TYPEPC   ;; ECHO CHAR
13537 066160 021627 000060    CMP    (SP), #60     ;; CHAR < 0?
13538 066164 002420          BLT    18$           ;; BRANCH IF YES
13539 066166 021627 000067    CMP    (SP), #67     ;; CHAR > 7?
13540 066172 003015          BGT    18$           ;; BRANCH IF YES
13541 066174 042726 000060    BIC    #60, (SP)+    ;; STRIP-OFF ASCII
13542 066200 005766 000002    TST    2(SP)         ;; IS THIS THE FIRST CHAR
13543 066204 001403          BEQ    17$           ;; BRANCH IF YES
13544 066206 006316          ASL    (SP)          ;; NO, SHIFT PRESENT
13545 066210 006316          ASL    (SP)          ;; CHAR OVER TO MAKE
13546 066212 006316          ASL    (SP)          ;; ROOM FOR NEW ONE.
13547 066214 005266 000002 17$:   INC    2(SP)         ;; KEEP COUNT OF CHAR
13548 066220 056616 177776    BIS    -2(SP), (SP)  ;; SET IN NEW CHAR
13549 066224 000667          BR     7$            ;; GET THE NEXT ONE
13550 066226 104401 001216    18$:   TYPE   $QUES        ;; TYPE ?<CR><LF>
13551 066232 000720          BR     20$          ;; SIMULATE CONTROL-U
13552
13553
13554
13555
13556
13557
13558
; *****
; THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
; *CALL:
; *      RDCHR          ;; GET A CHARACTER FROM THE QUEUE

```

MOS

CZRMEBO RMO3/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 271
TTY INPUT ROUTINE

SEQ 0271

```

13559          ;*      RETURN HERE          ;; CHARACTER IS ON THE STACK
13560          ;*                               ;; WITH PARITY BIT STRIPPED OFF
13561          ;
13562          ;
13563          ;RDCHR: MOV      (SP), -(SP)      ;; PUSH DOWN THE PC AND
13564          066234 011646 000004 000002  MOV      4(SP), 2(SP)      ;; THE PS
13565          066236 016666 005066 000004  CLR      4(SP)          ;; GET READY FOR A CHARACTER
13566          066244 005066 005046          CLR      -(SP)         ;; PUT NEW PS ON STACK
13567          066250 005046 066260  MOV      #64$, -(SP)    ;; PUT NEW PC ON STACK
13568          066252 012746 066260  RTI          ;; POP NEW PC AND PS
13569          066256 000002
13570          64$: TST      $TKCNT          ;; WAIT ON A CHARACTER
13571          1$: BEQ      1$
13572          066260 005737 065374  DEC      $TKCNT        ;; DECREMENT THE COUNTER
13573          066264 001775 065374  MOV      $TKQOUT, 4(SP)  ;; GET ONE CHARACTER
13574          066266 005337 177102 000004  MOV      $TKQOUT        ;; UPDATE THE POINTER
13575          066272 117766 065400  INC      $TKQOUT        ;; DID IT GO OFF OF THE END?
13576          066300 005237 065400 065403  CMP      $TKQOUT, #STKQEND ;; BRANCH IF NO
13577          066304 023727 065400 065402  BNE      2$           ;; RESET THE POINTER
13578          066312 001003 065402 065400  MOV      #STKQSRST, $TKQOUT
13579          066314 012737 065402 065400  RTI          ;; RETURN
13580          066322 000002
13581          ;*****
13582          ; THIS ROUTINE WILL INPUT A STRING FROM THE TTY
13583          ; CALL:
13584          ;*      RDLIN          ;; INPUT A STRING FROM THE TTY
13585          ;*      RETURN HERE  ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
13586          ;*                               ;; TERMINATOR WILL BE A BYTE OF ALL D'S
13587          ;
13588          ;RDLIN: MOV      R3, -(SP)        ;; SAVE R3
13589          066324 010346 005046          CLR      -(SP)         ;; CLEAR THE RUBOUT KEY
13590          066326 005046 066560 1$: MOV      #STTYIN, R3    ;; GET ADDRESS
13591          066330 012703 066570 2$: CMP      #STTYIN+8., R3  ;; BUFFER FULL?
13592          066334 022703 066570  BLOS     4$           ;; BR IF YES
13593          066340 101456 066570  RDCHR    4$           ;; GO READ ONE CHARACTER FROM THE TTY
13594          066342 104411 066570  MOV      (SP)+, (R3)    ;; GET CHARACTER
13595          066344 112613 066570  CMP      #177, (R3)    ;; IS IT A RUBOUT
13596          066346 122713 000177  BNE      5$           ;; BR IF NO
13597          066352 001022 066570  TST      (SP)         ;; IS THIS THE FIRST RUBOUT?
13598          066354 005716 066570  BNE      6$           ;; BR IF NO
13599          066356 001007 066570  MOV      #' \, 9$     ;; TYPE A BACK SLASH
13600          066360 112737 000134 066556  TYPE     9$
13601          066366 104401 066556  MOV      #-1, (SP)    ;; SET THE RUBOUT KEY
13602          066372 012716 177777 6$: DEC      R3         ;; BACKUP BY ONE
13603          066376 005303 066560  CMP      R3, #STTYIN  ;; STACK EMPTY?
13604          066400 020327 066560  BLO      4$           ;; BR IF YES
13605          066404 103434 066556  MOV      (R3), 9$    ;; SETUP TO TYPEOUT THE DELETED CHAR.
13606          066406 111337 066556  TYPE     9$
13607          066412 104401 066556  BR      2$           ;; GO TYPE
13608          066416 000746 066556  TST      (SP)         ;; GO READ ANOTHER CHAR.
13609          066420 005716 066556  BNE      5$           ;; RUBOUT KEY SET?
13610          066422 001406 066556  BEQ      7$           ;; BR IF NO
13611          066424 001406 000134 066556  MOV      #' \, 9$     ;; TYPE A BACK SLASH
13612          066426 112737 000134 066556  TYPE     9$
13613          066432 104401 066556  CLR      (SP)        ;; CLEAR THE RUBOUT KEY
13614          066436 005016 000025 7$: CMP      #25, (R3)  ;; IS CHARACTER A CTRL U?
13615          066440 122713 000025  BNE      8$           ;; BR IF NO
13616          066444 001003 066575  TYPE     $CNTLU      ;; TYPE A CONTROL "U"
13617          066446 104401 066575  BR      1$           ;; GO START OVER
13618          066452 000726

```

```

13615 066454 122713 000022      8$:  CMPB    #22,(R3)      ;; IS CHARACTER A "r"?
13616 066460 001011              BNE     3$              ;; BRANCH IF NO
13617 066462 105013              CLRB   (R3)            ;; CLEAR THE CHARACTER
13618 066464 104401 001217      TYPE   ,SCLF           ;; TYPE A "CR" & "LF"
13619 066470 104401 066560      TYPE   $TTYIN         ;; TYPE THE INPUT STRING
13620 066474 000717              BR     2$              ;; GO PICKUP ANOTHER CHACTER
13621 066476 104401 001216      4$:  TYPE   $QUES      ;; TYPE A '?'
13622 066502 000712              BR     1$              ;; CLEAR THE BUFFER AND LOOP
13623 066504 111337 066556      3$:  MOVB   (R3),9$     ;; ECHO THE CHARACTER
13624 066510 104401 066556      TYPE   9$              ;;
13625 066514 122723 000015      CMPB   #15,(R3)+      ;; CHECK FOR RETURN
13626 066520 001305              BNE     2$              ;; LOOP IF NOT RETURN
13627 066522 105063 177777      CLRB   -1(R3)         ;; CLEAR RETURN (THE 15)
13628 066526 104401 001220      TYPE   $LF            ;; TYPE A LINE FEED
13629 066532 005726              TST    (SP)+           ;; CLEAN RUBOUT KEY FROM THE STACK
13630 066534 012603              MOV    (SP)+,R3       ;; RESTORE R3
13631 066536 011646              MOV    (SP),-(SP)     ;; ADJUST THE STACK AND PUT ADDRESS OF THE
13632 066540 016666 000004 000002  MOV    4(SP),2(SP)    ;; FIRST ASCII CHARACTER ON IT
13633 066546 012766 066560 000004  MOV    $TTYIN,4(SP)  ;;
13634 066554 000002              RTI                    ;; RETURN
13635 066556 000          9$:  .BYTE   0              ;; STORAGE FOR ASCII CHAR. TO TYPE
13636 066557 000          .BYTE   0              ;; TERMINATOR
13637 066560 000010      $TTYIN: .BLKB   8.      ;; RESERVE 8 BYTES FOR TTY INPUT
13638 066570 041536 005015 000      $CNTLC: .ASCIZ /C<15><12> ;; CONTROL "C"
13639 066575 136 006525 000012  $CNTLU: .ASCIZ /U<15><12> ;; CONTROL "U"
13640 066602 043536 005015 000      $CNTLG: .ASCIZ /G<15><12> ;; CONTROL "G"
13641 066607 015 051412 051127  $MSWR:  .ASCIZ <15><12>/SWR = /
13642 066614 036440 000040      $MNEW:  .ASCIZ / NEW = /
13643 066620 020040 042516 020127
13644 066626 020075 000
13645 066632
13646
13647
13648
13649
13650
13651
13652
13653
13654
13655
13656 066632 011646 000004 000002  $RDOCT: MOV    (SP),-(SP)  ;; PROVIDE SPACE FOR THE
13657 066634 016666              MOV    4(SP),2(SP)    ;; INPUT NUMBER
13658 066642 010046              MOV    R0,-(SP)       ;; PUSH R0 ON STACK
13659 066644 010146              MOV    R1,-(SP)       ;; PUSH R1 ON STACK
13660 066646 010246              MOV    R2,-(SP)       ;; PUSH R2 ON STACK
13661 066650 104412      1$:  ROLIN   ;; READ AN ASCII LINE
13662 066652 012600      MOV    (SP)+,R0       ;; GET ADDRESS OF 1ST CHARACTER
13663 066654 005001      CLR    R1              ;; CLEAR DATA WORD
13664 066656 005002      CLR    R2
13665 066660 112046      2$:  MOVB   (R0)+,-(SP)   ;; PICKUP THIS CHARACTER
13666 066662 001412      BEQ    3$              ;; IF ZERO GET OUT
13667 066664 006301      ASL   R1                ;; *2
13668 066666 006102      ROL   R2
13669 066670 006301      ASL   R1                ;; *4
13670 066672 006102      ROL   R2

```

```

.EVEN
$RATTL READ AN OCTAL NUMBER FROM THE TTY
*****
*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
*CHANGE IT TO BINARY.
*CALL:
*   RDOCT
*   RETURN HERE
*   READ AN OCTAL NUMBER
*   LOW ORDER BITS ARE ON TOP OF THE STACK
*   HIGH ORDER BITS ARE IN $HIOCT

```

13671 066674 006301
 13672 066676 006102
 13673 066700 042716 177770
 13674 066704 062601
 13675 066706 000764
 13676 066710 005726
 13677 066712 010166 000012
 13678 066716 010237 066732
 13679 066722 012602
 13680 066724 012601
 13681 066726 012600
 13682 066730 000002
 13683 066732 000000
 13684
 13685
 13686
 13687
 13688
 13689
 13690
 13691
 13692 066734 016646 000002
 13693 066740 042716 000020
 13694 066744 012746 066752
 13695 066750 000002
 13696 066752 010046
 13697 066754 016600 000002
 13698 066760 005740
 13699 066762 111000
 13700 066764 006300
 13701 066766 016000 067006
 13702 066772 000200
 13703
 13704
 13705
 13706
 13707 066774 011646
 13708 066776 016666 000004 000002
 13709 067004 000002
 13710
 13711
 13712
 13713
 13714
 13715
 13716
 13717
 13718 067006 066774
 13719 067010 064316
 13720 067012 064114
 13721 067014 064070
 13722 067016 064130
 13723 067020 063644
 13724 067022 063570
 13725
 13726 067024 065762

```

ASL R1 ;;#8
ROL R2
BIC #1C7,(SP) ;;STRIP THE ASCII JUNK
ADD (SP)+,R1 ;;ADD IN THIS DIGIT
BR 2$ ;;LOOP
3$: TST (SP)+ ;;CLEAN TERMINATOR FROM STACK
MOV R1,12(SP) ;;SAVE THE RESULT
MOV R2,$HI OCT
MOV (SP)+,R2 ;;POP STACK INTO R2
MOV (SP)+,R1 ;;POP STACK INTO R1
MOV (SP)+,R0 ;;POP STACK INTO R0
RTI ;;RETURN
$HI OCT: .WORD 0 ;;HIGH ORDER BITS GO HERE
.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.
$TRAP: MOV 2(SP),-(SP) ;;ASSUME THE STATUS OF
BIC #20,(SP) ;;THE CALLER--DO NOT ALLOW
MOV #1$,-(SP) ;;1-BIT TRAPS
RTI ;;SET THE NEW STATUS
1$: MOV R0,-(SP) ;;SAVE R0
MOV 2(SP),R0 ;;GET TRAP ADDRESS
TST -(R0) ;;BACKUP BY 2
MOVB (R0),R0 ;;GET RIGHT BYTE OF TRAP
ASL R0 ;;POSITION FOR INDEXING
MOV $TRPAD(R0),R0 ;;INDEX TO TABLE
RTS R0 ;;GO TO ROUTINE

;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
$TRAP2: 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
; -----
$TRPAD: .WORD $TRAP2
$TYPE ;;CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
$TYPOC ;;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
$TYPOS ;;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
$TYPON ;;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
$TYPDS ;;CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
$TYPBN ;;CALL=TYPBN TRAP+6(104406) TYPE BINARY (ASCII) NUMBER
$GTSWR ;;CALL=GTSWR TRAP+7(104407) GET SOFT-SWR SETTING

```

13727									
13728	067026	065672			\$CKSWR	::CALL=CKSWR	TRAP+10(104410)	TEST FOR CHANGE IN SOFT-SWR	
13729	067030	066234			\$RDCHR	::CALL=RDCHR	TRAP+11(104411)	TTY TYPEIN CHARACTER ROUTINE	
13730	067032	066324			\$RDLIN	::CALL=RDLIN	TRAP+12(104412)	TTY TYPEIN STRING ROUTINE	
13731	067034	066632			\$RDOCT	::CALL=RDOCT	TRAP+13(104413)	READ AN OCTAL NUMBER FROM TTY	
13732	067036	063474			\$\$SAVREG	::CALL=SAVREG	TRAP+14(104414)	SAVE R0-R5 ROUTINE	
13733	067040	063532			\$\$RESREG	::CALL=RESREG	TRAP+15(104415)	RESTORE R0-R5 ROUTINE	

.SBTTL POWER DOWN AND UP ROUTINES

```

*****
:POWER DOWN ROUTINE
$PWRDN: MOV    $SILLUP, @PWRVEC    ;; SET FOR FAST UP
        MOV    @340, @PWRVEC+2  ;; PRIO:7
        RO, -(SP)                ;; PUSH R0 ON STACK
        R1, -(SP)                ;; PUSH R1 ON STACK
        R2, -(SP)                ;; PUSH R2 ON STACK
        R3, -(SP)                ;; PUSH R3 ON STACK
        R4, -(SP)                ;; PUSH R4 ON STACK
        R5, -(SP)                ;; PUSH R5 ON STACK
        @SWR, -(SP)              ;; PUSH @SWR ON STACK
        SP, $$SAVR6              ;; SAVE SP
        MOV    @SPWRUP, @PWRVEC  ;; SET UP VECTOR
        HALT
        BR     .-2                ;; HANG UP

```

```

*****
:POWER UP ROUTINE
$PWRUP: MOV    $SILLUP, @PWRVEC  ;; SET FOR FAST DOWN
        MOV    $$SAVR6, SP      ;; GET SP
        CLR    $$SAVR6         ;; WAIT LOOP FOR THE TTY
1$:     INC    $$SAVR6         ;; WAIT FOR THE INC
        BNE   1$              ;; OF WORD
        MOV   (SP)+, @SWR      ;; POP STACK INTO @SWR
        MOV   (SP)+, R5       ;; POP STACK INTO R5
        MOV   (SP)+, R4       ;; POP STACK INTO R4
        MOV   (SP)+, R3       ;; POP STACK INTO R3
        MOV   (SP)+, R2       ;; POP STACK INTO R2
        MOV   (SP)+, R1       ;; POP STACK INTO R1
        MOV   (SP)+, R0       ;; POP STACK INTO R0
        MOV    @SPWRDN, @PWRVEC ;; SET UP THE POWER DOWN VECTOR
        MOV    @340, @PWRVEC+2 ;; PRIO:7
        TYPE  $POWER          ;; REPORT THE POWER FAILURE
        RTI   $POWER          ;; POWER FAIL MESSAGE POINTER
$SILLUP: HALT                ;; THE POWER UP SEQUENCE WAS STARTED
        BR     .-2            ;; BEFORE THE POWER DOWN WAS COMPLETE
$$SAVR6: 0                    ;; PUT THE SP HERE
$POWER:  .ASCIZ <15><12>"POWER"

```

.SBTTL .EVEN APT COMMUNICATIONS ROUTINE

```

*****
$ATY1:  MOVB  #1, $FFLG        ;; TO REPORT FATAL ERROR
$ATY3:  MOVB  #1, $MFLG        ;; TO TYPE A MESSAGE
        BR    $ATYC

```

```

13783 067236 112737 000001 067464 $ATY4: MOVB #1,$FFLG ;; TO ONLY REPORT FATAL ERROR
13784 067244 $ATYC:
13785 067244 010046 MOV RO,-(SP) ;; PUSH RO ON STACK
13786 067246 010146 MOV R1,-(SP) ;; PUSH R1 ON STACK
13787 067250 105737 067462 TSTB $MFLG ;; SHOULD TYPE A MESSAGE?
13788 067254 001450 BEQ $$ ;; IF NOT: BR
13789 067256 122737 000001 001242 CMPB #APTENV,$ENV ;; OPERATING UNDER APT?
13790 067264 001031 BNE $$ ;; IF NOT: BR
13791 067266 132737 000100 001243 BITB #APTPOOL,$ENVM ;; SHOULD SPOOL MESSAGES?
13792 067274 001425 BEQ $$ ;; IF NOT: BR
13793 067276 017600 000004 MOV #4(SP),RO ;; GET MESSAGE ADDR.
13794 067302 062766 000002 000004 ADD #2,4(SP) ;; BUMP RETURN ADDR.
13795 067310 005737 001222 1$: TST $MSGTYPE ;; SEE IF DONE W/ LAST XMISSION?
13796 067314 001375 BNE $$ ;; IF NOT: WAIT
13797 067316 010037 001236 MOV RO,$MSGAD ;; PUT ADDR IN MAILBOX
13798 067322 105720 2$: TSTB (RO)+ ;; FIND END OF MESSAGE
13799 067324 001376 BNE 2$
13800 067326 163700 001236 SUB $MSGAD,RO ;; SUB START OF MESSAGE
13801 067332 006200 ASR RO ;; GET MESSAGE LNTH IN WORDS
13802 067334 010037 001240 MOV RO,$MSGLGTT ;; PUT LENGTH IN MAILBOX
13803 067340 012737 000004 001222 MOV #4,$MSGTYPE ;; TELL APT TO TAKE MSG.
13804 067346 000413 BR $$
13805 067350 017637 000004 067374 3$: MOV #4(SP),4$ ;; PUT MSG ADDR IN JSR LINKAGE
13806 067356 062766 000002 000004 ADD #2,4(SP) ;; BUMP RETURN ADDRESS
13807 067364 013746 177776 MOV 177776,-(SP) ;; PUSH 177776 ON STACK
13808 067370 004737 064316 JSR PC,$TYPE ;; CALL TYPE MACRO
13809 067374 000000 4$: .WORD 0
13810 067376 5$:
13811 067376 105737 067464 10$: TSTB $FFLG ;; SHOULD REPORT FATAL ERROR?
13812 067402 001416 BEQ 12$ ;; IF NOT: BR
13813 067404 005737 001242 TST $ENV ;; RUNNING UNDER APT?
13814 067410 001413 BEQ 12$ ;; IF NOT: BR
13815 067412 005737 001222 11$: TST $MSGTYPE ;; FINISHED LAST MESSAGE?
13816 067416 001375 BNE 11$ ;; IF NOT: WAIT
13817 067420 017637 000004 001224 MOV #4(SP),$FATAL ;; GET ERROR #
13818 067426 062766 000002 000004 ADD #2,4(SP) ;; BUMP RETURN ADDR.
13819 067434 005237 001222 INC $MSGTYPE ;; TELL APT TO TAKE ERROR
13820 067440 105037 067464 12$: CLRB $FFLG ;; CLEAR FATAL FLAG
13821 067444 105037 067463 CLRB $LFLG ;; CLEAR LOG FLAG
13822 067450 105037 067462 CLRB $MFLG ;; CLEAR MESSAGE FLAG
13823 067454 012601 MOV (SP)+,R1 ;; POP STACK INTO R1
13824 067456 012600 MOV (SP)+,RO ;; POP STACK INTO RO
13825 067460 000207 RTS PC ;; RETURN
13826 067462 000 $MFLG: .BYTE 0 ;; MESSG. FLAG
13827 067463 000 $LFLG: .BYTE 0 ;; LOG FLAG
13828 067464 000 $FFLG: .BYTE 0 ;; FATAL FLAG
13829 067466 .EVEN
13830 000200 APTSIZE=200
13831 000001 APTENV=001
13832 000100 APTPOOL=100
13833 000040 APTCSUP=040
13834
13835 .NLIST BEX

```

.SBTTL CONSOLE MESSAGES

```

067466          SCTMSG:
067466 005015 040503 047116 .ASCII <CR><LF>@CANNOT RECOVER THE BAD SECTOR FILES FROM LAST @<CR><LF>
067550 051124 041501 020113 .ASCIZ @TRACK FOR THIS DEVICE@
067576          CLSPRN: .ASCII @)@
067577          051          000 EQUALS: .ASCIZ @=@
067601          075          000 PROMPT: .ASCIZ <CR><LF>@*@
067605          015 025012 000 QSTMK: .ASCIZ @'@
067607          077          000 HELPOST:
067607          015 006412 052012 .ASCII <CR><LF><CR><LF>/THIS PROGRAM SHOULD BE HALTED BY TYPE ↑C./
067664 005015 005015 044124 .ASCII <CR><LF><CR><LF>/THE PROGRAM WILL BE HALTED AT END OF PASS/
067741          015 052012 050131 .ASCIZ <CR><LF>@TYPE HELP TEXT (Y OR N)??@
067775          UBUSQST:
067775          015 041412 040510 .ASCIZ <CR><LF>@CHANGE RMO3 UNIBUS ADDRESS OR VECTOR ADDRESS (Y OR N)<CR> ??@
070074 005015 051525 020105 CNSL00: .ASCIZ <CR><LF>@USE SAME DEVICES (Y OR N) ??@
070133          015 051012 030115 CNSL01: .ASCIZ <CR><LF>@RMO3 BUS ADDRESS (@
070160 005015 047105 051124 CNSL02: .ASCII <CR><LF>@ENTRY NOT IN I/O PAGE@
070207          015 040412 042104 .ASCIZ <CR><LF>@ADDRESS MUST BE >160000@
070241          015 051012 030115 CNSL03: .ASCIZ <CR><LF>@RMO3 VECTOR ADDRESS (@
070271          015 042412 052116 CNSL04: .ASCII <CR><LF>@ENTRY OUT OF RANGE@
070315          015 040412 042104 .ASCIZ <CR><LF>@ADDRESS MUST BE <1000@
070345          015 051012 030115 CNSL05: .ASCIZ <CR><LF>@RMO3 INTERRUPT PRIORITY (@
070401          015 042412 052116 CNSL06: .ASCIZ <CR><LF>@ENTRY OUT OF RANGE@
070426          CNSL07:
070426 005015 054524 042520 .ASCII <CR><LF>@TYPE (A) TO TEST ALL DEVICES, OR TYPE DEVICE@
070504 047040 046525 042502 .ASCII @ NUMBER(S)@
070516 005015 042524 046522 .ASCIZ <CR><LF>@TERMINATE INPUT WITH CARRIAGE RETURN@
070565          015 041412 040510 XDPMG: .ASCII <CR><LF>/CHANGE XXDP PACK,CLEAR LUC 40/
070624 005015 042522 052123 .ASCIZ <CR><LF>/RESTART THE PROGRAM/
070652 005015 047516 020124 NOTEX: .ASCIZ <CR><LF>/NOT EXIST DRIVE /

```

.EVEN

.SBTTL FUNCTION CODE TABLE

; THE FUNCTION CODE TABLE IS USED TO DEFINE STATUS CONDITIONS FOR
; EACH FUNCTION CODE. BIT USAGE IS AS FOLLOWS:

; ATA - BIT 15 IS SET IN THE ENTRY FOR A GIVEN FUNCTION CODE.
; IF ATA SHOULD BE SET WHEN THE FUNCTION CODE IS EXECUTED, OTHERWISE,
; BIT 15 IS ZERO, INDICATING THAT ATA SHOULD NOT NORMALLY BE SET.
; NOTE THAT ATA MAY BE SET WHEN A COMMAND IS EXECUTED EVEN THOUGH
; IT IS NOT EXPECTED AS A RESULT OF THE COMMAND.

; WCE - BIT 14 IS SET IN THE ENTRY FOR A GIVEN FUNCTION CODE
; IF WRITE CHECK ERRORS ARE ENABLED AS A FUNCTION OF THE COMMAND.

; OPI - BIT 13 IS SET IN THE ENTRY FOR A GIVEN FUNCTION CODE
; IF OPI ERRORS ARE ENABLED DURING THE EXECUTION OF THAT COMMAND.

; IVC - BIT 12 IS SET IN THE ENTRY FOR A GIVEN FUNCTION CODE
; IF IVC ERRORS ARE ENABLED DURING THE EXECUTION OF THAT COMMAND.

; WLE - BIT 11 IS SET IN THE ENTRY FOR A GIVEN FUNCTION CODE
; IF WRITE ERRORS ARE ENABLED DURING THE EXECUTION OF THAT COMMAND.
; THE WRITE ERRORS WHICH ARE ENABLED ARE "WLE", "WCF", "DPE", "UPE".

; IAE - BIT 10 IS SET IN THE ENTRY FOR A GIVEN FUNCTION CODE
; IF INVALID ADDRESS ERROR IS ENABLED FOR THAT COMMAND.

; ROE - BIT 09 IS SET IN THE ENTRY FOR A GIVEN FUNCTION CODE
; IF READ AND WRITE ERRORS ARE ENABLED DURING THE EXECUTION OF THE
; COMMAND. THE ERRORS ENABLED BY THIS BIT ARE "TRE", "DLT", "NEM",
; "MXF", "LBT", AND "AOE".

; BIT 08 IS NOT USED.

; HCE - BIT 07 IS SET IN THE ENTRY FOR A GIVEN FUNCTION CODE
; IF HEADER ERRORS ARE ENABLED DURING THE EXECUTION OF THAT COMMAND.
; HEADER ERRORS INCLUDE "HCRC", "HCE", "FER", AND "BSE".

; ECH - BIT 06 IS SET IN THE ENTRY FOR A GIVEN FUNCTION CODE
; IF DATA FIELD ERRORS ARE ENABLED DURING THE EXECUTION OF THAT
; COMMAND. THESE ERRORS INCLUDE "MDPE", "DCK", AND "ECH".

; BIT 05 IS NOT USED.

; BIT 04 IS NOT USED.

; BIT 03 IS NOT USED.

; BIT 02 IS NOT USED.

; BIT 01 IS NOT USED.

; ILF - BIT 00 IS SET IF THE FUNCTION CODE IS ILLEGAL.

070676

FNCOTB:

;FUNCTION CODE TABLE

070676	020000	.WORD	OPI	:NOP
070700	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (2)
070702	132000	.WORD	ATA:OPI:IVC:IAE	:SEEK
070704	130000	.WORD	ATA:OPI:IVC	:RECALIBRATE
070706	020000	.WORD	OPI	:DRIVE CLEAR
070710	030000	.WORD	OPI:IVC	:RELEASE
070712	130000	.WORD	OPI:ATA:IVC	:OFFSET
070714	130000	.WORD	OPI:ATA:IVC	:RETURN TO CENTERLINE
070716	020000	.WORD	OPI	:READ IN PRESET
070720	020000	.WORD	OPI	:PACK ACKNOWLEDGE
070722	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (24)
070724	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (26)
070726	132000	.WORD	ATA:OPI:IVC:IAE	:SEARCH
070730	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (32)
070732	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (34)
070734	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (36)
070736	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (40)
070740	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (42)
070742	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (44)
070744	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (46)
070746	073300	.WORD	WCE:OPI:IVC:IAE:AOE:HCE:ECH	:WRITE CHECK DATA
070750	073300	.WORD	WCE:OPI:IVC:IAE:AOE:HCE:ECH	:WRITE CHECK HEADER AND DATA
070752	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (54)
070754	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (56)
070756	037200	.WORD	OPI:IVC:WLE:IAE:AOE:HCE	:WRITE DATA
070760	037000	.WORD	OPI:IVC:WLE:IAE:AOE	:WRITE HEADER AND DATA
070762	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (64)
070764	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (66)
070766	033300	.WORD	OPI:IVC:IAE:AOE:HCE:ECH	:READ DATA
070770	033300	.WORD	OPI:IVC:IAE:AOE:HCE:ECH	:READ HEADER AND DATA
070772	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (74)
070774	130001	.WORD	OPI:ATA:ILF:IVC	:ILLEGAL FUNCTION (76)

CZRMED0 RM03 2 FCTNL TST 3
CZRMED.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 279
ATTENTION (ATA) TABLE

H06

SEG 0279

.SBTTL ATTENTION (ATA) TABLE

070776	001
070777	002
071000	004
071001	010
071002	020
071003	040
071004	100
071005	200

ATNTBL:	.BYTE	1.
	.BYTE	2.
	.BYTE	4.
	.BYTE	8.
	.BYTE	16.
	.BYTE	32.
	.BYTE	64.
	.BYTE	128.

.SBTTL DATA PATTERN TABLE

071006			
071006		RGDTPT:	
071006	000000	MIXED:	.WORD 0.
071010	000001		.WORD 1.
071012	000003		.WORD 3.
071014	000007		.WORD 7.
071016	000017		.WORD 15.
071020	000037		.WORD 31.
071022	000077		.WORD 63.
071024	000177		.WORD 127.
071026	000377		.WORD 255.
071030	000777		.WORD 511.
071032	001777		.WORD 1023.
071034	003777		.WORD 2047.
071036	007777		.WORD 4095.
071040	017777		.WORD 8191.
071042	037777		.WORD 16383.
071044	077777		.WORD 32767.
071046	177777	ONES:	.WORD 65535.
071050	177777		.WORD 65535.
071052	077777		.WORD 32767.
071054	037777		.WORD 16383.
071056	017777		.WORD 8191.
071060	007777		.WORD 4095.
071062	003777		.WORD 2047.
071064	001777		.WORD 1023.
071066	000777		.WORD 511.
071070	000377		.WORD 255.
071072	000177		.WORD 127.
071074	000077		.WORD 63.
071076	000037		.WORD 31.
071100	000017		.WORD 15.
071102	000007		.WORD 7.
071104	000003		.WORD 3.
071106	000001		.WORD 1.
071110	000000	ZEROS:	.WORD 0.
071112	000000		.WORD 0.
071114	000001		.WORD 1.
071116	000002		.WORD 2.
071120	000004		.WORD 4.
071122	000010		.WORD 8.
071124	000020		.WORD 16.
071126	000040		.WORD 32.
071130	000100		.WORD 64.
071132	000200		.WORD 128.
071134	000400		.WORD 256.
071136	001000		.WORD 512.
071140	002000		.WORD 1024.
071142	004000		.WORD 2048.
071144	010000		.WORD 4096.
071146	020000		.WORD 8192.
071150	040000		.WORD 16384.
071152	100000		.WORD 32768.
071154	100000		.WORD 32768.

071156	040000	.WORD	16384.
071160	020000	.WORD	8192.
071162	010000	.WORD	4096.
071164	004000	.WORD	2048.
071166	002000	.WORD	1024.
071170	001000	.WORD	512.
071172	000400	.WORD	256.
071174	000200	.WORD	128.
071176	000100	.WORD	64.
071200	000040	.WORD	32.
071202	000020	.WORD	16.
071204	000010	.WORD	8.
071206	000004	.WORD	4.
071210	000002	.WORD	2.
071212	000001	.WORD	1.
071214	000000	.WORD	0.
071216	177777	.WORD	65535.
071220	177776	.WORD	65534.
071222	177774	.WORD	65532.
071224	177770	.WORD	65528.
071226	177760	.WORD	65520.
071230	177740	.WORD	65504.
071232	177700	.WORD	65472.
071234	177600	.WORD	65408.
071236	177400	.WORD	65280.
071240	177000	.WORD	65024.
071242	176000	.WORD	64512.
071244	174000	.WORD	63488.
071246	170000	.WORD	61440.
071250	160000	.WORD	57344.
071252	140000	.WORD	49152.
071254	100000	.WORD	32768.
071256	000000	.WORD	0.
071260	000000	.WORD	0.
071262	100000	.WORD	32768.
071264	140000	.WORD	49152.
071266	160000	.WORD	57344.
071270	170000	.WORD	61440.
071272	174000	.WORD	63488.
071274	176000	.WORD	64512.
071276	177000	.WORD	65024.
071300	177400	.WORD	65280.
071302	177600	.WORD	65408.
071304	177700	.WORD	65472.
071306	177740	.WORD	65504.
071310	177760	.WORD	65520.
071312	177770	.WORD	65528.
071314	177774	.WORD	65532.
071316	177776	.WORD	65534.
071320	177777	.WORD	65535.
071322	125252	.WORD	43690.
071324	152525	.WORD	43690. /2
071326	125252	.WORD	43690.
071330	177777	.WORD	65535.
071332	177776	.WORD	65534.
071334	177775	.WORD	65533.

EARLY:

071336	177773	.WORD	65531.
071340	177767	.WORD	65527.
071342	177757	.WORD	65519.
071344	177737	.WORD	65503.
071346	177677	.WORD	65471.
071350	177577	.WORD	65407.
071352	177377	.WORD	65279.
071354	176777	.WORD	65023.
071356	175777	.WORD	64511.
071360	173777	.WORD	63487.
071362	167777	.WORD	61439.
071364	157777	.WORD	57343.
071366	137777	.WORD	49151.
071370	077777	.WORD	32767.
071372	077777	.WORD	32767.
071374	137777	.WORD	49151.
071376	157777	.WORD	57343.
071400	167777	.WORD	61439.
071402	173777	.WORD	63487.
071404	175777	.WORD	64511.
071406	176777	.WORD	65023.
071410	177377	.WORD	65279.
071412	177577	.WORD	65407.
071414	177677	.WORD	65471.
071416	177737	.WORD	65503.
071420	177757	.WORD	65519.
071422	177767	.WORD	65527.
071424	177773	.WORD	65531.
071426	177775	.WORD	65533.
071430	177776	.WORD	65534.
071432	177777	.WORD	65535.
071434			

ENRGDT:

.SBTTL ERROR MESSAGE TABLE

071434	076030	000000		EMT1:	.WORD	EMS1,0
071440	076077	076114	000000	EMT2:	.WORD	EMS2,EMS3,0
071446	076077	076157	000000	EMT3:	.WORD	EMS2,EMS4,0
071454	076222	076252	000000	EMT4:	.WORD	EMS5,EMS6,0
071462	076222	076364	000000	EMT5:	.WORD	EMS5,EMS10,0
071470	103057	100245	000000	EMT6:	.WORD	EMS16,EMS64,0
071476	101013	103104	000000	EMT7:	.WORD	EMS110,EMS170,0
071504	076317	000000		EMT10:	.WORD	EMS7,0
071510	076364	000000		EMT11:	.WORD	EMS10,0
071514	076426	076437	000000	EMT12:	.WORD	EMS11,EMS12,0
071522	076500	076511	076522	EMT13:	.WORD	EMS13,EMS14,EMS15,EMS16,0
071534	076574	100245	000000	EMT14:	.WORD	EMS17,EMS64,0
071542	076426	076657	000000	EMT15:	.WORD	EMS11,EMS21,0
071550	076426	076702	077031	EMT16:	.WORD	EMS11,EMS22,EMS27,0
071560	076426	076716	077042	EMT17:	.WORD	EMS11,EMS23,EMS30,0
071570	076426	076744	077042	EMT20:	.WORD	EMS11,EMS24,EMS30,0
071600	076426	076773	077031	EMT21:	.WORD	EMS11,EMS25,EMS30,0
071610	076426	077010	077031	EMT22:	.WORD	EMS11,EMS26,EMS30,0
071620	076426	077052	077042	EMT23:	.WORD	EMS11,EMS27,EMS30,0
071630	076426	077101	077042	EMT24:	.WORD	EMS11,EMS28,EMS30,0
071640	076426	077130	077042	EMT25:	.WORD	EMS11,EMS29,EMS30,0
071650	076426	077156	077042	EMT26:	.WORD	EMS11,EMS30,0
071660	076426	077227	077042	EMT27:	.WORD	EMS11,EMS30,0
071670	076426	077256	077042	EMT30:	.WORD	EMS11,EMS30,0
071700	076426	077305	077042	EMT31:	.WORD	EMS11,EMS30,0
071710	076426	077333	077042	EMT32:	.WORD	EMS11,EMS30,0
071720	076426	077362	077042	EMT33:	.WORD	EMS11,EMS30,0
071730	076426	077410	077042	EMT34:	.WORD	EMS11,EMS30,0
071740	076426	077437	077042	EMT35:	.WORD	EMS11,EMS30,0
071750	076426	077466	077042	EMT36:	.WORD	EMS11,EMS30,0
071760	076426	077541	077042	EMT37:	.WORD	EMS11,EMS30,0
071770	100331	076634	000000	EMT40:	.WORD	EMS66,EMS20,0
071776	100517	102225	100525	EMT41:	.WORD	EMS75,EMS141,EMS76,0
072006	102316	102326	100453	EMT42:	.WORD	EMS144,EMS145,EMS72,EMS76,0
072020	077633	077751	100525	EMT43:	.WORD	EMS47,EMS53,EMS76,0
072030	100556	077751	100525	EMT44:	.WORD	EMS77,EMS53,EMS76,0
072040	100604	077751	100525	EMT45:	.WORD	EMS100,EMS53,EMS76,0
072050	100632	077751	100525	EMT46:	.WORD	EMS101,EMS53,EMS76,0
072060	100263	100245	000000	EMT47:	.WORD	EMS65,EMS64,0
072066	076500	076522	100217	EMT50:	.WORD	EMS13,EMS15,EMS63,0
072076	076426	077604	077042	EMT51:	.WORD	EMS11,EMS46,EMS30,EMS67,0
072110	077633	077751	100401	EMT52:	.WORD	EMS47,EMS53,EMS67,EMS115,EMS140,EMS141,0
072126	077633	077751	100401	EMT53:	.WORD	EMS47,EMS53,EMS67,EMS115,EMS141,EMS164,0
072144	077662	077751	100401	EMT54:	.WORD	EMS50,EMS53,EMS67,0
072154	102253	102270	077751	EMT55:	.WORD	EMS142,EMS143,EMS53,EMS67,0
072166	077711	100453	100401	EMT56:	.WORD	EMS51,EMS72,EMS67,EMS115,EMS50,EMS70,0
072204	103057	100463	100401	EMT57:	.WORD	EMS16,EMS73,EMS67,0
072214	100034	100401	101213	EMT60:	.WORD	EMS56,EMS67,EMS115,EMS150,EMS152,EMS70,0
072232	100440	100034	100401	EMT61:	.WORD	EMS71,EMS56,EMS67,EMS115,EMS150,EMS152,EMS72,0
072252	102204	100401	101213	EMT62:	.WORD	EMS140,EMS67,EMS115,EMS47,EMS70,0
072266	000000			EMT63:	.WORD	
072270	102411	102454	100426	EMT64:	.WORD	EMS150,EMS152,EMS70,EMS115,EMS56,EMS73,EMS67,0
072310	077737	103504	103665	EMT65:	.WORD	EMS52,EMS205,EMS214,EMS206,EMS115,EMS51,EMS72,0
072330	103016	100707	100453	EMT66:	.WORD	EMS165,EMS103,EMS72,EMS124,0

072342	103016	100707	100453	EMT67:	.WORD	EMS165, EMS103, EMS72, EMS171, 0
072354	077604	076634	102711	EMT70:	.WORD	EMS46, EMS20, EMS163, 0
072364	100440	100632	102711	EMT71:	.WORD	EMS71, EMS101, EMS163, 0
072374	077633	100453	102711	EMT72:	.WORD	EMS47, EMS72, EMS163, 0
072412	077633	100453	102711	EMT73:	.WORD	EMS47, EMS72, EMS163, EMS115, EMS140, EMS141, 0
072430	100034	077751	102711	EMT74:	.WORD	EMS56, EMS53, EMS163, 0
072440	100440	100034	102711	EMT75:	.WORD	EMS71, EMS56, EMS163, EMS115, EMS150, EMS152, EMS72, 0
072460	077662	077751	102711	EMT76:	.WORD	EMS50, EMS53, EMS163, 0
072470	102253	102270	077751	EMT77:	.WORD	EMS142, EMS143, EMS53, EMS163, 0
072502	100117	102225	102711	EMT100:	.WORD	EMS75, EMS141, EMS163, EMS115, EMS47, EMS70, 0
072520	100117	102411	102711	EMT101:	.WORD	EMS75, EMS150, EMS163, EMS115, EMS56, EMS73, 0
072536	103057	100463	102711	EMT102:	.WORD	EMS167, EMS73, EMS163, 0
072546	102374	102430	100500	EMT103:	.WORD	EMS47, EMS74, EMS163, 0
072560	077711	100500	102711	EMT104:	.WORD	EMS51, EMS74, EMS163, EMS115, EMS50, EMS70, 0
072576	103016	100604	102711	EMT105:	.WORD	EMS165, EMS100, EMS163, 0
072606	103016	100556	102711	EMT106:	.WORD	EMS165, EMS77, EMS163, 0
072616	102316	102326	077751	EMT107:	.WORD	EMS145, EMS146, EMS53, EMS163, EMS115, EMS143, EMS70, 0
072636	101013	101053	101220	EMT110:	.WORD	EMS110, EMS112, EMS116, EMS111, 0
072650	101137	076157	000000	EMT111:	.WORD	EMS111, EMS113, EMS114, 0
072656	101137	076114	000000	EMT112:	.WORD	EMS111, EMS113, EMS114, 0
072664	101013	101213	101220	EMT113:	.WORD	EMS110, EMS111, EMS116, EMS117, EMS114, 0
072700	101137	101272	000000	EMT114:	.WORD	EMS111, EMS113, EMS114, 0
072706	101307	101747	101773	EMT115:	.WORD	EMS121, EMS132, EMS133, 0
072716	101352	101747	101773	EMT116:	.WORD	EMS121, EMS132, EMS133, 0
072726	101407	101747	101773	EMT117:	.WORD	EMS121, EMS132, EMS133, 0
072736	101452	101747	101773	EMT120:	.WORD	EMS121, EMS132, EMS133, 0
072746	101504	101747	101773	EMT121:	.WORD	EMS121, EMS132, EMS133, 0
072756	101547	101747	101773	EMT122:	.WORD	EMS121, EMS132, EMS133, 0
072766	102147	101747	101773	EMT123:	.WORD	EMS121, EMS132, EMS133, 0
072776	101651	101747	101773	EMT124:	.WORD	EMS121, EMS132, EMS133, 0
073006	101707	101747	101773	EMT125:	.WORD	EMS121, EMS132, EMS133, 0
073016	101307	101747	102016	EMT126:	.WORD	EMS121, EMS132, EMS133, 0
073030	101352	101747	102016	EMT127:	.WORD	EMS121, EMS132, EMS133, 0
073042	101407	101747	102016	EMT130:	.WORD	EMS121, EMS132, EMS133, 0
073054	101452	101747	102016	EMT131:	.WORD	EMS121, EMS132, EMS133, 0
073066	101504	101747	102016	EMT132:	.WORD	EMS121, EMS132, EMS133, 0
073100	101547	101747	102016	EMT133:	.WORD	EMS121, EMS132, EMS133, 0
073112	102147	101747	102016	EMT134:	.WORD	EMS121, EMS132, EMS133, 0
073124	101651	101747	102016	EMT135:	.WORD	EMS121, EMS132, EMS133, 0
073136	101707	101747	102016	EMT136:	.WORD	EMS121, EMS132, EMS133, 0
073150	101307	101747	102060	EMT137:	.WORD	EMS121, EMS132, EMS133, 0
073160	101407	101747	102060	EMT140:	.WORD	EMS121, EMS132, EMS133, 0
073170	101307	101747	102122	EMT141:	.WORD	EMS121, EMS132, EMS133, 0
073200	102147	101747	102122	EMT142:	.WORD	EMS121, EMS132, EMS133, 0
073210	101452	101747	102122	EMT143:	.WORD	EMS121, EMS132, EMS133, 0
073220	101504	101747	102122	EMT144:	.WORD	EMS121, EMS132, EMS133, 0
073230	101547	101747	102122	EMT145:	.WORD	EMS121, EMS132, EMS133, 0
073240	101707	101747	102122	EMT146:	.WORD	EMS121, EMS132, EMS133, 0
073250	102656	101747	102122	EMT147:	.WORD	EMS121, EMS132, EMS133, 0
073260	101651	101747	102122	EMT150:	.WORD	EMS121, EMS132, EMS133, 0
073270	102204	101213	102225	EMT151:	.WORD	EMS140, EMS115, EMS141, EMS70, 0
073302	102253	101213	102270	EMT152:	.WORD	EMS142, EMS115, EMS143, EMS72, 0
073314	102316	102326	102355	EMT153:	.WORD	EMS144, EMS145, EMS146, EMS115, EMS143, EMS72, 0
073332	102316	102326	076634	EMT154:	.WORD	EMS144, EMS145, EMS20, EMS115, EMS143, EMS70, 0
073350	102411	102454	102522	EMT155:	.WORD	EMS150, EMS152, EMS154, EMS153, 0
073362	102374	102430	102522	EMT156:	.WORD	EMS147, EMS151, EMS154, EMS155, 0

073374	102374	102430	102556	EMT157:	.WORD	EMS147, EMS151, EMS156, EMS157, 0
073406	102626	102556	102611	EMT160:	.WORD	EMS161, EMS156, EMS160, 0
073416	076773	077031	102556	EMT161:	.WORD	EMS229, EMS27, EMS156, EMS160, 0
073430	077010	077031	102556	EMT162:	.WORD	EMS26, EMS27, EMS156, EMS160, 0
073442	077633	102711	102204	EMT163:	.WORD	MS47, EMS163, EMS140, 0
073452	077633	076634	000000	EMT164:	.WORD	MS47, EMS20, 0
073460	100034	076634	000000	EMT165:	.WORD	MS56, EMS20, 0
073466	077604	076634	000000	EMT166:	.WORD	MS46, EMS20, 0
073474	104205	076634	000000	EMT167:	.WORD	MS224, EMS20, 0
073502	103057	102522	103032	EMT170:	.WORD	MS167, EMS154, EMS166, 0
073512	103057	102522	102574	EMT171:	.WORD	MS167, EMS154, EMS157, 0
073522	103057	102522	102536	EMT172:	.WORD	MS167, EMS154, EMS155, 0
073532	103133	101747	101773	EMT173:	.WORD	MS171, EMS132, EMS133, 0
073542	103133	101747	102016	EMT174:	.WORD	MS171, EMS132, EMS134, EMS123, 0
073554	101137	103306	000000	EMT175:	.WORD	MS112, EMS177, 0
073562	103330	103345	000000	EMT176:	.WORD	MS200, EMS201, 0
073570	103443	102225	077042	EMT177:	.WORD	MS203, EMS141, EMS30, EMS202, 0
073602	103443	077711	077042	EMT200:	.WORD	MS203, EMS51, EMS30, EMS202, 0
073614	103443	103456	077042	EMT201:	.WORD	MS203, EMS204, EMS30, EMS202, 0
073626	103443	077662	077042	EMT202:	.WORD	MS203, EMS50, EMS30, EMS202, 0
073640	103443	102270	077042	EMT203:	.WORD	MS203, EMS143, EMS30, EMS202, 0
073652	102411	100500	103413	EMT204:	.WORD	MS150, EMS74, EMS202, EMS115, EMS152, EMS72, 0
073670	077662	100707	100453	EMT205:	.WORD	MS50, EMS103, EMS72, 0
073700	100716	100463	103413	EMT206:	.WORD	MS104, EMS73, EMS202, 0
073710	100517	102225	101213	EMT207:	.WORD	MS75, EMS141, EMS115, EMS140, 0
073722	100517	102411	000000	EMT210:	.WORD	MS75, EMS150, 0
073730	077711	100453	101213	EMT211:	.WORD	MS51, EMS72, EMS115, EMS50, EMS70, 0
073744	102253	077751	101213	EMT212:	.WORD	MS142, EMS53, EMS115, EMS143, EMS72, 0
073760	077662	100707	077751	EMT213:	.WORD	MS50, EMS103, EMS53, 0
073770	077737	103504	103032	EMT214:	.WORD	MS52, EMS205, EMS166, EMS206, EMS115, EMS51, EMS72, 0
074010	077737	101250	100034	EMT215:	.WORD	MS52, EMS117, EMS56, EMS163, 0
074022	100034	077042	100245	EMT216:	.WORD	MS56, EMS30, EMS64, 0
074032	077604	077042	100245	EMT217:	.WORD	MS46, EMS30, EMS64, 0
074042	077052	077042	100245	EMT220:	.WORD	MS31, EMS30, EMS64, 0
074052	077633	077042	100245	EMT221:	.WORD	MS47, EMS30, EMS64, 0
074062	077737	101250	100556	EMT222:	.WORD	MS52, EMS117, EMS77, 0
074072	077737	101250	100217	EMT223:	.WORD	MS52, EMS117, EMS63, 0
074102	000000			EMT224:	.WORD	
074104	000000			EMT225:	.WORD	
074106	000000			EMT226:	.WORD	
074110	000000			EMT227:	.WORD	
074112	000000			EMT230:	.WORD	
074114	000000			EMT231:	.WORD	
074116	000000			EMT232:	.WORD	
074120	000000			EMT233:	.WORD	
074122	000000			EMT234:	.WORD	
074124	000000			EMT235:	.WORD	
074126	000000			EMT236:	.WORD	
074130	000000			EMT237:	.WORD	
074132	000000			EMT240:	.WORD	
074134	000000			EMT241:	.WORD	
074136	000000			EMT242:	.WORD	
074140	000000			EMT243:	.WORD	
074142	000000			EMT244:	.WORD	
074144	000000			EMT245:	.WORD	
074146	103057	101747	103543	EMT246:	.WORD	EMS167, EMS132, EMS207, 0

075332	101220	101250	077156	EMT337:	.WORD	EMS116, EMS117, EMS34, 0
075342	077156	077751	077763	EMT340:	.WORD	EMS34, EMS53, EMS54, EMS111, 0
075354	100005	077156	000000	EMT341:	.WORD	EMS55, EMS34, 0
075362	077737	101250	100034	EMT342:	.WORD	EMS55, EMS117, EMS56, EMS57, 0
075374	077737	101250	077541	EMT343:	.WORD	EMS55, EMS117, EMS45, EMS57, 0
075406	077737	101250	077466	EMT344:	.WORD	EMS55, EMS117, EMS44, EMS57, 0
075420	077737	101250	104027	EMT345:	.WORD	EMS55, EMS117, EMS221, 0
075430	100774	076634	101213	EMT346:	.WORD	EMS106, EMS20, EMS115, EMS223, EMS72, 0
075444	077737	103504	104077	EMT347:	.WORD	EMS106, EMS205, EMS222, EMS206, 0
075456	100517	102411	100660	EMT350:	.WORD	EMS73, EMS150, EMS102, EMS115, EMS56, EMS73, 0
075474	103057	100463	100660	EMT351:	.WORD	EMS167, EMS73, EMS102, 0
075504	103703	000000		EMT352:	.WORD	EMS215, 0
075510	103754	103443	103724	EMT353:	.WORD	EMS217, EMS203, EMS216, 0
075520	104027	076634	000000	EMT354:	.WORD	EMS221, EMS20, 0

075526	104257	105063	105140	EHT1:	.WORD	EH1,STSH1,STSH2,STSH4,0
075540	105063	105140	105265	EHT2:	.WORD	STSH1,STSH2,STSH4,0
075550	104276	000000		EHT110:	.WORD	EH110,0
075554	104305	000000		EHT111:	.WORD	EH111,0
075560	104324	000000		EHT114:	.WORD	EH114,0
075564	104353	105063	105140	EHT223:	.WORD	EH223,STSH1,STSH2,STSH4,0
075576	104401	105063	105140	EHT256:	.WORD	EH256,STSH1,STSH2,STSH4,0
075610	104456	105063	105140	EHT336:	.WORD	EH336,STSH1,STSH2,STSH4,0
075622	104515	105063	105140	EHT337:	.WORD	EH337,STSH1,STSH2,STSH4,0
075634	104654	105063	105140	EHT344:	.WORD	EH344,STSH1,STSH2,STSH4,0
075646	105014	000000		EHT353:	.WORD	EH353,0

E07

CZRMEBO RMO3/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 289
ERROR MESSAGE TABLE

SEQ 0289

075652	105324	105420	105436	EDT1:	.WORD	ED1,STSD1,STSD2,STSD4
075662	105420	105436	105470	EDT2:	.WORD	STSD1,STSD2,STSD4
075670	105332			EDT110:	.WORD	ED110
075672	105336			EDT111:	.WORD	ED111
075674	105344			EDT114:	.WORD	ED114
075676	105354	105420	105436	EDT223:	.WORD	ED223,STSD1,STSD2,STSD4
075706	105364	105420	105436	EDT336:	.WORD	ED336,STSD1,STSD2,STSD4
075716	105376	105420	105436	EDT337:	.WORD	ED337,STSD1,STSD2,STSD4
075726	105376	105420	105436	EDT344:	.WORD	ED337,STSD1,STSD2,STSD4,0
075740	105410			EDT353:	.WORD	ED353

075742	105503	105521	105521	EFT1:	.WORD	EF111,STSF,STSF,STSF
075752	105521	105521	105521	EFT2:	.WORD	STSF,STSF,STSF
075760	105502			EFT110:	.WORD	EF110
075762	105503			EFT111:	.WORD	EF111
075764	105505			EFT114:	.WORD	EF114
075766	105505	105521	105521	EFT223:	.WORD	EF114,STSF,STSF,STSF
075776	105510	105521	105521	EFT336:	.WORD	EF336,STSF,STSF,STSF
076006	105510	105521	105521	EFT337:	.WORD	EF336,STSF,STSF,STSF
076016	105510	105521	105521	EFT344:	.WORD	EF336,STSF,STSF,STSF
076026	105505			EFT353:	.WORD	EF114

.SBTTL ERROR MESSAGE STRINGS

076030	051127	047117	020107	EMS1:	.ASCIZ	WRONG UNIT SELECTED (RMCS2, BITS 0-2) a
076077	104	053105	041511	EMS2:	.ASCIZ	DEVICE WENT a
076114	047125	053101	044501	EMS3:	.ASCIZ	UNAVAILABLE "DVA" (RMCS1, BIT 11) a
076157	116	047117	054105	EMS4:	.ASCIZ	NONEXISTENT "MED" (RMCS2, BIT 12) a
076222	047503	046515	047101	EMS5:	.ASCIZ	COMMAND NOT COMPLETED a
076252	047503	052116	047522	EMS6:	.ASCIZ	CONTROLLER NOT READY (RMCS1, BIT 7) a
076317	104	044522	042526	EMS7:	.ASCIZ	DRIVE NOT READY "DRY" (RMDS, BIT 7) a
076364	047507	047040	052117	EMS10:	.ASCIZ	GO NOT RESET "GO" (RMCS1, BIT 0) a
076426	047111	040526	044514	EMS11:	.ASCIZ	INVALID a
076437	106	047125	052103	EMS12:	.ASCIZ	FUNCTION CODE (RMCS1, BITS 1-5) a
076500	040515	051523	052502	EMS13:	.ASCIZ	MASSBUS a
076511	103	047117	051124	EMS14:	.ASCIZ	CONTROL a
076522	052502	020123	040520	EMS15:	.ASCIZ	BUS PARITY ERROR a
076544	046442	050103	021105	EMS16:	.ASCIZ	"MCPE" (RMCS1, BIT 13) a
076574	051124	047101	043123	EMS17:	.ASCIZ	TRANSFER ERROR (RMCS1, BIT 14) a
076634	044123	052517	042114	EMS20:	.ASCIZ	SHOULD NOT BE SET a
076657	127	051117	020104	EMS21:	.ASCIZ	WORD COUNT (RMWC) a
076702	052502	020123	051050	EMS22:	.ASCIZ	BUS (RMB) a
076716	046042	052102	020042	EMS23:	.ASCIZ	"LBT" (RMDS, BIT 10) a
076744	040442	042517	020042	EMS24:	.ASCIZ	"AOE" (RMER1, BIT 09) a
076773	104	051511	020113	EMS25:	.ASCIZ	DISK (RMDA) a
077010	054503	044514	042116	EMS26:	.ASCIZ	CYLINDER (RMDC) a
077031	101	042104	042522	EMS27:	.ASCIZ	ADDRESS a
077042	052123	052101	051525	EMS30:	.ASCIZ	STATUS a
077052	053442	042514	020042	EMS31:	.ASCIZ	"WLE" (RMER1, BIT 11) a
077101	042	050125	021105	EMS32:	.ASCIZ	"UPE" (RMCS2, BIT 13) a
077130	053442	043103	020042	EMS33:	.ASCIZ	"WCF" (RMER1, BIT 5) a
077156	051127	052111	020105	EMS34:	.ASCIZ	WRITE CHECK ERROR-"WCE" (RMCS2, BIT 14) a
077227	042	042115	042520	EMS35:	.ASCIZ	"MOPE" (RMCS2, BIT 8) a
077256	042042	045503	020042	EMS36:	.ASCIZ	"DCK" (RMER1, BIT 15) a
077305	042	041505	021110	EMS37:	.ASCIZ	"ECH" (RMER1, BIT 6) a
077333	042	046104	021124	EMS40:	.ASCIZ	"DLT" (RMCS2, BIT 15) a
077362	046442	043130	020042	EMS41:	.ASCIZ	"MXF" (RMCS2, BIT 9) a
077410	042042	042524	020042	EMS42:	.ASCIZ	"DTE" (RMER1, BIT 12) a
077437	042	041510	041522	EMS43:	.ASCIZ	"HCRC" (RMER1, BIT 8) a
077466	042510	042101	051105	EMS44:	.ASCIZ	HEADER COMPARE ERROR "HCE" (RMER1, BIT 7) a
077541	106	051117	040515	EMS45:	.ASCIZ	FORMAT ERROR "FER" (RMER1, BIT 4) a
077604	044442	042501	020042	EMS46:	.ASCIZ	"IAE" (RMER1, BIT 10) a
077633	042	050117	021111	EMS47:	.ASCIZ	"OPT" (RMER1, BIT 13) a
077662	051442	044513	020042	EMS50:	.ASCIZ	"SKI" (RMER2, BIT 14) a
077711	042	044520	021120	EMS51:	.ASCIZ	"PIP" (RMDS, BIT 13) a
077737	124	042510	051040	EMS52:	.ASCIZ	THE RM03 a
077751	104	052105	041505	EMS53:	.ASCIZ	DETECTED a
077763	101	020124	047101	EMS54:	.ASCIZ	AT AN UNEXPECTED a
100005	111	041516	051117	EMS55:	.ASCIZ	INCORRECT DATA DURING a
100034	047111	040526	044514	EMS56:	.ASCIZ	INVALID COMMAND ERROR "IVC" (RMER2, BIT 12) a
100111	104	051125	047111	EMS57:	.ASCIZ	DURING DATA TRANSFER a
100137	104	052101	020101	EMS60:	.ASCIZ	DATA READ a
100152	047504	051505	047040	EMS61:	.ASCIZ	DOES NOT COMPARE WITH a
100201	104	052101	020101	EMS62:	.ASCIZ	DATA WRITTEN a
100217	042	040520	021122	EMS63:	.ASCIZ	"PAR" (RMER1, BIT 3) a
100245	111	020123	047111	EMS64:	.ASCIZ	IS INCORRECT a
100263	103	046517	047520	EMS65:	.ASCIZ	COMPOSITE ERROR "ERR" (RMDS, BIT 14) a
100331	104	052101	020101	EMS66:	.ASCIZ	DATA PARITY ERROR "DPE" (RMER2, BIT 3) a

100401	104	051125	047111	EMS67:	.ASCIZ	@DURING SEEK COMMAND @
100426	051511	051040	051505	EMS70:	.ASCIZ	@IS RESET @
100440	051105	047522	042516	EMS71:	.ASCIZ	@ERRONEOUS @
100453	111	020123	042523	EMS72:	.ASCIZ	@IS SET @
100463	104	042111	047040	EMS73:	.ASCIZ	@DID NOT SET @
100500	044504	020104	047516	EMS74:	.ASCIZ	@DID NOT RESET @
100517	114	051517	020124	EMS75:	.ASCIZ	@LST @
100525	104	051125	047111	EMS76:	.ASCIZ	@DURING PACK ACK COMMAND @
100556	051042	051115	020042	EMS77:	.ASCIZ	@"RMR" (RMR1, BIT 2) @
100604	044442	051114	020042	EMS100:	.ASCIZ	@"ILR" (RMR1, BIT 1) @
100632	044442	043114	020042	EMS101:	.ASCIZ	@"ILF" (RMR1, BIT 0) @
100660	052504	044522	043516	EMS102:	.ASCIZ	@DURING SEARCH COMMAND @
100707	105	051122	051117	EMS103:	.ASCIZ	@ERROR @
100716	046042	041502	020042	EMS104:	.ASCIZ	@"LBC" (RMR2, BIT 10) @
100745	042	051514	021103	EMS105:	.ASCIZ	@"LSC" (RMR2, BIT 11) @
100774	042510	042101	051105	EMS106:	.ASCIZ	@HEADER ERRORS @
101013	102	051525	052040	EMS110:	.ASCIZ	@BUS TIMEOUT (04 TRAP) @
101042	042101	051104	051505	EMS111:	.ASCIZ	@ADDRESS @
101053	127	042510	020116	EMS112:	.ASCIZ	@WHEN READING/WRITING RH REGISTERS @
101115	101	020124	044124	EMS113:	.ASCIZ	@AT THE FOLLOWING @
101137	124	042510	051440	EMS114:	.ASCIZ	@THE SELECTED DEVICE IS @
101167	116	047117	054105	EMS115:	.ASCIZ	@NONEXISTENT DEVICE @
101213	040	006455	000012	EMS116:	.ASCIZ	@ -@<CR><LF>
101220	044124	020105	040515	EMS117:	.ASCIZ	@THE MASSBUS CONTROLLER @
101250	040506	046111	042105	EMS118:	.ASCIZ	@FAILED TO DETECT @
101272	047516	020124	047101	EMS120:	.ASCIZ	@NOT AN RM03 @
101307	103	047117	051124	EMS121:	.ASCIZ	@CONTROL STATUS REGISTER 1, RMCS1, @
101352	052502	020123	042101	EMS122:	.ASCIZ	@BUS ADDRESS REGISTER, RMB@, @
101407	103	047117	051124	EMS123:	.ASCIZ	@CONTROL STATUS REGISTER 2, RMCS2, @
101452	051105	047522	020122	EMS124:	.ASCIZ	@ERROR REGISTER 1, RMR1 @
101504	052101	042524	052116	EMS125:	.ASCIZ	@ATTENTION SUMMARY REGISTER, RMA@, @
101547	115	044501	052116	EMS126:	.ASCIZ	@MAINTENANCE REGISTER #1, RMMR #1, @
101612	041505	020103	047520	EMS127:	.ASCIZ	@ECC POSITION REGISTER, RMEC1, @
101651	105	041503	050040	EMS130:	.ASCIZ	@ECC PATTERN REGISTER, RMEC2 @
101707	115	044501	052116	EMS131:	.ASCIZ	@MAINTENANCE REGISTER 2, RMMR2, @
101747	116	052117	044440	EMS132:	.ASCIZ	@NOT INITIALIZED BY @
101773	125	044516	052502	EMS133:	.ASCIZ	@UNIBUS INITIALIZE @
102016	047503	052116	047522	EMS134:	.ASCIZ	@CONTROLLER CLEAR, I.E. BIT 5 OF @
102060	044122	030461	042440	EMS135:	.ASCIZ	@RH11 ERROR CLEAR (RMCS1, BIT 14) @
102122	051104	053111	020105	EMS136:	.ASCIZ	@DRIVE CLEAR COMMAND @
102147	104	044522	042526	EMS137:	.ASCIZ	@DRIVE STATUS REGISTER, RMD@ @
102204	042515	044504	046525	EMS140:	.ASCIZ	@MEDIUM OFF LINE @
102225	042	047515	021114	EMS141:	.ASCIZ	@"MOL" (RMD@, BIT 12) @
102253	104	044522	042526	EMS142:	.ASCIZ	@DRIVE FAULT @
102270	042042	041526	020042	EMS143:	.ASCIZ	@"DVC" (RMR2, BIT 7) @
102316	047125	040523	042506	EMS144:	.ASCIZ	@UNSAFE @
102326	052442	051516	020042	EMS145:	.ASCIZ	@"UNS" (RMR1, BIT 14) @
102355	123	047510	046125	EMS146:	.ASCIZ	@SHOULD BE SET @
102374	043117	051506	052105	EMS147:	.ASCIZ	@OFFSET MODE @
102411	040	047526	052514	EMS150:	.ASCIZ	@ VOLUME VALID @
102430	047442	021115	024040	EMS151:	.ASCIZ	@"OM" (RMD@, BIT 0) @
102454	053042	021126	024040	EMS152:	.ASCIZ	@"VV" (RMD@, BIT 6) @
102500	040520	045503	040440	EMS153:	.ASCIZ	@PACK ACK COMMAND @
102522	047516	020124	042523	EMS154:	.ASCIZ	@NOT SET BY @
102536	043117	051506	052105	EMS155:	.ASCIZ	@OFFSET COMMAND @
102556	047516	020124	042522	EMS156:	.ASCIZ	@NOT RESET BY @

102574	052122	020103	047503	EMS157:	.ASCIZ	@RTC COMMAND @
102611	122	050111	041440	EMS160:	.ASCIZ	@RIP COMMAND @
102626	043117	051506	052105	EMS161:	.ASCIZ	@OFFSET REGISTER (RMOF) @
102656	051105	047522	020122	EMS162:	.ASCIZ	@ERROR REGISTER #2, RMER2, @
102711	104	051125	047111	EMS163:	.ASCIZ	@DURING RECALIBRATE @
102735	111	020123	047111	EMS164:	.ASCIZ	@IS INTERMITTENT OR DRIVE DIDNT DROP ON @
103004	054503	044514	042116			@CYLINDER @
103016	047125	054105	042520	EMS165:	.ASCIZ	@UNEXPECTED @
103032	042522	040503	044514	EMS166:	.ASCIZ	@RECALIBRATE COMMAND @
103057	042	052101	021101	EMS167:	.ASCIZ	@"ATA" (RMDS, BIT15) @
103104	044127	047105	051040	EMS170:	.ASCIZ	@WHEN READING REGISTER @
103133	105	051122	051117	EMS171:	.ASCIZ	@ERROR REGISTER #2, RMER2, @
103166	047516	051116	041505	EMS172:	.ASCIZ	@NONRECOVERABLE @
103206	042522	047503	042526	EMS173:	.ASCIZ	@RECOVERABLE @
103223	104	052101	020101	EMS174:	.ASCIZ	@DATA @
103231	104	051125	047111	EMS175:	.ASCIZ	@DURING WRITE COMMAND @
103257	042	050117	021105	EMS176:	.ASCIZ	@"OPE" (RMER2, BIT 13) @
103306	047111	053440	044522	EMS177:	.ASCIZ	@IN WRITE PROTECT @
103330	040503	020116	047516	EMS200:	.ASCIZ	@CAN NOT SET @
103345	104	040511	047107	EMS201:	.ASCIZ	@DIAGNOSTIC MODE "DMD" (RMMR1, BIT 0) @
103413	104	051125	047111	EMS202:	.ASCIZ	@DURING DIAGNOSTIC MODE @
103443	111	041516	051117	EMS203:	.ASCIZ	@INCORRECT @
103456	053442	046122	020042	EMS204:	.ASCIZ	@"WRL" (RMDS, BIT 11) @
103504	054105	041505	052125	EMS205:	.ASCIZ	@EXECUTED @
103516	044527	044124	041440	EMS206:	.ASCIZ	@WITH COMP ERROR SET @
103543	042	047507	020042	EMS207:	.ASCIZ	@"GO" (RMCS1, BIT 0) @
103570	051127	052111	047111	EMS210:	.ASCIZ	@WRITING @
103601	127	051501	051040	EMS211:	.ASCIZ	@WAS RESET BY @
103617	120	047522	051107	EMS212:	.ASCIZ	@PROGRAM INTERRUPT @
103642	040527	020123	047516	EMS213:	.ASCIZ	@WAS NOT GENERATED @
103665	123	042505	020113	EMS214:	.ASCIZ	@SEEK COMMAND @
103703	120	047522	051107	EMS215:	.ASCIZ	@PROGRAM TIMEOUT @
103724	052504	044522	043516	EMS215:	.ASCIZ	@DURING LOOK AHEAD TEST @
103754	047514	045517	040440	EMS217:	.ASCIZ	@LOOK AHEAD REGISTER, RMLA, @
104007	123	040505	041522	EMS220:	.ASCIZ	@SEARCH COMMAND @
104027	102	042101	051440	EMS221:	.ASCIZ	@BAD SECTOR ERROR "BSE" (RMER2, BIT 15) @
104077	101	042040	052101	EMS222:	.ASCIZ	@A DATA TRANSFER COMMAND @
104130	042510	042101	051105	EMS223:	.ASCIZ	@HEADER COMPARE INHIBIT "HCI" (RMOF, BIT 10) @
104205	116	047117	054105	EMS224:	.ASCIZ	@NONEXISTENT MEMORY "NEM" (RMCS2, BIT 11) @

104257	105	050130	052103	EH1:	.ASCIZ	RECEVD	RECEVD		
104276	052502	040523	051104	EH110:	.ASCIZ	RECEVD	RECEVD		
104305	040	046522	051503	EH111:	.ASCIZ	RMCS2	RMCS1		
104324	042522	042503	042126	EH114:	.ASCIZ	RECEVD	SNGPRT	DULPRT	
104353	105	050130	052103	EH223:	.ASCIZ	RECEVD	RECEVD	DATA	
104401	105	050130	052103	EH256:	.ASCII	RECEVD	RECEVD	RGSTR	<CR><LF>
104430	052123	052101	051525		.ASCIZ	STATUS	STATUS	INDEX	
104456	042107	042101	051522	EH336:	.ASCIZ	GDADRS	GDADRS	BDADRS	BDDATA
104515	122	041515	031123	EH337:	.ASCII	RMCS2	STATUS	FAILING	DATA
104555	137	057537	057537		.ASCII	*****	*****	*****	*****
104615	105	050130	052103		.ASCIZ	RECEVD	RECEVD	BIT	ADRESS
104654	046522	051105	020061	EH344:	.ASCII	RMER1	STATUS	HEADER	FAILING
104715	137	057537	057537		.ASCII	*****	*****	WORD	BIT
104754	054105	041520	042124		.ASCIZ	RECEVD	RECEVD	NUMBER	POSITION
105014	054105	041520	042124	EH353:	.ASCII	RECEVD	RECEVD	<CR><LF>	
105034	051040	046115	020101		.ASCIZ	RMLA	RMLA	RMOF	
105063	040	046522	051503	STSH1:	.ASCII	RMCS1	RMCS2	RMDS	RMER1
105130	020040	051040	040515		.ASCIZ	RMAS			RMER2
105140	051040	053515	020103	STSH2:	.ASCII	RMWC	RMBA	RMDA	RMOF
105205	040	020040	051040		.ASCIZ	RMEC1	RMEC2		RMDC
105227	040	046522	040504	STSH3:	.ASCIZ	RMDA	RMDC	RMOF	RMLA
105265	040	046522	051115	STSH4:	.ASCIZ	RMMR1	RMMR2	RMDT	RMSN

	105324			.EVEN		
105324	001140	001142	000000	ED1:	.WORD	\$GDDAT,\$BDDAT,0
105332	001276	000000		ED110:	.WORD	\$BASE,0
105336	001174	001176	000000	ED111:	.WORD	\$TMP0,\$TMP1,0
105344	001356	001176	001200	ED114:	.WORD	RMDTI,\$TMP1,\$TMP2,0
105354	001140	001142	001174	ED223:	.WORD	\$GDDAT,\$BDDAT,\$TMP0,0
105364	001134	001140	001136	ED336:	.WORD	\$GDADR,\$GDDAT,\$BDADR,\$BDDAT,0
105376	001140	001142	001174	ED337:	.WORD	\$GDDAT,\$BDDAT,\$TMP0,\$TMP1,0
105410	001140	001142	001432	ED353:	.WORD	\$GDDAT,\$BDDAT,RMOFO,0
105420	001330	001340	001342	STSD1:	.WORD	RMCS1I, RMCS2I, RMDSI, RMER1I, RMER2I, RMASI, 0
105436	001332	001334	001336	STSD2:	.WORD	RMWCI, RMBAI, RMDAI, RMOFI, RMDCI, RMECI
105452	001376	000000			.WORD	RMEC2I, 0
105456	001336	001364	001362	STSD3:	.WORD	RMDAI, RMDCI, RMOFI, RMLAI, 0
105470	001354	001370	001356	STSD4:	.WORD	RMMR1I, RMMR2I, RMDTI, RMSNI, 0

105502	000			EF110:	.BYTE	0
105503	000	000		EF111:	.BYTE	0,0
105505	000	000	000	EF114:	.BYTE	0,0,0
105510	000	000	000	EF336:	.BYTE	0,0,0,0
105514	000	000	000	EF337:	.BYTE	0,0,0,0,0
105521	000	000	000	STSF:	.BYTE	0,0,0,0,0,0,0

MO7

CZARMEBO RMO3/2 FCTNL TST 3
CZARMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 297
ERROR MESSAGE STRINGS

SEQ 0297

105530 000402
106534 177777
106536 177777
106540 000402
107544 177777
107546 177777

.EVEN
MFGFIL: .BLKW 258.
.WORD -1
USRFIL: .WORD -1
.BLKW 258.
.WORD -1
.WORD -1
.EVEN

107550
107550 000402
110554 000402

BUFFER:
BUFONE: .BLKW 258.
BUFTWO: .BLKW 258.

107550 107550
107550 005015

.HELP: = BUFFER

107552	046011	051511	020124
107572	057411	057537	057537
107612	030524	041411	047117
107645	124	004462	042504
107677	124	004463	051104
107723	124	004464	051127
107751	124	004465	051127
110006	033124	053411	044522
110060	033524	053411	044522
110105	124	030061	053411
110142	030524	004461	051127
110214	030524	004462	051127
110265	124	031461	053411
110326	030524	004464	051127
110400	030524	004465	051127
110455	124	033061	053411
110513	124	033461	053411
110543	124	030062	053411
110601	124	030462	053411
110633	124	031062	053411
110663	124	031462	053411
110727	124	032062	053411
111012	031124	004465	051127
111054	031124	004466	051127
111116	031124	004467	051127
111152	005015		
111154	047411	042520	040522
111212	057411	057537	057537
111250	005015		
111252	053523	052111	044103
111270	026455	026455	026455
111326	020040	032461	004411
111353	040	030440	004464
111377	040	030440	004463
111435	040	030440	004462
111445	040	030440	004461
111477	040	030440	004460
111524	020040	034440	004411
111551	040	020040	004470
111611	040	020040	004467
111626	020040	033040	004411
111642	020040	032440	004411
111656	020040	032040	004411
111672	020040	031440	004411
111705	040	020040	004462

```

.ASCII <CR><LF>
;.ASCII @CAUTION: PARTS 2 AND 3 OF THE FUNCTIONAL TEST LEAVE BAD@<CR><LF>
;.ASCII @HEADERS ON THE PACK SO BE SURE TO FORMAT THE PACK WHEN@<CR><LF>
;.ASCII @DONE.@<CR><LF>
.ASCII @
@ LIST OF TESTS@<CR><LF>
.ASCII @
@*****@<CR><LF>
.ASCII @T1 CONTROLLER ACCESS TEST@<CR><LF>
.ASCII @T2 DEVICE AVAILABLE TEST@<CR><LF>
.ASCII @T3 DRIVE TYPE TEST@<CR><LF>
.ASCII @T4 WRITE, READ ZEROS@<CR><LF>
.ASCII @T5 WRITE, WRITE CHECK ZEROS@<CR><LF>
.ASCII @T6 WRITE, WRITE CHECK ZEROS W/ WCE ERROR@<CR><LF>
.ASCII @T7 WRITE, READ ONES@<CR><LF>
.ASCII @T10 WRITE, WRITE CHECK ONES@<CR><LF>
.ASCII @T11 WRITE, WRITE CHECK ONES W/ WCE ERROR@<CR><LF>
.ASCII @T12 WRITE, WRITE CHECK MULTIPLE SECTORS@<CR><LF>
.ASCII @T13 WRITE, READ W/ IMPLIED SEEK@<CR><LF>
.ASCII @T14 WRITE, WRITE CHECK W/ HEAD SWITCHING@<CR><LF>
.ASCII @T15 WRITE, WRITE CHECK W/ MID-TRANSFER SEEK@<CR><LF>
.ASCII @T16 WRITE, READ W/ WCE ERROR@<CR><LF>
.ASCII @T17 WRITE, READ W/ HCI@<CR><LF>
.ASCII @T20 WRITE, READ W/ IVC ERROR@<CR><LF>
.ASCII @T21 WRITE, READ W/ ABORT@<CR><LF>
.ASCII @T22 WRITE, READ W/ BAI@<CR><LF>
.ASCII @T23 WRITE, READ EACH CURRENT LEVEL@<CR><LF>
.ASCII @T24 WRITE, WRITE CHECK W/ CURRENT LEVEL SWITCHING@<CR><LF>
.ASCII @T25 WRITE, READ EARLY PEAK SHIFT@<CR><LF>
.ASCII @T26 WRITE, READ MIXED PEAK SHIFT@<CR><LF>
.ASCII @T27 WRITE, READ EACH TRACK@<CR><LF>
.ASCII <CR><LF>
@ OPERATIONAL SWITCH SETTINGS@<CR><LF>
@*****@<CR><LF>
.ASCII <CR><LF>
@SWITCH
@-----@<CR><LF>
@ 15 HALT ON ERROR@<CR><LF>
@ 14 LOOP ON TEST@<CR><LF>
@ 13 INHIBIT ERROR TYPEOUTS@<CR><LF>
@ 12 @<CR><LF>
@ 11 INHIBIT ITERATIONS@<CR><LF>
@ 10 BELL ON ERROR@<CR><LF>
@ 9 LOOP ON ERROR@<CR><LF>
@ 8 LOOP ON TEST IN SWR<7:0>@<CR><LF>
@ 7 TN128@<CR><LF>
@ 6 TN64@<CR><LF>
@ 5 TN32@<CR><LF>
@ 4 TN16@<CR><LF>
@ 3 TN8@<CR><LF>
@ 2 TN4@<CR><LF>

```

CZRMEBO RMO3/2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 299
ERROR MESSAGE STRINGS

SEQ 0299

111720	020040	030440	004411	.ASCII	2	1	TN20<CR><LF>
111733	040	020040	004460	.ASCII	2	0	TN10<CR><LF>
111746	005015			.ASCII			<CR><LF>
111750	005015	000		.ASCIZ			<CR><LF>
	000001			.END			

3752#	3761#	3780#	3802#	3810#	3813#	3816#	3824#	3831#	3840#	3847#	3874#	3892#		
3914#	3922#	3925#	3928#	3936#	3956#	3980#	3988#	3991#	3994#	4002#	4009#	4018#		
4037#	4058#	4066#	4069#	4072#	4080#	4087#	4096#	4122#	4140#	4162#	4170#	4173#		
4176#	4184#	4204#	4228#	4236#	4239#	4242#	4250#	4257#	4266#	4288#	4309#	4317#		
4320#	4323#	4331#	4340#	4381#	4414#	4432#	4454#	4462#	4465#	4468#	4476#	4496#		
4520#	4528#	4531#	4534#	4542#	4549#	4558#	4577#	4599#	4607#	4610#	4613#	4621#		
4628#	4637#	4644#	4671#	4689#	4711#	4719#	4722#	4725#	4733#	4753#	4777#	4785#		
4788#	4791#	4799#	4806#	4815#	4834#	4855#	4863#	4866#	4869#	4877#	4884#	4893#		
4919#	4937#	4959#	4967#	4970#	4973#	4981#	5001#	5025#	5033#	5036#	5039#	5047#		
5054#	5063#	5085#	5106#	5114#	5117#	5120#	5128#	5137#	5178#	5211#	5229#	5251#		
5259#	5262#	5265#	5273#	5300#	5322#	5330#	5333#	5336#	5344#	5351#	5360#	5379#		
5400#	5408#	5411#	5414#	5422#	5429#	5438#	5464#	5482#	5504#	5512#	5515#	5518#		
5526#	5549#	5560#	5568#	5569#	5570#	5576#	5600#	5608#	5611#	5619#	5626#	5635#		
5657#	5668#	5676#	5677#	5678#	5684#	5707#	5716#	5719#	5727#	5734#	5743#	5750#		
5780#	5798#	5820#	5828#	5831#	5834#	5842#	5869#	5891#	5899#	5902#	5905#	5913#		
5920#	5929#	5948#	5969#	5977#	5980#	5983#	5991#	5998#	6007#	6033#	6051#	6073#		
6081#	6084#	6087#	6095#	6122#	6144#	6152#	6155#	6158#	6166#	6173#	6182#	6201#		
6222#	6230#	6233#	6236#	6244#	6251#	6260#	6290#	6308#	6337#	6345#	6348#	6351#		
6359#	6379#	6403#	6411#	6414#	6417#	6425#	6446#	6465#	6482#	6494#	6512#	6535#		
6543#	6546#	6549#	6557#	6577#	6596#	6613#	6625#	6654#	6699#	6717#	6746#	6754#		
6757#	6760#	6768#	6788#	6812#	6820#	6823#	6826#	6834#	6854#	6863#	6881#	6903#		
6911#	6914#	6917#	6925#	6945#	6954#	6961#	6990#	7035#	7050#	7092#	7100#	7103#		
7106#	7114#	7123#	7137#	7156#	7197#	7205#	7208#	7211#	7219#	7228#	7242#	7274#		
7288#	7322#	7323#	7329#	7343#	7346#	7354#	7362#	7381#	7413#	7415#	7423#	7438#		
7441#	7449#	7456#	7482#	7500#	7522#	7530#	7533#	7536#	7544#	7564#	7593#	7601#		
7604#	7607#	7615#	7622#	7631#	7650#	7675#	7683#	7686#	7689#	7697#	7704#	7713#		
7720#	7747#	7765#	7787#	7795#	7798#	7801#	7809#	7829#	7853#	7861#	7864#	7867#		
7875#	7882#	7891#	7910#	7933#	7941#	7944#	7947#	7955#	7962#	7971#	7978#	8011#		
8029#	8051#	8059#	8062#	8065#	8073#	8093#	8117#	8125#	8128#	8131#	8139#	8146#		
8155#	8174#	8195#	8203#	8206#	8209#	8217#	8224#	8233#	8259#	8277#	8299#	8307#		
8310#	8313#	8321#	8341#	8365#	8373#	8376#	8379#	8387#	8394#	8403#	8422#	8444#		
8452#	8455#	8458#	8466#	8473#	8482#	8489#	8516#	8534#	8556#	8564#	8567#	8570#		
8578#	8598#	8622#	8630#	8633#	8636#	8644#	8651#	8660#	8679#	8701#	8709#	8712#		
8715#	8723#	8730#	8739#	8746#	8773#	8791#	8813#	8821#	8824#	8827#	8835#	8855#		
8879#	8887#	8890#	8893#	8901#	8908#	8917#	8936#	8958#	8966#	8972#	8980#			
8987#	8996#	9003#												
1440#	6647	6984	9747*	9776	9822*	9823	9825*	9826*	9827	9829*	9874			
1439#	6646	6983	9746*	9775	9830*	9831	9833*	9873						
1316	1329													
ATA =	100000	10609	10610	10612	11344	11345	11380	11383	11875	11876	11915	11918	13835	
ATESTN=	000000	1316	1320											
ATNMSK=	000377	1075#	12072											
ATNTBL	070776	3422	3440	13835#										
AUNIT =	000000	1316	1323											
AUSMR =	000000	1316	1330											
AVECT1=	120254	1209#	1316	1355										
AVECT2=	000000	1316	1356											
A16 =	000400	1158#												
A17 =	001000	1157#												
BACK =	000000	3617#	3622	3635#	3639	3657#	3661	3665#	3668#	3671#	3675	3679#	3683	3699#
		3704	3723#	3727	3731#	3734#	3737#	3741	3745#	3749	3752#	3756	3761#	3765
		3780#	3785	3802#	3806	3810#	3813#	3816#	3820	3824#	3828	3831#	3835	3840#
		3844	3847#	3853	3874#	3879	3892#	3896	3914#	3918	3922#	3925#	3928#	3932
		3936#	3940	3956#	3961	3980#	3984	3988#	3991#	3994#	3998	4002#	4006	4009#
		4013	4018#	4022	4037#	4042	4058#	4062	4066#	4069#	4072#	4076	4080#	4084
		4087#	4091	4096#	4100	4122#	4127	4140#	4144	4162#	4166	4170#	4173#	4176#

ASNDA 001500
 ASNDC 001476
 ASWREG= 000000
 ATA = 100000
 ATESTN= 000000
 ATNMSK= 000377
 ATNTBL 070776
 AUNIT = 000000
 AUSMR = 000000
 AVECT1= 120254
 AVECT2= 000000
 A16 = 000400
 A17 = 001000
 BACK = 000000

4180	4184	4188	4204	4209	4228	4232	4236	4239	4242	4246	4250	4254
4257	4261	4266	4270	4288	4293	4309	4313	4317	4320	4323	4327	4331
4335	4340	4344	4381	4385	4414	4419	4432	4436	4454	4458	4462	4465
4468	4472	4476	4480	4496	4501	4520	4524	4528	4531	4534	4538	4542
4546	4549	4553	4558	4562	4577	4582	4599	4603	4607	4610	4613	4617
4621	4625	4628	4632	4637	4641	4644	4650	4671	4676	4689	4693	4711
4715	4719	4722	4725	4729	4733	4737	4753	4758	4777	4781	4785	4788
4791	4795	4799	4803	4806	4810	4815	4819	4834	4839	4855	4859	4863
4866	4869	4873	4877	4881	4884	4888	4893	4897	4919	4924	4937	4941
4959	4963	4967	4970	4973	4977	4981	4985	5001	5006	5025	5029	5033
5036	5039	5043	5047	5051	5054	5058	5063	5067	5085	5090	5106	5110
5114	5117	5120	5124	5128	5132	5137	5141	5178	5182	5211	5216	5229
5233	5251	5255	5259	5262	5265	5269	5273	5277	5300	5305	5322	5326
5330	5333	5336	5340	5344	5348	5351	5355	5360	5364	5379	5384	5400
5404	5408	5411	5414	5418	5422	5426	5429	5433	5438	5442	5464	5469
5482	5486	5504	5508	5512	5515	5518	5522	5526	5530	5549	5554	5560
5564	5568	5569	5570	5574	5576	5580	5600	5604	5608	5611	5615	5619
5623	5626	5630	5635	5639	5657	5662	5668	5672	5676	5677	5678	5682
5684	5688	5707	5711	5716	5719	5723	5727	5731	5734	5738	5743	5747
5750	5756	5780	5785	5798	5802	5820	5824	5828	5831	5834	5838	5842
5846	5869	5874	5891	5895	5899	5902	5905	5909	5913	5917	5920	5924
5929	5933	5948	5953	5969	5973	5977	5980	5983	5987	5991	5995	5998
6002	6007	6011	6033	6038	6051	6055	6073	6077	6081	6084	6087	6091
6095	6099	6122	6127	6144	6148	6152	6155	6158	6162	6166	6170	6173
6177	6182	6186	6201	6206	6222	6226	6230	6233	6236	6240	6244	6248
6251	6255	6260	6264	6290	6295	6308	6312	6337	6341	6345	6348	6351
6355	6359	6363	6379	6384	6403	6407	6411	6414	6417	6421	6425	6429
6446	6450	6465	6469	6482	6486	6494	6498	6512	6517	6535	6539	6543
6546	6549	6553	6557	6561	6577	6581	6596	6600	6613	6617	6625	6629
6654	6659	6699	6704	6717	6721	6746	6750	6754	6757	6760	6764	6768
6772	6788	6793	6812	6816	6820	6823	6826	6830	6834	6838	6854	6858
6863	6867	6881	6886	6903	6907	6911	6914	6917	6921	6925	6929	6945
6949	6954	6958	6961	6967	6990	6995	7035	7050	7055	7092	7096	7100
7103	7106	7110	7114	7118	7123	7127	7137	7141	7156	7161	7197	7201
7205	7208	7211	7215	7219	7223	7228	7232	7242	7246	7274	7288	7293
7322	7323	7327	7329	7333	7343	7346	7350	7354	7358	7362	7366	7381
7386	7413	7415	7419	7423	7427	7438	7441	7445	7449	7453	7456	7460
7482	7487	7500	7504	7522	7526	7530	7533	7536	7540	7544	7548	7564
7569	7593	7597	7601	7604	7607	7611	7615	7619	7622	7626	7631	7635
7650	7655	7675	7679	7683	7686	7689	7693	7697	7701	7704	7708	7713
7717	7720	7726	7747	7752	7765	7769	7787	7791	7795	7798	7801	7805
7809	7813	7829	7834	7853	7857	7861	7864	7867	7871	7875	7879	7882
7886	7891	7895	7910	7915	7933	7937	7941	7944	7947	7951	7955	7959
7962	7966	7971	7975	7978	7984	8011	8016	8029	8033	8051	8055	8059
8062	8065	8069	8073	8077	8093	8098	8117	8121	8125	8128	8131	8135
8139	8143	8146	8150	8155	8159	8174	8179	8195	8199	8203	8206	8209
8213	8217	8221	8224	8228	8233	8237	8259	8264	8277	8281	8299	8303
8307	8310	8313	8317	8321	8325	8341	8346	8365	8369	8373	8376	8379
8383	8387	8391	8394	8398	8403	8407	8422	8427	8444	8448	8452	8455
8458	8462	8466	8470	8473	8477	8482	8486	8489	8495	8516	8521	8534
8538	8556	8560	8564	8567	8570	8574	8578	8582	8598	8603	8622	8626
8630	8633	8636	8640	8644	8648	8651	8655	8660	8664	8679	8684	8701
8705	8709	8712	8715	8719	8723	8727	8730	8734	8739	8743	8746	8752
8773	8778	8791	8795	8813	8817	8821	8824	8827	8831	8835	8839	8855
8860	8879	8883	8887	8890	8893	8897	8901	8905	8908	8912	8917	8921
8936	8941	8958	8962	8966	8969	8972	8976	8980	8984	8987	8991	8996

F08

CZRMEBC RMO3-2 FCTNL TST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 304
CROSS REFERENCE TABLE -- USER SYMBOLS

SEG 0303

BADSC*	040734	9000	9003*	9009	4432	4689	4937	5229	5482	5798	6051	6308	6717	7500
		3635	3892	4140	8534	8791	9463*							
		7765	8029	8277										
BAI	= 000010	1177*	7578	10998										
BB00	= 000001	1095*												
BB01	= 000002	1094*												
BB02	= 000004	1093*												
BB03	= 000010	1092*												
BB04	= 000020	1091*												
BB05	= 000040	1090*												
BB06	= 000100	1089*												
BB07	= 000200	1088*												
BB08	= 000400	1087*												
BB09	= 001000	1086*												
BIT0	= 000001	884*	3269	3590	3592	10285								
BIT00	= 000001	874*	884	910	959	978	997	1035	1054	1095	1181	1197		
BIT01	= 000002	873*	883	909	958	996	1034	1053	1094	1180	1196			
BIT02	= 000004	872*	882	908	957	995	1033	1052	1093	1179	1195			
BIT03	= 000010	871*	881	907	956	994	1032	1051	1092	1107	1177	1194		
BIT04	= 000020	870*	880	906	955	993	1050	1091	1176					
BIT05	= 000040	869*	879	905	992	1031	1049	1090	1175					
BIT06	= 000100	868*	878	977	991	1013	1030	1048	1089	1160	1174	1193		
BIT07	= 000200	867*	877	976	990	1012	1029	1047	1071	1088	1106	1159	1173	
BIT08	= 000400	866*	876	954	975	989	1011	1028	1046	1087	1158	1171	13268	
BIT09	= 001000	865*	875	953	974	988	1010	1027	1045	1086	1157	1170	13286	13378
BIT1	= 000002	883*	9753	9911	11015									
BIT10	= 002000	864*	952	973	987	1009	1026	1044	1070	1085	1105	1156	1169	1192
		13355												
BIT11	= 004000	863*	904	972	986	1025	1043	1061	1069	1084	1104	1168	1191	9381
		13293												
BIT12	= 010000	862*	971	985	1024	1042	1068	1083	1103	1167	1190	9331		
BIT13	= 020000	861*	970	984	1023	1041	1060	1082	1102	1154	1166	1189	6286	6640
		13362												
BIT14	= 040000	860*	969	983	1022	1040	1059	1081	1101	1112	1153	1165	1188	9296
		13254												
BIT15	= 100000	859*	968	982	1021	1039	1058	1080	1100	1111	1152	1164	1187	9283
		9996	10050											
BIT2	= 000004	882*	10285											
BIT3	= 000010	881*	9409	12457	12544									
BIT4	= 000020	880*	9359	12563										
BIT5	= 000040	879*												
BIT6	= 000100	878*	9308											
BIT7	= 000200	877*	3264	10291										
BIT8	= 000400	876*												
BIT9	= 001000	875*												
BOTADR	037756	9147*	9165*	9168	9183	9228*								
BOTFLG	037760	9133*	9175*	9178	9181*	9229*								
BPTVEC	= 000014	891*												
BSE	= 100000	1100*	6479	6490	6610	6621	6846	6849	6937	6940	9659	10902		
BUFFER	107550	9478*	9727	13835*										
BUFONE	107550	3628	3711	3848	3885	3968	4133	4217	4425	4508	4645	4682	4765	4930
		5014	5222	5286	5475	5589	5751	5791	5855	6044	6109	6301	6319	6321*
		6323*	6391	6648	6710	6728	6730*	6732*	6802	6962	6985	7029	7082	7268
		7493	7581	7721	7758	7842	7979	8022	8106	8270	8354	8490	8527	8611
		8747	8784	8867	9004	13835*								
BUFTWO	110554	3791	3849	4285*	4351	4366	4368	4389*	4588	4646	5082*	5148	5163	5165

DULPRT=	024024	1064#	3588	3592	3595									
DVA =	004000	904#	3277	3548	10136	10209	10374	10377	11168	11486	12034	12035		
OVC =	000200	1106#	11315	11570	11822	11860	11863	11996	12369	12373	12376	12421	12795	12800
EARLY	071322	12803#												
EBL =	020000	9283#	13835#											
ECH =	000100	1041#												
ECI =	004000	991#	999	9652	9986	10923	10938	10956	10962	12560	13835			
ECRC =	001000	1069#	9988	10958	12557									
EDY1	075652	1045#												
		1476	1483	1490	1497	1504	1511	1525	1532	1539	1546	1553	1560	1567
		1574	1581	1588	1595	1602	1609	1616	1623	1630	1637	1644	1651	1658
		1665	1672	1679	1686	1693	1700	1707	1714	1721	1728	1735	1742	1749
		1756	1763	1771	1778	1785	1792	1799	1807	1815	1823	1837	1844	1851
		1858	1865	1872	1879	1887	1894	1901	1908	1915	1922	1929	1936	1943
		1950	1957	1964	1971	2013	2020	2027	2034	2041	2048	2055	2062	2069
		2076	2083	2090	2097	2104	2111	2118	2125	2132	2139	2146	2153	2160
		2167	2174	2181	2188	2195	2202	2209	2216	2223	2230	2237	2244	2251
		2258	2265	2272	2279	2286	2293	2300	2307	2314	2321	2328	2335	2342
		2364	2371	2378	2385	2392	2399	2406	2413	2420	2427	2434	2441	2448
		2469	2476	2483	2490	2497	2504	2511	2518	2525	2532	2539	2546	2553
		2714	2721	2728	2735	2742	2749	2756	2763	2770	2777	2784	2791	2798
		2810	2817	2824	2831	2838	2845	2852	2859	2866	2873	2880	2887	2894
		2905	2912	2919	2926	2933	2940	2947	2954	2961	2968	2975	2982	2989
		2996	3003	3010	3017	3024	3031	3038	3045	3052	3059	3066	3073	3080
		3136	13835#											
		1978	13835#											
		1985	1992	13835#										
		2006	13835#											
		2455	2462	2658	2665	13835#								
		2504	2693	13835#										
		2280	3038	3052	3059	13835#								
		3045	13835#											
		3080	13835#											
		3129	13835#											
		105324	13835#											
		105332	13835#											
		105336	13835#											
		105344	13835#											
		105354	13835#											
		105364	13835#											
		105376	13835#											
		105410	13835#											
		000020	1050#											
		075742	1477	1484	1491	1498	1505	1512	1526	1533	1540	1547	1554	1561
			1575	1582	1589	1596	1603	1610	1617	1624	1631	1638	1645	1652
			1666	1673	1680	1687	1694	1701	1708	1715	1722	1729	1736	1743
			1757	1764	1772	1779	1786	1793	1800	1808	1816	1824	1838	1845
			1859	1866	1873	1880	1888	1895	1902	1909	1916	1923	1930	1937
			1951	1958	1965	1972	2014	2021	2028	2035	2042	2049	2056	2063
			2077	2084	2091	2098	2105	2112	2119	2126	2133	2140	2147	2154
			2168	2175	2182	2189	2196	2203	2210	2217	2224	2231	2238	2245
			2259	2266	2273	2280	2287	2294	2301	2308	2315	2322	2329	2336
			2365	2372	2379	2386	2393	2400	2407	2414	2421	2428	2435	2442
			2470	2477	2484	2491	2498	2505	2512	2519	2526	2533	2540	2547
			2715	2722	2729	2736	2743	2750	2757	2764	2771	2778	2785	2792
			2811	2820	2828	2836	2843	2850	2857	2864	2871	2878	2885	2892

EMS101	100632	13835#
EMS102	100660	13835#
EMS103	100707	13835#
EMS104	100716	13835#
EMS105	100745	13835#
EMS106	100774	13835#
EMS11	076426	13835#
EMS110	101013	13835#
EMS111	101042	13835#
EMS112	101053	13835#
EMS113	101137	13835#
EMS114	101167	13835#
EMS115	101213	13835#
EMS116	101220	13835#
EMS117	101250	13835#
EMS118	076437	13835#
EMS120	101272	13835#
EMS121	101307	13835#
EMS122	101352	13835#
EMS123	101407	13835#
EMS124	101452	13835#
EMS125	101504	13835#
EMS126	101547	13835#
EMS127	101612	13835#
EMS13	076500	13835#
EMS130	101651	13835#
EMS131	101707	13835#
EMS132	101747	13835#
EMS133	101773	13835#
EMS134	102016	13835#
EMS135	102060	13835#
EMS136	102122	13835#
EMS137	102147	13835#
EMS14	076511	13835#
EMS140	102204	13835#
EMS141	102225	13835#
EMS142	102253	13835#
EMS143	102270	13835#
EMS144	102316	13835#
EMS145	102326	13835#
EMS146	102355	13835#
EMS147	102374	13835#
EMS15	076522	13835#
EMS150	102411	13835#
EMS151	102430	13835#
EMS152	102454	13835#
EMS153	102500	13835#
EMS154	102522	13835#
EMS155	102536	13835#
EMS156	102556	13835#
EMS157	102574	13835#
EMS16	076544	13835#
EMS160	102611	13835#
EMS161	102626	13835#
EMS162	102656	13835#
EMS163	102711	13835#

EMS164	102735	13835#
EMS165	103016	13835#
EMS166	103032	13835#
EMS167	103057	13835#
EMS17	076574	13835#
EMS170	103104	13835#
EMS171	103133	13835#
EMS172	103166	13835#
EMS173	103206	13835#
EMS174	103223	13835#
EMS175	103231	13835#
EMS176	103257	13835#
EMS177	103306	13835#
EMS2	076077	13835#
EMS20	076634	13835#
EMS200	103330	13835#
EMS201	103345	13835#
EMS202	103413	13835#
EMS203	103443	13835#
EMS204	103456	13835#
EMS205	103504	13835#
EMS206	103516	13835#
EMS207	103543	13835#
EMS21	076657	13835#
EMS210	103570	13835#
EMS211	103601	13835#
EMS212	103617	13835#
EMS213	103642	13835#
EMS214	103665	13835#
EMS215	103703	13835#
EMS216	103724	13835#
EMS217	103754	13835#
EMS22	076702	13835#
EMS220	104007	13835#
EMS221	104027	13835#
EMS222	104077	13835#
EMS223	104130	13835#
EMS224	104205	13835#
EMS23	07671E	13835#
EMS24	076744	13835#
EMS25	076773	13835#
EMS26	077010	13835#
EMS27	077031	13835#
EMS3	076114	13835#
EMS30	077042	13835#
EMS31	077052	13835#
EMS32	077101	13835#
EMS33	077130	13835#
EMS34	077156	13835#
EMS35	077227	13835#
EMS36	077256	13835#
EMS37	077305	13835#
EMS4	076157	13835#
EMS40	077333	13835#
EMS41	077362	13835#
EMS42	077410	13835#

EMS43	077437	13835#		
EMS44	077466	13835#		
EMS45	077541	13835#		
EMS46	077604	13835#		
EMS47	077633	13835#		
EMS5	076222	13835#		
EMS50	077662	13835#		
EMS51	077711	13835#		
EMS52	077737	13835#		
EMS53	077751	13835#		
EMS54	077763	13835#		
EMS55	100005	13835#		
EMS56	100034	13835#		
EMS57	100111	13835#		
EMS6	076252	13835#		
EMS60	100137	13835#		
EMS61	100152	13835#		
EMS62	100201	13835#		
EMS63	100217	13835#		
EMS64	100245	13835#		
EMS65	100263	13835#		
EMS66	100331	13835#		
EMS67	100401	13835#		
EMS7	076317	13835#		
EMS70	100426	13835#		
EMS71	100440	13835#		
EMS72	100453	13835#		
EMS73	100463	13835#		
EMS74	100500	13835#		
EMS75	100517	13835#		
EMS76	100525	13835#		
EMS77	100556	13835#		
EMTVEC=	000030	894#	3195*	3196*
EMT1	071434	1474	13835#	
EMT10	071504	1523	13835#	
EMT100	072502	1920	13835#	
EMT101	072520	1927	13835#	
EMT102	072536	1934	13835#	
EMT103	072546	1941	13835#	
EMT104	072560	1948	13835#	
EMT105	072576	1955	13835#	
EMT106	072606	1962	13835#	
EMT107	072616	1969	13835#	
EMT11	071510	1530	13835#	
EMT110	072636	1976	13835#	
EMT111	072650	1983	13835#	
EMT112	072656	1990	13835#	
EMT113	072664	1997	13835#	
EMT114	072700	2004	13835#	
EMT115	072706	2011	13835#	
EMT116	072716	2018	13835#	
EMT117	072726	2025	13835#	
EMT12	071514	1537	13835#	
EMT120	072736	2032	13835#	
EMT121	072746	2039	13835#	
EMT122	072756	2046	13835#	

EMT123	072766	2053	13835#
EMT124	072776	2060	13835#
EMT125	073006	2067	13835#
EMT126	073016	2074	13835#
EMT127	073030	2081	13835#
EMT128	071522	1544	13835#
EMT129	073042	2088	13835#
EMT130	073054	2095	13835#
EMT131	073066	2102	13835#
EMT132	073100	2109	13835#
EMT133	073112	2116	13835#
EMT134	073124	2123	13835#
EMT135	073136	2130	13835#
EMT136	073150	2137	13835#
EMT137	071534	1551	13835#
EMT138	073160	2144	13835#
EMT139	073170	2151	13835#
EMT140	073200	2158	13835#
EMT141	073210	2165	13835#
EMT142	073220	2172	13835#
EMT143	073230	2179	13835#
EMT144	073240	2186	13835#
EMT145	073250	2193	13835#
EMT146	071542	1558	13835#
EMT147	073260	2200	13835#
EMT148	073270	2207	13835#
EMT149	073302	2214	13835#
EMT150	073314	2221	13835#
EMT151	073332	2228	13835#
EMT152	073350	2235	13835#
EMT153	073362	2242	13835#
EMT154	073374	2249	13835#
EMT155	071550	1565	13835#
EMT156	073406	2256	13835#
EMT157	073416	2263	13835#
EMT158	073430	2270	13835#
EMT159	073442	13835#	13835#
EMT160	073452	2285	13835#
EMT161	073460	2292	13835#
EMT162	073466	2299	13835#
EMT163	073474	2306	13835#
EMT164	071560	1572	13835#
EMT165	073502	13835#	13835#
EMT166	073512	2320	13835#
EMT167	073522	2327	13835#
EMT168	073532	2334	13835#
EMT169	073542	2341	13835#
EMT170	073554	2348	13835#
EMT171	073562	2355	13835#
EMT172	073570	2362	13835#
EMT173	071440	1481	13835#
EMT174	071570	1579	13835#
EMT175	073602	2369	13835#
EMT176	073614	2376	13835#
EMT177	073626	2383	13835#
EMT178	073640	2390	13835#

EMT204	073652	2397	13835#
EMT205	073670	2404	13835#
EMT206	073700	2411	13835#
EMT207	073710	2418	13835#
EMT21	071600	1586	13835#
EMT210	073722	2425	13835#
EMT211	073730	2432	13835#
EMT212	073744	2439	13835#
EMT213	073760	2446	13835#
EMT214	073770	2453	13835#
EMT215	074010	2460	13835#
EMT216	074022	2467	13835#
EMT217	074032	2474	13835#
EMT22	071610	1593	13835#
EMT220	074042	2481	13835#
EMT221	074052	2488	13835#
EMT222	074062	2495	13835#
EMT223	074072	2502	13835#
EMT224	074102	13835#	
EMT225	074104	13835#	
EMT226	074106	13835#	
EMT227	074110	13835#	
EMT23	071620	1600	13835#
EMT230	074112	13835#	
EMT231	074114	13835#	
EMT232	074116	13835#	
EMT233	074120	13835#	
EMT234	074122	13835#	
EMT235	074124	13835#	
EMT236	074126	13835#	
EMT237	074130	13835#	
EMT24	071630	1607	13835#
EMT240	074132	13835#	
EMT241	074134	13835#	
EMT242	074136	13835#	
EMT243	074140	13835#	
EMT244	074142	13835#	
EMT245	074144	13835#	
EMT246	074146	2635	13835#
EMT247	074156	2642	13835#
EMT25	071640	1614	13835#
EMT250	074170	2649	13835#
EMT251	074204	2656	13835#
EMT252	074212	2663	13835#
EMT253	074220	2670	13835#
EMT254	074236	2677	13835#
EMT255	074246	2684	13835#
EMT256	074256	2691	13835#
EMT257	074266	2698	13835#
EMT26	071650	1621	13835#
EMT260	074300	2705	13835#
EMT261	074312	2712	13835#
EMT262	074332	2719	13835#
EMT263	074342	2726	13835#
EMT264	074352	2733	13835#
EMT265	074372	2740	13835#

EMT266	074412	2747	13835#
EMT267	074424	2755	13835#
EMT27	071660	1628	13835#
EMT270	074442	2762	13835#
EMT271	074456	2770	13835#
EMT272	074474	2777	13835#
EMT273	074512	2784	13835#
EMT274	074530	2792	13835#
EMT275	074546	2800	13835#
EMT276	074560	2808	13835#
EMT277	074574	2817	13835#
EMT3	071446	1488	13835#
EMT30	071670	1635	13835#
EMT300	074612	2825	13835#
EMT301	074632	2833	13835#
EMT302	074654	2840	13835#
EMT303	074666	2847	13835#
EMT304	074700	2854	13835#
EMT30	074712	2861	13835#
EMT306	074724	2868	13835#
EMT307	074734	2875	13835#
EMT31	071700	1642	13835#
EMT310	074754	2882	13835#
EMT311	074766	2889	13835#
EMT312	075000	2896	13835#
EMT313	075012	2903	13835#
EMT314	075032	2910	13835#
EMT315	075044	2917	13835#
EMT316	075056	2924	13835#
EMT317	075070	2931	13835#
EMT32	071710	1649	13835#
EMT320	075100	2938	13835#
EMT321	075110	2945	13835#
EMT322	075120	2952	13835#
EMT323	075126	2959	13835#
EMT324	075136	2966	13835#
EMT325	075150	2973	13835#
EMT326	075162	2980	13835#
EMT327	075200	2987	13835#
EMT33	071720	1656	13835#
EMT330	075212	2994	13835#
EMT331	075222	3001	13835#
EMT332	075234	3008	13835#
EMT333	075246	3015	13835#
EMT334	075254	3022	13835#
EMT335	075302	3029	13835#
EMT336	075322	2278	3036
EMT337	075332	3043	13835#
EMT34	071730	1663	13835#
EMT340	075342	3050	13835#
EMT341	075354	3057	13835#
EMT342	075362	3064	13835#
EMT343	075374	3071	13835#
EMT344	075406	3078	13835#
EMT345	075420	3085	13835#
EMT346	075430	3092	13835#

13835#

RMEC2 = 000046	1131#	10078												
RMEC2I 001376	1398#	10016	10077	11547	12111	13835								
RMEC20 001446	1425#													
RMER1 = 000014	1119#	7297	7389	10452										
RMER1I 001344	1385#	6443	6453	6462	6473	6574	6584	6593	6604	6841	6843	6932	6934	
	9651	9984	9986	10444	10459	10461	10551	10627	10654	10697	10712	10755	10832	
	10858	10873	10888	10941	10960	10963	11051	11066	11081	11092	11220	11227	11254	
	11328	11331	11352	11524	11648	11652	11654	11655	11666	11668	11669	11680	11682	
	11683	11694	11696	11697	11708	11710	11711	11754	11759	11763	11765	11775	11777	
	11779	11791	11793	11795	11808	11810	11812	11885	11962	11966	11968	11970	11980	
	11982	11984	11994	11998	12000	12058	12180	12184	12186	12223	12225	12227	12268	
	12271	12273	12275	12285	12287	12289	12299	12301	12303	12326	12416	12419	12423	
	12425	12434	12436	12438	12448	12450	12451	12485	12489	12491	12493	12502	12504	
	12506	12515	12517	12519	12528	12530	12532	12550	12552	12554	12560	12616	12618	
	12620	12667	12748	12859	13835									
	1412#	7299*	7391*	10448										
RMER10 001414	1129#													
RMER2 = 000042	1396#	6479	6489	6610	6620	6846	6848	6937	6939	7121	7131	7132	7226	
RMER2I 001372	7235	7237	9653	9658	10446	10553	10671	10727	10903	11222	11273	11276	11287	
	11315	11318	11367	11396	11569	11761	11822	11829	11831	11833	11846	11848	11850	
	11860	11862	11864	11901	11948	11996	12120	12182	12251	12254	12256	12340	12344	
	12369	12373	12375	12377	12387	12391	12393	12403	12405	12407	12421	12603	12605	
	12607	12650	12683	12764	12780	12794	12800	12802	12804	12814	12816	12818	12862	
	13835													
RMER20 001442	1423#													
RMLA = 000020	1121#													
RMLAI 001350	1387#	13835												
RMLAO 001420	1414#													
RMMR1 = 000024	1122#	7059	7068	7165	7174									
RMMR1I 001354	1389#	11534	12086	13835										
RMMR10 001424	1416#	7061*	7070*	7167*	7176*									
RMMR2 = 000040	1128#													
RMMR2I 001370	1395#	11556	12098	13835										
RMMR20 001440	1422#													
RMOF = 000032	1125#	3651	3717	3795	3908	3974	4051	4156	4222	4302	4448	4514	4592	
	4705	4771	4848	4953	5019	5099	5245	5316	5393	5498	5594	5700	5814	
	5885	5962	6067	6138	6215	6331	6397	6528	6664	6740	6806	6896	7000	
	7086	7190	7314	7405	7516	7587	7668	7781	7847	7926	8045	8111	8188	
	8293	8359	8437	8550	8616	8694	8807	8873	8951	9495				
RMOFI 001362	1392#	9988	9990	10848	10958	12481	12557	13835						
RMOFO 001432	1419#	3630*	3887*	4135*	4427*	4684*	4932*	5224*	5477*	5793*	6046*	6303*	6650*	
	6712#	6800*	6987*	7031*	7270*	7495*	7760*	8024*	8272*	8529*	8786*	9488*	9917*	
	11247	12323	13835											
RMR = 000004	995#	11648	11680	11684	11962	11980	11983	12268	12299	12302				
RMSN = 000030	1124#													
RMSNI 001360	1391#	13835												
RMSNO 001430	1418#													
RMWC = 000002	1201#	3495*	3496	3650	3719	3798	3907	3976	4054	4155	4224	4305	4447	
	4516	4595	4704	4773	4851	4952	5021	5102	5244	5318	5396	5497	5596	
	5703	5813	5887	5965	6066	6140	6218	6330	6399	6531	6667	6739	6808	
	6899	7003	7088	7193	7316	7408	7515	7589	7671	7780	7849	7929	8044	
	8113	8191	8292	8361	8440	8549	8618	8697	8806	8875	8954	9494		
RMWCI 001332	1380#	10033	10982	10994	11012	11076	13835							
RMWCO 001402	1407#	3629*	3712*	3886*	3969*	4134*	4216*	4426*	4509*	4683*	4766*	4931*	5013*	
	5223#	5285*	5476*	5588*	5792*	5854*	6045*	6108*	6302*	6392*	6524*	6649*	6711*	
	6996#	7030*	7269*	7494*	7580*	7759*	7841*	7922*	8023*	8105*	8271*	8353*	8528*	

CALCLR	757#	3529	3573												
CALGET	757#														
CALPUT	757#	6670	7006												
CALSUB	757#	3617	3635	3657	3665	3668	3671	3679	3699	3723	3731	3734	3737	3745	3752
	3761	3780	3802	3810	3813	3816	3824	3831	3840	3847	3874	3892	3914	3922	3925
	3928	3936	3956	3980	3988	3991	3994	4002	4009	4018	4037	4058	4066	4069	4072
	4080	4087	4096	4122	4140	4162	4170	4173	4176	4184	4204	4228	4236	4239	4242
	4250	4257	4257	4288	4309	4317	4320	4323	4331	4340	4381	4414	4432	4454	4462
	4465	4468	4476	4496	4520	4528	4531	4534	4542	4549	4558	4577	4599	4607	4610
	4613	4621	4628	4637	4644	4671	4689	4711	4719	4722	4725	4733	4753	4777	4785
	4788	4791	4799	4806	4815	4834	4855	4863	4866	4869	4877	4884	4893	4919	4937
	4959	4967	4970	4973	4981	5001	5025	5033	5036	5039	5047	5054	5063	5085	5106
	5114	5117	5120	5128	5137	5178	5211	5229	5251	5259	5262	5265	5273	5300	5322
	5330	5333	5336	5344	5351	5360	5379	5400	5408	5411	5414	5422	5429	5438	5464
	5482	5504	5512	5515	5518	5526	5549	5560	5568	5569	5570	5576	5600	5608	5611
	5619	5626	5635	5657	5668	5676	5677	5678	5684	5707	5716	5719	5727	5734	5743
	5750	5780	5798	5820	5828	5831	5834	5842	5869	5891	5899	5902	5905	5913	5920
	5929	5948	5969	5977	5980	5983	5991	5998	6007	6033	6051	6073	6081	6084	6087
	6095	6122	6144	6152	6155	6158	6166	6173	6182	6201	6222	6230	6233	6236	6244
	6251	6260	6290	6308	6337	6345	6348	6351	6359	6379	6403	6411	6414	6417	6425
	6446	6465	6482	6494	6512	6535	6543	6546	6549	6557	6577	6596	6613	6625	6654
	6699	6717	6746	6754	6757	6768	6768	6788	6812	6820	6823	6826	6834	6854	6863
	6881	6903	6911	6914	6917	6925	6945	6954	6961	6990	7035	7050	7092	7100	7103
	7106	7114	7123	7137	7156	7197	7205	7208	7211	7219	7228	7242	7274	7288	7322
	7323	7329	7343	7346	7354	7362	7381	7413	7415	7423	7438	7441	7449	7456	7482
	7500	7522	7530	7533	7536	7544	7564	7593	7601	7604	7607	7615	7622	7631	7650
	7675	7683	7686	7689	7697	7704	7713	7720	7747	7765	7787	7795	7798	7801	7809
	7829	7853	7861	7864	7867	7875	7882	7891	7910	7933	7941	7944	7947	7955	7962
	7971	7978	8011	8029	8051	8059	8062	8065	8073	8093	8117	8125	8128	8131	8139
	8146	8155	8174	8195	8203	8206	8209	8217	8224	8233	8259	8277	8299	8307	8310
	8313	8321	8341	8365	8373	8376	8379	8387	8394	8403	8422	8444	8452	8455	8458
	8466	8473	8482	8489	8516	8534	8556	8564	8567	8570	8578	8598	8622	8630	8633
	8636	8644	8651	8660	8679	8701	8709	8712	8715	8723	8730	8739	8746	8773	8791
	8813	8821	8824	8827	8835	8855	8879	8887	8890	8893	8901	8908	8917	8936	8958
	8966	8969	8972	8980	8987	8996	9003								
CNACMS	3476#	3478													
COMEN	899#														
DYAVMS	3517#	3519													
DVTYMS	3561#	3563													
ENDCOM	899#														
ERROR	793#														
	3452	3510	3532	3546	3550	3556	3576	3585	3596	3621	3638	3660	3674	3682	
	3703	3726	3740	3748	3755	3764	3784	3805	3819	3827	3834	3843	3852	3878	3895
	3917	3931	3939	3960	3983	3997	4005	4021	4041	4041	4061	4075	4083	4090	4099
	4126	4143	4165	4179	4187	4208	4231	4245	4253	4260	4269	4292	4312	4326	4334
	4343	4352	4361	4373	4377	4384	4418	4435	4457	4471	4479	4500	4523	4537	4545
	4552	4561	4581	4602	4616	4624	4631	4640	4649	4675	4692	4714	4728	4736	4757
	4780	4794	4802	4809	4818	4838	4858	4872	4880	4887	4896	4923	4940	4962	4976
	4984	5005	5028	5042	5050	5057	5066	5089	5109	5123	5131	5140	5149	5159	5170
	5174	5181	5215	5232	5254	5268	5276	5304	5325	5339	5347	5354	5363	5383	5403
	5417	5425	5432	5441	5468	5485	5507	5521	5529	5553	5563	5573	5579	5603	5614
	5622	5629	5638	5661	5671	5681	5687	5710	5722	5730	5737	5746	5755	5784	5801
	5823	5837	5845	5873	5894	5908	5916	5923	5932	5952	5972	5986	5994	6001	6010
	6037	6054	6076	6090	6098	6126	6147	6161	6169	6176	6185	6205	6225	6239	6247
	6254	6263	6294	6311	6340	6354	6362	6383	6406	6420	6428	6449	6457	6468	6474
	6485	6491	6497	6516	6538	6552	6560	6580	6588	6599	6605	6616	6622	6628	6658
	6673	6703	6720	6749	6763	6771	6792	6815	6829	6837	6851	6857	6866	6885	6906

G10

CZRMBO RMO3/2 FCTNL TST 3
CZRMBO.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 332
CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0330

	6920	6928	6942	6948	6957	6966	6994	7009	7054	7065	7074	7095	7109	7117	7126
	7134	7140	7160	7171	7180	7200	7214	7222	7231	7238	7245	7292	7303	7326	7332
	7339	7349	7357	7365	7385	7395	7418	7426	7433	7444	7452	7459	7486	7503	7525
	7539	7547	7568	7596	7610	7618	7625	7634	7654	7678	7692	7700	7707	7716	7725
	7751	7768	7790	7804	7812	7833	7856	7870	7878	7885	7894	7914	7936	7950	7958
	7965	7974	7983	8015	8032	8054	8068	8076	8097	8120	8134	8142	8149	8158	8178
	8198	8212	8220	8227	8236	8263	8280	8302	8316	8324	8345	8368	8382	8390	8397
	8406	8426	8447	8461	8469	8476	8485	8494	8520	8537	8559	8573	8581	8602	8625
	8639	8647	8654	8663	8683	8704	8718	8726	8733	8742	8751	8777	8794	8816	8830
	8838	8859	8882	8896	8904	8911	8920	8940	8961	8975	8983	8990	8999	9008	
ESCAPE	899#	4277	4393	5074	5190	5538	5583	5647	5691						
GETCSI	757#														
GETDB	757#	4356	5153												
GETDS	757#														
GETDT	757#	3579													
GETMP1	757#														
GETPRI	899#														
GETSWR	757#	899#	3235#	3456											
MULT	899#														
NEWST	899#	3476	3517	3561	3600	3857	4105	4397	4654	4902	5194	5447	5763	6016	6269
	6680	7015	7253	7465	7730	7994	8242	8499	8756						
NWST	757#	3479	3521	3565	3603	3860	4108	4400	4657	4905	5197	5450	5766	6019	6272
	6683	7018	7256	7468	7733	7997	8245	8502	8759						
PJP	899#	3503	3504	3508	3509	3542	3543	3554	3555	9499	9718	9719	9732	9875	9876
	9877	9949	9951	9952	9953	9954	10055	10057	10058	10059	10088	10089	10090	10163	10165
	10166	10167	10168	10169	10170	10230	10232	10233	10234	10235	10236	10237	10259	10261	10301
	10303	10304	10305	10306	10454	10458	10602	11186	11187	11188	11189	11449	11451	11452	11453
	11513	12075	12076	12983	13067	13679	13759	13760	13823	13824					
PJSH	899#	3489	3490	3534	3536	9476	9701	9702	9725	9743	9744	9745	9901	9903	9904
	9905	9906	9977	9978	9979	9980	10072	10074	10075	10122	10123	10124	10125	10126	10127
	10128	10195	10196	10197	10198	10199	10200	10201	10243	10244	10272	10273	10274	10275	10276
	10450	10599	11150	11151	11152	11153	11433	11435	11436	11437	11507	12068	12069	12963	13026
	13658	13740	13746	13784	13786	13807									
PUTCSI	757#														
PUTDC	757#														
PUTER1	757#	7297	7389												
PUTMR1	757#	7059	7068	7165	7174										
REPORT	899#														
RGBFMC	757#	1377	1404												
SCOPE	794#	3522	3566	3604	3861	4109	4401	4658	4906	5198	5451	5767	6020	6273	6684
	7019	7257	7469	7734	7998	8246	8503	8760							
SETPRI	899#	3180	3470	12914	12920	13566									
SETTRA	13711#	13720	13721	13722	13723	13724	13726	13728	13729	13730	13731	13732	13733		
SETUP	899#	3185													
SKIP	899#														
SLASH	899#														
SPACE	899#														
STARS	899#	1225	1243	1245	1252	1266	1312	1315	3476	3478	3517	3520	3561	3564	3600
	3602	3611	3692	3774	3857	3859	3868	3949	4031	4105	4107	4116	4197	4279	4397
	4399	4408	4489	4571	4654	4656	4665	4746	4828	4902	4904	4913	4994	5076	5194
	5196	5205	5293	5373	5447	5449	5458	5542	5651	5763	5765	5774	5862	5942	6016
	6018	6027	6115	6195	6269	6271	6280	6284	6372	6506	6636	6680	6682	6691	6781
	6875	7015	7017	7043	7150	7253	7255	7281	7375	7465	7467	7476	7557	7644	7730
	7732	7741	7822	7904	7994	7996	8005	8086	8168	8242	8244	8253	8334	8416	8499
	8501	8510	8591	8673	8756	8758	8767	8848	8930	9031	9080	9281	9294	9306	9328
	9357	9379	9407	9471	9737	10497	10505	10590	10973	12947	12992	13016	13083	13160	13239

CZRMEBO RMO3/2 FCTNL IST 3
CZRMEB.P11 23-NOV-77 12:23

MACY11 30(1046) 23-NOV-77 12:49 PAGE 334
CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0332

.\$SAVE	757#	12945
.\$SCOP	757#	13237
.\$SIZE	757#	
.\$STRAP	757#	13684
.\$STYPB	757#	12990
.\$STYPD	757#	13014
.\$STYPE	757#	13158
.\$STYPO	757#	13081

. ABS. 111753 000

ERRORS DETECTED: 0

RMO3:CZRMEB.BIN,RMO3:CZRMEB.SEQ/DOC/NL:TOC/SOL/CRF=RMO3:CZRMEB.P11

RUN-TIME: 100 97 4 SECONDS

RUN-TIME RATIO: 918/201=4.5

CORE USED: 34K (67 PAGES)

DOCUMENT PAGES: 332