

.REM

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DZRJB-A-D
PRODUCT NAME: RPD4/5/6 FORMATTER PROGRAM
DATE CREATED: MAY 1976
MAINTAINER: DIAGNOSTIC ENGINEERING
AUTHOR: C. HESS

COPYRIGHT (C) 1976 DIGITAL EQUIPMENT CORP., MAYNARD, MASS.

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

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

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

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

CONTENTS

1. ABSTRACT
2. REQUIREMENTS
 - 2.1 EQUIPMENT
 - 2.2 PRELIMINARY PROGRAMS
3. LOADING PROCEDURES
4. STARTING PROCEDURES
 - 4.1 STARTING ADDRESSES
 - 4.2 OPERATOR ACTION
 - 4.3 RH11 - RH70 UNIBUS ADDRESS
 - 4.4 OTHER UNIBUS ADDRESSES
5. SWITCH REGISTER SETTINGS
6. ERROR MESSAGES
7. MISCELLANEOUS
 - 7.1 FORMAT TIMES
 - 7.2 HALTING THE PROGRAM
 - 7.3 SURFACE VERIFICATION
8. PROGRAM DESCRIPTION
 - 8.1 FORMAT OPERATION
 - 8.2 CHECK OPERATION
 - 8.3 POSITIONER VERIFICATION
9. PROGRAM LISTING

1. ABSTRACT

THE RPD4/5/6 FORMATTER PROGRAM FORMATS THE DISK PACK AND PERFORMS A CURSORY CHECK OF THE PACK'S SURFACE. THE PROGRAM ALLOWS THE OPERATOR TO SPECIFY ADDRESS LIMITS, PATTERNS, AND EITHER 16 BIT OR 18 BIT FORMAT MODE. THE PROGRAM VERIFIES EACH TRACK WRITTEN AS WELL AS VERIFYING THE FORMAT OPERATION.

2. REQUIREMENTS

2.1 EQUIPMENT

PDP-11 PROCESSOR
8K MEMORY
TELETYPE
PROGRAM LOAD DEVICE
KW11-L OR KW11-P CLOCK
RH11 OR RH70 WITH 1 - 8 RPD4, RPD5, RPD6 DISK DRIVES

2.2 PRELIMINARY PROGRAMS

RPD4/5/6 DISKLESS CONTROLLER TEST
PART 1 (MAINDEC-11-DZRJG)
PART 2 (MAINDEC-11-DZRJH)

RPD4/5/6 FUNCTIONAL CONTROLLER TEST
PART 1 (MAINDEC-11-DZRJI)
PART 2 (MAINDEC-11-DZRJJ)

3. LOADING PROCEDURES

THE PROGRAM MAY BE LOADED FROM PAPER TAPE USING THE ABSOLUTE LOADER OR IT MAY BE LOADED FROM THE APPROPRIATE 'XXDP' MEDIA USING THE ASSOCIATED LOADER. THE PROGRAM MAY NOT BE INCLUDED IN AN 'XXDP' CHAIN.

4. STARTING PROCEDURES

4.1 STARTING ADDRESSES

THE PROGRAM IS STARTED FROM LOCATION 200(8) IF THE ADDRESS OF THE

RH11 OR RH70 WILL NOT BE CHANGED

THE PROGRAM IS STARTED FROM LOCATION 204(8) IF THE ADDRESS OF THE RH11 OR RH70 IS TO BE CHANGED FROM THE PRELOADED VALUE (SEE SECTION 4.3)

4.2 OPERATION ACTION

1. LOAD THE PROGRAM INTO MEMORY (SEE SECTION 3).
2. LOAD THE STARTING ADDRESS - 200(8) OR 204(8).
3. SET THE SWITCHES AS REQUIRED AND PRESS 'START'.

IF THIS IS THE PROGRAM'S FIRST START, THE STATUS OF THE DRIVES ON THE SELECTED MASSBUS SUBSYSTEM WILL BE TYPED OUT. THIS TYPEOUT MAY BE INHIBITED ON SUBSEQUENT STARTS BY SETTING SW<02>.

4. THE PROGRAM WILL THEN TYPE THE FOLLOWING MESSAGE:

'PROGRAM MODE (C OR F):'

ENTER THE APPROPRIATE CODE: 'C' FOR 'CHECK' OPERATION OR 'F' FOR 'FORMAT & VERIFY'. IF A 'CARRIAGE RETURN' IS ENTERED IN RESPONSE TO THE REQUEST, THE PROGRAM WILL ASSUME 'FORMAT & VERIFY'.

5. THE PROGRAM WILL THEN ASK FOR THE FORMATTING MODE:

'OPERATE IN 22 SECTOR (16 BIT) MODE (Y OR N)'

ENTER THE APPROPRIATE CHARACTER: 'Y' FOR 16 BIT MODE OR 'N' FOR 18 BIT MODE. IF A 'CARRIAGE RETURN' IS ENTERED IN RESPONSE TO THIS REQUEST, THE PROGRAM WILL ASSUME 16 BIT MODE.

6. THE PROGRAM WILL THEN ASK FOR A DRIVE:

'DRIVE: '

ENTER THE ADDRESS OF THE DRIVE TO BE FORMATTED. A 'PERIOD' OR 'CARRIAGE RETURN' ENTRY WILL SELECT DRIVE 0. IF THE DRIVE SELECTED IS NOT AVAILABLE, THE PROGRAM WILL TYPE AN ERROR MESSAGE AND RETURN TO THE DRIVE ADDRESS REQUEST.

7. AFTER THE OPERATOR HAS SELECTED A DRIVE, THE PROGRAM WILL ASK FOR ADDRESS LIMITS FOR THE SELECTED DRIVE:

'ENTER ADDRESS LIMITS: '

THE PREVIOUSLY SELECTED OR DEFAULT VALUES FOR BEGINNING CYLINDER AND TRACK AND FOR ENDING CYLINDER AND TRACK WILL BE TYPED OUT. IF A 'CARRIAGE RETURN' IS TYPED AS A RESPONSE, THE PRESENT VALUE WILL BE USED; IF A 'PERIOD' IS TYPED, THE PROGRAM WILL BYPASS THE REMAINING ENTRIES AND WILL USE THEIR PRESENT VALUES. NOTE

THAT THE CYLINDER AND TRACK VALUES ARE D E C I M A L NUMBERS.
THE ADDRESS SPECIFIED BY THE BEGINNING CYLINDER AND TRACK MUST
BE LESS THAN THE ADDRESS SPECIFIED BY THE ENDING CYLINDER AND
TRACK ADDRESS.

7. THE PROGRAM WILL THEN ASK FOR THE DATA PATTERN:

'SELECT DATA PATTERN
(0) ZERO'S
(1) ONES
(2) WORST CASE:'

ENTER THE CODE FOR THE REQUIRED PATTERN. 'CARRIAGE RETURN' OR
'PERIOD' ENTRIES WILL CAUSE THE PROGRAM TO USE THE 'WORST CASE'
PATTERN.

THE 'WORST CASE' PATTERN IS THE FOLLOWING OCTAL SEQUENCE REPEATED
THROUGH THE DATA AREA OF THE SECTOR:

165555
133333

8. THE PROGRAM WILL THEN TYPE:

'STARTING FORMAT ON DRIVE N'

OR

'STARTING CHECK ON DRIVE N'

9. THE OPERATOR CAN DETERMINE WHERE THE DRIVE IS DURING THE FORMAT
OR CHECK OPERATION BY TYPING A 'CONTROL C'. THE PROGRAM WILL
TYPE THE FOLLOWING MESSAGE:

'PRESENT ADDRESS IS: CXXX TXX'

IF A 'CONTROL C' IS TYPED WHILE THE PROGRAM IS TYPING THE CURRENT
ADDRESS, THE PROGRAM WILL ABORT THE FORMAT (OR CHECK) OPERATION
AND RETURN TO THE 'DRIVE' REQUEST.

10. IF THE OPERATOR DOES NOT SELECT A DIFFERENT DRIVE WHEN THE FORMAT
OR CHECK OPERATION HAS COMPLETED, THE PROGRAM WILL NOT ALTER THE
ADDRESS LIMITS SPECIFIED AT THE START OF THE PREVIOUS OPERATION.
IF THE FORMAT OR CHECK OPERATION IS TERMINATED BY A 'CONTROL C' OR
IF A DIFFERENT DRIVE IS SELECTED, THE PROGRAM WILL RESET THE
ADDRESS LIMITS TO THE VALUES APPROPRIATE FOR THE DRIVE TYPE.

11. TO CHANGE EITHER THE ADDRESSING MODE OR THE OPERATION MODE, THE
PROGRAM MUST BE STARTED FROM LOCATION 200(8) OR 204(8) AGAIN.

4.3 RH11 - RH70 UNIBUS ADDRESS

THE PROGRAM ASSUMES THAT THE RH11 OR RH70 ADDRESSES START AT
176700 AND THAT THE VECTOR ADDRESS IS 254. THESE ADDRESSES MAY

BE CHANGED WHEN THE PROGRAM IS STARTED FROM LOCATION 204(8).
IF THE RH11 - RH70 IS NOT AT THE DEFAULT ADDRESS, THE PROGRAM MUST
BE STARTED FROM 204(9) INITIALLY AS THE PROGRAM GOES THROUGH THE
ADDRESS CHANGE ROUTINE AT INITIAL START ONLY. ENTER THE RH11 RH70
ADDRESS IN RESPONSE TO THE REQUEST FROM THE PROGRAM.

4.4 OTHER UNIBUS ADDRESSES

LOC	TAG	CONTENTS	FUNCTION
1144	\$TKS	177560	TTY KEYBOARD STATUS REGISTER
1146	\$TKB	177562	TTY KEYBOARD BUFFER REGISTER
1150	\$TPS	177564	TTY PRINTER STATUS REGISTER
1152	\$TPB	177566	TTY PRINTER BUFFER REGISTER
1176	\$LKCSR	172540	KW11-P CONTROL REGISTER
1200	\$LKCSB	172542	KW11-P COUNTER REGISTER
1202	\$LPVEC	104	KW11-P VECTOR ADDRESS
1206	\$LKS	177546	KW11-L CONTROL REGISTER
1210	\$LKV	100	;ADDRESS OF KW11-L VECTOR

5. SWITCH REGISTER SETTINGS

SW<15>=1...HALT ON ERROR
SW<13>=1...INHIBIT ERROR TYPEOUTS
SW<10>=1...BELL ON ERROR
SW<09>=1...LOOP ON ERROR
SW<02>=1...DON'T DISPLAY SYSTEM STATUS AFTER INITIAL START
SW<01>=1...LOOP ON THE CURRENT TRACK
SW<00>=1...LOOP THE PROGRAM ON THE SELECTED DRIVE

IF THE PROGRAM IS BEING RUN ON A SWITCHLESS PROCESSOR (I.E. AN 11/34)
THE PROGRAM WILL DETERMINE THAT THE HARDWARE SWITCH REGISTER IS
NOT PRESENT AND WILL USE A 'SOFTWARE' SWITCH REGISTER. THE
'SOFTWARE' SWITCH REGISTER IS LOCATED AT LOCATION 176 (8). THE
SETTINGS OF THE 'SOFTWARE' SWITCHES ARE CONTROLLED THROUGH A KEYBOARD
ROUTINE WHICH IS CALLED BY TYPING A 'CONTROL G'. THE PROGRAM WILL
RECOGNIZE THE 'CONTROL G' AT ANY TIME EXCEPT WHEN THE PROGRAM
IS AT A HIGHER PRIORITY PROCESSING AN RPD4/5/6 INTERRUPT. THE
'SOFTWARE' SWITCH VALUES ARE ENTERED AS AN OCTAL NUMBER IN RESPONSE
TO THE PROMPT FROM THE SWITCH ENTRY ROUTINE:

'SWR = NNNNNN NEW ='

EACH TIME SWITCH SETTING ARE ENTERED, THE ENTIRE SWITCH REGISTER
IMAGE MUST BE ENTERED. LEADING ZEROS ARE NOT REQUIRED. 'RUBOUT' AND
'CONTROL U' FUNCTIONS MAY BE USED TO CORRECT TYPING ERRORS
DURING SWITCH ENTRY.

ON PROCESSORS WITH HARDWARE SWITCH REGISTERS, THE 'SOFTWARE' SWITCH

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

6. ERROR MESSAGES

1. 'RH11 INTERRUPT OCCURRED (RPAS=0) - AN INTERRUPT OCCURRED, BUT NOTHING ON THE MASSBUS IS INDICATING AN ATTENTION.
2. 'UNEXPECTED ATTENTION OCCURRED' - THE INDICATED DRIVE INTERRUPTED, BUT NO INTERRUPT WAS EXPECTED FROM THE INDICATED DRIVE.
3. 'MASSBUS PARITY ERROR (MCPE=1)' - A CONTROL BUS PARITY ERROR WAS DETECTED BY THE RH11 WHEN THE INDICATED REGISTER WAS READ.
4. 'MASSBUS PARITY ERROR (PAR=1)' - A CONTROL BUS PARITY ERROR OCCURRED WHEN THE INDICATED REGISTER WAS WRITTEN.
5. 'ADDRESS PLUG CHANGE BIT SET' - THE PROGRAM FOUND THE 'OPE' BIT SET FOR THE INDICATED DRIVE.
6. 'RH11 DIDN'T RESPOND TO ADDRESSING' - THE PROGRAM ACCESSED THE RH11 AT THE INDICATED ADDRESS AND RECEIVED NO RESPONSE.
7. 'DRIVE OFFLINE' - THE INDICATED DRIVE HAS GONE OFFLINE
8. 'PERSISTENT DRIVE UNSAFE ERROR' - THE INDICATED DRIVE HAS BECOME UNSAFE AND THE CONDITION CANNOT BE CLEARED BY ISSUING 'DRIVE CLEAR' INSTRUCTIONS.
9. 'UNCORRECTABLE MASSBUS PARITY ERROR' - THE PROGRAM ATTEMPTED TO PERFORM AN OPERATION AND DETECTED 3 SUCESSIVE MASSBUS PARITY ERRORS OR THE PROGRAM ATTEMPTED TO CLEAR A 'PAR' ERROR IN THE DRIVE AND A PARITY ERROR OCCURRED.
10. 'SOFTWARE TIMEOUT' - THE OPERATION FAILED TO COMPLETE WITHIN 1 SECOND.
11. 'DRIVE UNSAFE ERROR' - A NON-PERSISTENT UNSAFE ERROR OCCURRED DURING THE OPERATION.
12. 'CONTROLLER/DRIVE ERROR DURING WRITE' - THE INDICATED NON-DATA ERROR WAS DETECTED DURING A FORMAT OPERATION.
13. 'CONTROLLER/DRIVE ERROR DURING WRITE CHECK' - A NON-DATA ERROR OCCURRED DURING THE WRITE CHECK.
14. 'DATA ERROR DURING WRITE CHECK' - A DATA RELATED ERROR OCCURRED DURING A WRITE CHECK OPERATION. A DATA ERROR IS CONSIDERED

TO BE ONE OF THE FOLLOWING ERRORS: 'DCK', 'OPI', 'DTE', 'HCRC',
'HCE', OR 'FER'.

15. 'RETRIES NOT SUCCESSFUL - SECTOR NOT ACCEPTABLE' - THE INDICATED SECTOR FAILED DURING RETRY FOLLOWING A DATA ERROR.
16. 'CONTROLLER/DRIVE ERROR VERIFYING HEADERS' - AN ERROR OCCURRED DURING THE VERIFICATION PASS AFTER FORMATTING.
17. ''HCE' ERROR VERIFYING HEADERS' - A HEADER ERROR OCCURRED DURING THE VERIFICATION PASS AFTER FORMATTING.
18. 'CYLINDER FIELD IN HEADER IS NOT CORRECT' - THE CYLINDER FIELD FROM A HEADER READ DURING THE VERIFICATION PASS IS NOT CORRECT.
19. 'WRITE CHECK ERROR' - THE RH11 REPORTED A WRITE CHECK ERROR AND NO DRIVE ERRORS WERE SET.

7. MISCELLANEOUS

7.1 FORMAT TIME

IT TAKES APPROXIMATELY 8 MINUTES TO FORMAT AN ENTIRE RPO4/5 PACK AND APPROXIMATELY 16 MINUTES TO FORMAT AN RPO6 PACK. THE 'CHECK' MODE TIME FOR AN ENTIRE PACK IS 4 MINUTES FOR RPO4/5'S AND 8 MINUTES FOR RPO6'S.

7.2 HALTING THE PROGRAM

THE OPERATOR SHOULD NOT HALT THE PROGRAM DURING A FORMAT OPERATION. HALTING THE PROGRAM MAY LEAVE A SECTOR INCORRECTLY FORMATTED. TO TERMINATE THE FORMAT, TYPE A 'CONTROL C' AND WHILE THE PROGRAM IS TYPING THE ADDRESS, TYPE ANOTHER 'CONTROL C'; THIS SEQUENCE RETURNS THE PROGRAM TO THE DRIVE ADDRESS ENTRY ROUTINE.

7.3 SURFACE VERIFICATION

THE FORMATTER PROGRAM IS NOT INTENDED TO BE USED TO PERFORM DISK PACK VERIFICATION. IF THE PROGRAM REPORTS A SECTOR AS BEING 'NOT ACCEPTABLE', THIS MAY IN FACT INDICATE A BAD SURFACE; HOWEVER, SECTORS WHICH 'PASSED' MAY OR MAY NOT BE GOOD.

8. PROGRAM DESCRIPTION

8.1 FORMAT OPERATION

THE PROGRAM FORMATS THE PACK ONE TRACK AT A TIME BETWEEN THE LIMITS SPECIFIED BY THE DRIVE.

THE FORMAT OPERATION CONSISTS OF A WRITE HEADER AND DATA COMMAND FOR THE ENTIRE TRACK FOLLOWED BY A WRITE CHECK HEADER AND DATA COMMAND. IF AN ERROR OCCURS DURING THE WRITE, THE PROGRAM WILL RETRY THE WRITE BEFORE CONTINUING. IF A DATA RELATED ERROR IS DETECTED DURING THE WRITE CHECK (A DATA ERROR IS DEFINED AS ONE OF THE FOLLOWING ERRORS: 'DCK', 'OPI', 'DTE', 'HCRC', 'HCE', OR 'FER'), THE SECTOR IN ERROR WILL BE REWRITTEN. THE PROGRAM WILL CHECK THE SECTOR TWICE; IF A DATA ERROR IS DETECTED IN THE SECTOR DURING EITHER OF THE WRITE CHECKS, THE PROGRAM WILL DECLARE THE SECTOR AS BEING 'UNACCEPTABLE'. FOLLOWING THIS SEQUENCE, THE REMAINDER OF THE TRACK IS CHECKED. IF DATA ERRORS ARE ENCOUNTERED IN ANY OF THE REMAINING SECTORS, EACH SECTOR WILL BE HANDLED AS DESCRIBED ABOVE.

IF A NON-DATA RELATED ERROR OCCURS DURING THE WRITE CHECK, THE PROGRAM WILL RETRY THE COMMAND FOR THE ENTIRE TRACK BEFORE PROCEEDING TO THE NEXT TRACK.

IT SHOULD BE NOTED THAT THE FORMATTER PROGRAM FORMATS THE PACK ONLY AND IS NOT DIRECTLY VERIFYING THE SURFACE OF THE PACK. ERRORS THAT ARE ENCOUNTERED MAY BE THE RESULT OF BAD AREAS ON THE PACK BUT, AS THE PROGRAM IS NOT DESIGNED TO PERFORM AN EXHAUSTIVE CHECK OF THE DISK PACK SURFACE, THE PROGRAM CANNOT MAKE THIS CONCLUSION. IN GENERAL, HOWEVER, THE OPERATOR CAN ASSUME THAT SECTORS WHICH THE PROGRAM CALLS 'UNACCEPTABLE' INDICATE BAD AREAS OF THE PACK; UNFORTUNATELY, SECTORS WHICH 'PASS' CANNOT BE ASSUMED TO BE GOOD.

DURING THE FORMAT OPERATION, THE PROGRAM FILLS THE DATA FIELD WITH THE PATTERN SELECTED BY THE OPERATOR. THE KEYWORDS IN THE HEADER ARE ALWAYS SET TO ZERO.

8.2 CHECK OPERATION

THE CHECK OPERATION IS IDENTICAL TO THE WRITE CHECK PORTION OF THE FORMAT OPERATION DESCRIBED IN SECTION 8.1 EXCEPT THAT THE PROGRAM WILL NOT RE-WRITE ERROR SECTORS.

8.3 POSITIONER VERIFICATION

AFTER THE PROGRAM COMPLETES THE FORMAT OPERATION, THE POSITIONER IS RETURNED TO THE STARTING CYLINDER AND THE HEADER FROM SECTOR 0 ON THE STARTING TRACK IS READ. THE CYLINDER ADDRESS FIELD FROM THE HEADER IS COMPARED TO THE REQUESTED CYLINDER; IF THE CYLINDER ADDRESSES DO NOT COMPARE, AN ERROR MESSAGE IS TYPED. THE PROGRAM CHECKS THE CYLINDER ADDRESS FIELD FROM THE HEADER OF SECTOR 0 ON THE BEGINNING TRACK OF EACH CYLINDER FORMATTED. THIS CHECK IS PERFORMED TO CONFIRM THAT THE DISK'S POSITIONER ADVANCED PROPERLY DURING THE FORMAT OPERATION.

L01

.MAIN. MACY11 27(655) 27-APR-76 18:02 PAGE 10
DZRJBA.DOC

SEQ 0010

9. PROGRAM LISTING

MO1

.MAIN. MACY11 27(655) 27-APR-76 18:02 PAGE 11
DZRJBA.DOC

SEQ 0011

%
.END

ERRORS DETECTED: 0

*.DZRJBA/SOL=DZRJBA.DOC
RUN-TIME: 1 2 0 SECONDS
CORE USED: 3K

15	OPERATIONAL SWITCH SETTINGS
27	TRAP CATCHER
37	ACT11 HOOKS
48	STARTING ADDRESS = 200
52	STARTING ADDRESS TO CHANGE THE RH11 ADDRESS = 204
55	BASIC DEFINITIONS
168	RH11 REGISTERS
214	RPO4/5/6 REGISTERS
409	RPO4/5/6 DRIVER COMMANDS
435	COMMON TAGS
634	ERROR POINTER TABLE
791	MAIN PROGRAM
799	INITIALIZE THE COMMON TAGS
841	GET VALUE FOR SOFTWARE SWITCH REGISTER
1278	END OF PASS ROUTINE
1321	SUPPORT SUBROUTINES
1674	MACRO ROUTINES
1677	ERROR HANDLER ROUTINE
1799	TYPE ROUTINE
1870	BINARY TO OCTAL (ASCII) AND TYPE
1948	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
2016	TTY INPUT ROUTINE
2273	TYPE NUMERICAL ASCII STRING SUPPRESS LEADING ZEROS
2297	SINGLE LENGTH BINARY TO DECIMAL ASCII ROUTINE
2316	DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
2379	SAVE AND RESTORE R0-R5 ROUTINES
2425	TRAP DECODER
2448	TRAP TABLE
2472	SINGLE/DUAL PORT RH11/RPO4/5/6 DRIVER (REV 1.0)
3888	TELETYPE MESSAGES
4037	ERROR MESSAGES
4404	BUSADR - GET BUS ADDRESS AND VECTOR ADDRESS FOR RH11
4466	CK.NUM - CHECK NUMBER (OCTAL)

MD-11-DZRJB-A, RPO4/5/6 FORMATTER PROGRAM
RPO456.010
MACY!! 27(655) 27-APR-76 17:59 PAGE 2
SEQ 0014

```

.TITLE MD-11-DZRJB-A, RPO4/5/6 FORMATTER PROGRAM
.*COPYRIGHT (C) 1976
.*DIGITAL EQUIPMENT CORP.
.*MAYNARD, MASS. 01754
.*
.*PROGRAM BY C. HESS
.*
.*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
.*PACKAGE (MAINDEC-11-DZQAC-C1), MAR 24, 1976.
.*

```

.SBTTL OPERATIONAL SWITCH SETTINGS

```

.*
.*      SWITCH      USE
.*      -----
.*      15          HALT ON ERROR
.*      12          INHIBIT ERROR TYPEOUTS
.*      10          BELL ON ERROR
.*      9           LOOP ON ERROR
.*      2           DON'T DISPLAY SYSTEM STATUS AFTER INITIAL START
.*      1           LOOP ON THE CURRENT TRACK
.*      0           LOOP THE PROGRAM ON THE SELECTED DRIVE
.*

```

.SBTTL TRAP CATCHER

000000

```

.=0
.*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
.*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
.*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS

```

000174 000174
000176 000000

```

.=174
DISPREG: .WORD 0      ;;SOFTWARE DISPLAY REGISTER
SWREG:   .WORD 0      ;;SOFTWARE SWITCH REGISTER

```

.SBTTL ACT11 HOOKS

```

;*****
;HOOKS REQUIRED BY ACT11

```

000046 000200
000052 000046
000052 005174
000052 000052
000052 020000
000052 000200

```

$SVPC=.          ;SAVE PC
.=46
SENDAD          ;;1)SET LOC.46 TO ADDRESS OF SENDAD IN .SEOP
.=52
.WORD 20000     ;;2)SET LOC.52 TO 20000
.= $SVPC        ;; RESTORE PC

```

000200 000200 002076
000137

```

.SBTTL STARTING ADDRESS = 200
.=200
JMP BEGIN1     ;NORMAL STARTING ADDRESS

```

000204 000137 002066

```

.SBTTL STARTING ADDRESS TO CHANGE THE RH11 ADDRESS = 204
JMP BEGIN     ;CHANGE THE RH11 ADDRESS

```

.SBTTL BASIC DEFINITIONS

110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
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
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200

000002
000001

SW01= 2
SW00= 1
.EQUIV SW09, SW9
.EQUIV SW08, SW8
.EQUIV SW07, SW7
.EQUIV SW06, SW6
.EQUIV SW05, SW5
.EQUIV SW04, SW4
.EQUIV SW03, SW3
.EQUIV SW02, SW2
.EQUIV SW01, SW1
.EQUIV SW00, SW0

100000
040000
020000
010000
004000
002000
001000
000400
000200
000100
000040
000020
000010
000004
000002
000001

.*DATA BIT DEFINITIONS (BIT00 TO BIT15)

BIT15= 100000
BIT14= 40000
BIT13= 20000
BIT12= 10000
BIT11= 4000
BIT10= 2000
BIT09= 1000
BIT08= 400
BIT07= 200
BIT06= 100
BIT05= 40
BIT04= 20
BIT03= 10
BIT02= 4
BIT01= 2
BIT00= 1
.EQUIV BIT09, BIT9
.EQUIV BIT08, BIT8
.EQUIV BIT07, BIT7
.EQUIV BIT06, BIT6
.EQUIV BIT05, BIT5
.EQUIV BIT04, BIT4
.EQUIV BIT03, BIT3
.EQUIV BIT02, BIT2
.EQUIV BIT01, BIT1
.EQUIV BIT00, BIT0

.*BASIC "CPU" TRAP VECTOR ADDRESSES

000004
000010
000014
000014
000014
000020
000024
000030
000034
000060
000064
000240

ERRVEC= 4 ;: TIME OUT AND OTHER ERRORS
RESVEC= 10 ;: RESERVED AND ILLEGAL INSTRUCTIONS
TBITVEC= 14 ;: "T" BIT
TRTVEC= 14 ;: TRACE TRAP
BPTVEC= 14 ;: BREAKPOINT TRAP (BPT)
IOTVEC= 20 ;: INPUT/OUTPUT TRAP (IOT) **SCOPE**
PWRVEC= 24 ;: POWER FAIL
EMTVEC= 30 ;: EMULATOR TRAP (EMT) **ERROR**
TRAPVEC= 34 ;: "TRAP" TRAP
TKVEC= 60 ;: TTY KEYBOARD VECTOR
TPVEC= 64 ;: TTY PRINTER VECTOR
PIRQVEC= 240 ;: PROGRAM INTERRUPT REQUEST VECTOR

164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
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
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
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
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
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
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
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
628
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
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
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

.SBTTL RH11 REGISTERS

:CONTROL AND STATUS REGISTER 1 (RPCS1)

000100	IE=	100	: INTERRUPT ENABLE (BIT #6)
000200	RDY=	200	: READY (BIT #7)
000400	A16=	400	: HIGH ORDER BUS ADDRESS BIT (BIT #8)
001000	A17=	1000	: HIGH ORDER BUS ADDRESS BIT (BIT #9)
002000	PSEL=	2000	: PORT SELECT (BIT #10)
020000	MCPE=	20000	: MASSBUS PARITY ERROR (BIT #13)
040000	TRE=	40000	: TRANSFER ERROR (BIT #14)
	:SC=	100000	: SPECIAL CONDITION (BIT #15)

:WORD COUNT REGISTER (RPWC)
:(EACH BIT IS CALLED BY BIT NUMBER)

:BUS ADDRESS REGISTER (RPBA)
:(EACH BIT IS CALLED BY BIT NUMBER)

:CONTROL AND STATUS REGISTER 2 (RPCS2)

000001	US1=	1	: UNIT SELECT (BIT #0)
000002	US2=	2	: UNIT SELECT (BIT #1)
000004	US4=	4	: UNIT SELECT (BIT #2)
000010	BAT=	10	: BUS ADDRESS INCREMENT INHIBIT (BIT #3)
000020	PAT=	20	: MASSBUS PARITY TEST (BIT #4)
000040	CLR=	40	: CLEAR (BIT #5)
000100	IR=	100	: INPUT READY (BIT #6)
000200	OR=	200	: OUTPUT READY (BIT #7)
000400	MPE=	400	: MASS BUS PARITY ERROR (BIT #8)
001000	MXF=	1000	: MISSED TRANSFER ERROR (BIT #9)
002000	PGE=	2000	: PROGRAM ERROR (BIT #10)
004000	NEM=	4000	: NON EXISTENT MEMORY (BIT #11)
010000	NED=	10000	: NON EXISTENT DRIVE (BIT #12)
020000	UPE=	20000	: UNIBUS PARITY ERROR (BIT #13)
040000	WCE=	40000	: WRITE CHECK ERROR (BIT #14)
100000	DLT=	100000	: DATA LATE (BIT #15)

:DATA BUFFER REGISTER (RPDB)
:(EACH BIT IS CALLED BY BIT NUMBER)

.SBTTL RP04/5/6 REGISTERS

:CONTROL AND STATUS 1 REGISTER. (#00)

326	000001	WCU=	1	;WRITE CURRENT UNSAFE (BIT #0)
327	000002	CSF=	2	;CURRENT SINK FAILURE (BIT #1)
328	000004	WSU=	4	;WRITE SELECT UNSAFE (BIT #2)
329	000010	CSU=	10	;CURRENT SWITCH UNSAFE (BIT #3)
330	000020	MSE=	20	;MOTOR SEQUENCE ERROR (BIT #4)
331	000040	TDF=	40	;TRANSITIONS DETECTOR FAILURE (BIT #5)
332	000100	TUF=	100	;TRANSITIONS UNSAFE (BIT #6)
333	000200	FEN=	200	;FAILSAFE ENABLED (BIT #7)
334	000400	WRU=	400	;WRITE READY UNSAFE (BIT #8)
335	001000	MHS=	1000	;MULTIPLE HEAD SELECT (BIT #9)
336	002000	NHS=	2000	;NO HEAD SELECTION (BIT #10)
337	004000	IXE=	4000	;INDEX ERROR (BIT #11)
338	010000	VU30=	10000	;30VOLT UNSAFE (BIT #12)
339	020000	PLU=	20000	;PLO UNSAFE (BIT #13)
340	100000	ACU=	100000	;AC UNSAFE (BIT #15)

;RPO5/6 ERROR REGISTER #02 (RPER2) (#10)

341	000001	WCU=	1	;WRITE CURRENT UNSAFE (BIT #0)
342	000002	CSF=	2	;CURRENT SINK FAILURE (BIT #1)
343	000004	WSU=	4	;WRITE SELECT UNSAFE (BIT #2)
344	000010	CSU=	10	;CURRENT SWITCH UNSAFE (BIT #3)
345	000020	RAW=	20	;READ AND WRITE (BIT #4)
346	000040	TDF=	40	;TRANSITIONS DETECTOR FAILURE (BIT #5)
347	000100	TUF=	100	;TRANSITIONS UNSAFE (BIT #6)
348	000200	ABS=	200	;ABNORMAL STOP (BIT #7)
349	000400	WRU=	400	;WRITE READY UNSAFE (BIT #8)
350	001000	MHS=	1000	;MULTIPLE HEAD SELECT (BIT #9)
351	002000	NHS=	2000	;NO HEAD SELECTION (BIT #10)
352	004000	IXE=	4000	;INDEX ERROR (BIT #11)
353	0020000	PLU=	20000	;PLO UNSAFE (BIT #12)

;OFFSET REGISTER (RPOF) (#11)

354	000001	OF25=	1	;OFFSET 25 MICRO INCHES (BIT #0)
355	000002	OF50=	2	;OFFSET 50 MICRO INCHES (BIT #1)
356	000004	OF100=	4	;OFFSET 100 MICRO INCHES (BIT #2)
357	000010	OF200=	10	;OFFSET 200 MICRO INCHES (BIT #3)
358	000020	OF400=	20	;OFFSET 400 MICRO INCHES (BIT #4)
359	000040	OF800=	40	;OFFSET 800 MICRO INCHES (BIT #5)
360	000200	OFREV=	200	;OFFSET NEGATIVE (REVERSE) (BIT #5)
361	002000	HCI=	2000	;HEADER COMPARE INHIBIT (BIT #10)
362	004000	ECI=	4000	;ERROR CORRECTION CODE INHIBIT (BIT #11)
363	010000	FMT22=	10000	;FORMAT BIT (BIT #12)

;DESIRED CYLINDER ADDRESS (RPCA) (#12)
;(EACH BIT IS CALLED BY BIT NUMBER)

;CURRENT CYLINDER ADDRESS (RPCC) (#13)
;(EACH BIT IS CALLED BY BIT NUMBER)

;SERIAL NUMBER REGISTER (RPSN) (#14)
;(EACH IS CALLED BY BIT NUMBER)

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
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433

000001
000002
000010
000040
000100
040000
100000

000001
000002
000040
000100
020000
040000
100000

000101
000103
000105
000107
000111
000113
000115
000117
000121
000123
000131
000141
000143
000145
000151
000153
000161
000163
000171
000173

;RPO4 ERROR REGISTER #03 (RPER3) (#15)

PSU= 1 ;PACK SPEED UNSAFE (BIT #0)
VUF= 2 ;VELOCITY UNSAFE (BIT #1)
UWR= 10 ;ANY UNSAFE EXCEPT READ/WRITE (BIT #3)
ACL= 40 ;AC LOW (BIT #5)
DCL= 100 ;DC LOW (BIT #6)
SKI= 40000 ;SEEK INCOMPLETE (BIT #14)
OCYL= 100000 ;OFF CYLINDER (BIT #15)

;RPO5/6 ERROR REGISTER #03 (RPER3) (#15)

DCU= 1 ;DC UNSAFE (BIT #0)
WAO= 2 ;WRITE AND OFFSET (BIT #1)
ACL= 40 ;AC LOW (BIT #5)
DCL= 100 ;DC LOW (BIT #6)
OPE= 20000 ;OPERATOR PLUG ERROR (BIT #13)
SKI= 40000 ;SEEK INCOMPLETE (BIT #14)
OCYL= 100000 ;OFF CYLINDER ERROR (BIT #15)

;ECC POSITION REGISTER (RPEC1) (#16)
;(EACH BIT IS CALLED BY BIT NUMBER)

;ECC PATTERN REGISTER (RPEC2) (#17)
;(EACH BIT IS CALLED BY BIT NUMBER)

;;*****

.SBTTL RPO4/5/6 DRIVER COMMANDS

;;*****

RNOP = 101 ;NO OPERATION
UNLOAD = 103 ;UNLOAD
SEEK = 105 ;SEEK
RECAL = 107 ;RECALIBRATE
DRVCLR = 111 ;DRIVE CLEAR
RELSE = 113 ;RELEASE
OFFSET = 115 ;OFFSET
RTC = 117 ;RETURN TO CENTER LINE
READIN = 121 ;READ IN PRESET
ACK = 123 ;PACK ACKNOWLEDGE
SEARCH = 131 ;SEARCH
GETREG = 141 ;GET REGISTERS
SETFMT = 143 ;SET FORMAT (& ECI OR HCI)
SELDRV = 145 ;SELECT DRIVE
WCKD = 151 ;WRITE CHECK DATA
WCKHD = 153 ;WRITE CHECK HEADER & DATA
WRTDAT = 161 ;WRITE DATA
WRTHD = 163 ;WRITE HEADER & DATA
RDDAT = 171 ;READ DATA
RDHD = 173 ;READ HEADER & DATA

488	001220	000000	MODE:	.WORD	0	:	'FORMAT & VERIFY' OR 'CHECK' MODE INDICATOR
489	001222	000632	ENDCYL:	.WORD	410.	:	ENDING CYLINDER
490	001224	000000	BEGCYL:	.WORD	0	:	STARTING CYLINDER
491	001226	000022	ENDTRK:	.WORD	18.	:	ENDING TRACK
492	001230	000000	BEGTRK:	.WORD	0	:	STARTING TRACK
493	001232	000000	TTRKS:	.WORD	0	:	TOTAL # OF TRACKS TO BE FORMATTED
494	001234	000000	TTRKSC:	.WORD	0	:	TOTAL # OF TRACKS COUNTER
495	001236	000000	TRKCNT:	.WORD	0	:	COUNTS TRKS FROM 0-18 PER CYL
496	001240	000000	PATSEL:	.WORD	0	:	CONTAINS PATTERN SELECTED
497	001242	000000	PATA:	.WORD	0	:	1ST WORD OF PATTERN
498	001244	000000	PATB:	.WORD	0	:	2ND WORD OF PATTERN
499	001246	000000	CYLCK:	.WORD	0	:	STORE CYLINDER ADDRESS FOR FORMAT VERIFICATION
500	001250	000000	RETRY:	.WORD	0	:	MAINTAINS # OF WRITE RETRIES MADE
501	001252	000000	SAVSEC:	.WORD	0	:	CONTAINS LAST BAD SECTOR ON FORMAT
502	001254	000000	SAVWC:	.WORD	0	:	CONTAINS WC FOR REMAINING SECTORS ON ERROR
503	001256	000000	WC:	.WORD	0	:	20 OR 22 SECTOR TRACK SIZE (IN WORDS)
504	001260	000000	MWC:	.WORD	0	:	2'S COMPLEMENT OF 'WC'
505	001262	000000	HEDERR:	.WORD	0	:	POSITIONING ERROR DURING FORMAT INDICATOR
506	001264	000000	SEC20:	.WORD	0	:	20 OR 22 SECTOR MODE INDICATOR
507						:	0 = 20 SECTOR MODE
508						:	1'S = 22 SECTOR MODE
509	001266	000000	MAXSEC:	.WORD	0	:	MAXIMUM SECTOR ADDRESS (FOR EITHER 20 OR 22 SECTOR
510						:	FORMAT)
511	001270	000000	DS.CYL:	.WORD	0	:	ADDRESS OF CURRENT CYLINDER
512	001272	000000	DS.TRK:	.WORD	0	:	ADDRESS OF CURRENT TRACK
513	001274	000000	DDRIVE:	.WORD	0	:	DRIVE NUMBER OF 'DRIVER' ERROR MESSAGES
514	001276	000000	ATTN:	.WORD	0	:	ATTENTION REGISTER IMAGE FOR 'DRIVER'
515						:	ERROR MESSAGES
516	001300	000000	CNTLC:	.WORD	0	:	ADDRESS OF '↑C' RETURN
517	001302	000000	CHGADR:	.WORD	0	:	'CHANGE RH11 ADDRESS' FLAG
518						:	
519						:	
520						:	
521	001304	000000	RP.REG:	.WORD	0	:	RPCS1
522	001306	000000		.WORD	0	:	RPWC
523	001310	000000		.WORD	0	:	RPBA
524	001312	000000		.WORD	0	:	RPDA
525	001314	000000		.WORD	0	:	RPCS2
526	001316	000000		.WORD	0	:	RPDS1
527	001320	000000		.WORD	0	:	RPER1
528	001322	000000		.WORD	0	:	RPAS
529	001324	000000		.WORD	0	:	RPLA
530	001326	000000		.WORD	0	:	RPDB
531	001330	000000		.WORD	0	:	RPMR
532	001332	000000		.WORD	0	:	RPDT
533	001334	000000		.WORD	0	:	RPSN
534	001336	000000		.WORD	0	:	RPOF
535	001340	000000		.WORD	0	:	RPCA
536	001342	000000		.WORD	0	:	RPCC
537	001344	000000		.WORD	0	:	RPER2
538	001346	000000		.WORD	0	:	RPER3
539	001350	000000		.WORD	0	:	RPEC1
540	001352	000000		.WORD	0	:	RPEC2
541						:	

;RH11/RP04/5/6 REGISTERS STORED HERE AFTER AN OPERATION

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
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595

;DATA/PARAMETER BLOCK - USED FOR ALL DRIVE OPERATION

```
FMTDPB: .BYTE 0 ;DRIVE NUMBER
         .BYTE 0 ;OFFSET VALUE OR FMT22,ECI, AND HCI
         .BYTE 0 ;COMMAND
         .BYTE 0 ;PSEL AND A17 AND A16
         .WORD 0 ;WORD COUNT (NEG)
         .WORD 0 ;BUFFER ADDRESS
         .BYTE 0 ;SECTOR ADDRESS
         .BYTE 0 ;TRACK ADDRESS
         .WORD 0 ;CYLINDER ADDRESS
         .WORD RP.REG ;ERROR TABLE POINTER
         .WORD 0 ;STATUS-ERROR INDICATOR
         ;BIT 15 = 1: ERROR OCCURRED
         ;BIT 07 = 1: DONE
         ;BIT 14-10 AND BIT 06-03
         ;INDICATE TYPE OF ERROR
```

;PARAMETER POINTER TABLE

```
TABLE: PAR1,0,BEGCYL
        PAR2,18.,BEGTRK
        PAR3,0,ENDCYL
        PAR4,18.,ENDTRK,0
```

;ASCII MESSAGES FOR ADDRESS PARAMETERS

```
PAR1: .ASCIZ @START CYL @
PAR2: .ASCIZ @START TRK @
PAR3: .ASCIZ @END CYL @
PAR4: .ASCIZ @END TRK @
```

;SECTOR BUFFER ADDRESS TABLE

```
ADRTBL: .WORD BUFP ;ADDRESS OF SECTOR 0
         .WORD BUFP+(520.*1.) ;ADDRESS OF SECTOR 1
         .WORD BUFP+(520.*2.) ;ADDRESS OF SECTOR 2
         .WORD BUFP+(520.*3.) ;ADDRESS OF SECTOR 3
         .WORD BUFP+(520.*4.) ;ADDRESS OF SECTOR 4
         .WORD BUFP+(520.*5.) ;ADDRESS OF SECTOR 5
         .WORD BUFP+(520.*6.) ;ADDRESS OF SECTOR 6
         .WORD BUFP+(520.*7.) ;ADDRESS OF SECTOR 7
         .WORD BUFP+(520.*8.) ;ADDRESS OF SECTOR 8
         .WORD BUFP+(520.*9.) ;ADDRESS OF SECTOR 9
         .WORD BUFP+(520.*10.) ;ADDRESS OF SECTOR 10
         .WORD BUFP+(520.*11.) ;ADDRESS OF SECTOR 11
         .WORD BUFP+(520.*12.) ;ADDRESS OF SECTOR 12
```

```
001354 000
001355 000
001356 000
001357 000
001360 000000
001362 000000
001364 000
001365 000
001366 000000
001370 001304
001372 000000

001374 001426 000000 001224
001402 001441 000022 001230
001410 001454 000000 001222
001416 001465 000022 001226
001424 000000

052123 051101 020124
054503 020114 000
001441 123 040524 052122
001446 052040 045522 000040
001454 047105 020104 054503
001462 020114 000
001465 105 042116 052040
001472 045522 000040

025130
026140
027150
030160
031170
032200
033210
034220
035230
036240
037250
040260
041270
```

596	001530	042300	.WORD	BUFP+(520.*13.)	; ADDRESS OF SECTOR 13
597	001532	043310	.WORD	BUFP+(520.*14.)	; ADDRESS OF SECTOR 14
598	001534	044320	.WORD	BUFP+(520.*15.)	; ADDRESS OF SECTOR 15
599	001536	045330	.WORD	BUFP+(520.*16.)	; ADDRESS OF SECTOR 16
600	001540	046340	.WORD	BUFP+(520.*17.)	; ADDRESS OF SECTOR 17
601	001542	047350	.WORD	BUFP+(520.*18.)	; ADDRESS OF SECTOR 18
602	001544	050360	.WORD	BUFP+(520.*19.)	; ADDRESS OF SECTOR 19
603	001546	051370	.WORD	BUFP+(520.*20.)	; ADDRESS OF SECTOR 20
604	001550	052400	.WORD	BUFP+(520.*21.)	; ADDRESS OF SECTOR 21

; REMAINING WORD COUNT TABLE

605					
606					
607					
608	001552	000404	WCTBL: .WORD	260.	; REMAINING WORD COUNT AFTER SECTOR 0
609	001554	001010	.WORD	260.+(260.*1.)	; REMAINING WORD COUNT AFTER SECTOR 1
610	001556	001414	.WORD	260.+(260.*2.)	; REMAINING WORD COUNT AFTER SECTOR 2
611	001560	002020	.WORD	260.+(260.*3.)	; REMAINING WORD COUNT AFTER SECTOR 3
612	001562	002424	.WORD	260.+(260.*4.)	; REMAINING WORD COUNT AFTER SECTOR 4
613	001564	003030	.WORD	260.+(260.*5.)	; REMAINING WORD COUNT AFTER SECTOR 5
614	001566	003434	.WORD	260.+(260.*6.)	; REMAINING WORD COUNT AFTER SECTOR 6
615	001570	004040	.WORD	260.+(260.*7.)	; REMAINING WORD COUNT AFTER SECTOR 7
616	001572	004444	.WORD	260.+(260.*8.)	; REMAINING WORD COUNT AFTER SECTOR 8
617	001574	005050	.WORD	260.+(260.*9.)	; REMAINING WORD COUNT AFTER SECTOR 9
618	001576	005454	.WORD	260.+(260.*10.)	; REMAINING WORD COUNT AFTER SECTOR 10
619	001600	006060	.WORD	260.+(260.*11.)	; REMAINING WORD COUNT AFTER SECTOR 11
620	001602	006464	.WORD	260.+(260.*12.)	; REMAINING WORD COUNT AFTER SECTOR 12
621	001604	007070	.WORD	260.+(260.*13.)	; REMAINING WORD COUNT AFTER SECTOR 13
622	001606	007474	.WORD	260.+(260.*14.)	; REMAINING WORD COUNT AFTER SECTOR 14
623	001610	010100	.WORD	260.+(260.*15.)	; REMAINING WORD COUNT AFTER SECTOR 15
624	001612	010504	.WORD	260.+(260.*16.)	; REMAINING WORD COUNT AFTER SECTOR 16
625	001614	011110	.WORD	260.+(260.*17.)	; REMAINING WORD COUNT AFTER SECTOR 17
626	001616	011514	.WORD	260.+(260.*18.)	; REMAINING WORD COUNT AFTER SECTOR 18
627	001620	012120	.WORD	260.+(260.*19.)	; REMAINING WORD COUNT AFTER SECTOR 19
628	001622	012524	.WORD	260.+(260.*20.)	; REMAINING WORD COUNT AFTER SECTOR 20
629	001624	013130	.WORD	260.+(260.*21.)	; REMAINING WORD COUNT AFTER SECTOR 21

630
631 .EVEN
632

.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

001626

\$ERRTB:
:ERROR 1

001626 022112
001630 023231
001632 024502
001634 025014

EM1
DH1
DT1
DF1

:RH11 INTERRUPT OCCURRED (RPAS=0)

:ERROR 2

001636 022153
001640 023236
001642 024504
001644 025020

EM2
DH2
DT2
DF2

:UNEXPECTED ATTENTION OCCURRED

:ERROR 3

001646 022211
001650 023233
001652 024520
001654 025024

EM3
DH3
DT3
DF3

:MASSBUS PARITY ERROR (MCPPE=1)

:ERROR 4

001656 022247
001660 023341
001662 024526
001664 025030

EM4
DH4
DT4
DF4

:MASSBUS PARITY ERROR (PAR=1)

:ERROR 5

001666 022304
001670 023236
001672 024504
001674 025020

EM5
DH2
DT2
DF2

:ADDRESS PLUG CHANGE BIT SET

:ERROR 6

001676 022340
001700 023400

EM6
DH6

:RH11 DIDN'T RESPOND TO ADDRESSING

001678 022304
001680 023236
001682 024504
001684 025020
001686 022340
001688 023400
001690 022304
001692 023236
001694 024504
001696 025020
001698 022340
001700 023400

66	001702	024536	D16	
67	001704	025014	DF1	
68				
69				
70				
71				
72				
73				
74				
75				
76				
77				
78				
79				
80				
81				
82				
83				
84				
85				
86				
87				
88				
89				
90				
91				
92				
93				
94				
95				
96				
97				
98				
99				
00				
01				
02				
03				
04				
05				
06				
07				
08				
09				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				
27				
28				
29				
30				
31				
32				
33				
34				
35				
36				
37				
38				
39				
40				
41				
42				
43				
44				
45				
46				
47				
48				
49				
50				
51				
52				
53				
54				
55				
56				
57				
58				
59				
60				
61				
62				
63				
64				
65				
66				
67				
68				
69				
70				
71				
72				
73				
74				
75				
76				
77				
78				
79				
80				
81				
82				
83				
84				
85				
86				
87				
88				
89				
90				
91				
92				
93				
94				
95				
96				
97				
98				
99				
00				
01				
02				
03				
04				
05				
06				
07				
08				
09				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				
27				
28				
29				
30				
31				
32				
33				
34				
35				
36				
37				
38				
39				
40				
41				
42				
43				
44				
45				
46				
47				
48				
49				
50				
51				
52				
53				
54				
55				
56				
57				
58				
59				
60				
61				
62				
63				
64				
65				
66				
67				
68				
69				
70				
71				
72				
73				
74				
75				
76				
77				
78				
79				
80				
81				
82				
83				
84				
85				
86				
87				
88				
89				
90				
91				
92				
93				
94				
95				
96				
97				
98				
99				
00				
01				
02				
03				
04				
05				
06				
07				
08				
09				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				
27				
28				
29				
30				
31				
32				
33				
34				
35				
36				
37				
38				
39				
40				
41				
42				
43				
44				
45				
46				
47				
48				
49				
50				
51				
52				
53				
54				
55				
56				
57				
58				
59				
60				
61				
62				
63				
64				
65				
66				
67				
68				
69				
70				
71				
72				
73				
74				
75				
76				
77				
78				
79				
80				
81				
82				
83				
84				
85				
86				
87				
88				
89				
90				
91				
92				
93				
94				
95				
96				
97				
98				
99				
00				
01				
02				
03				
04				
05				
06				
07				
08				
09				
10				
11				


```

849 002362 005227 177777      INC      #-1      ;CHECK FOR FIRST START
850 002366 001010      BNE      SETVEC   ;BR IF NOT
851 002370 004737 025434      JSR      PC,BUSADR ;CHECK THE RH11 ADDRESS
852 002374 013737 001172 012246      MOV      $RPADR,RPADR ;RH11 ADDRESS
853 002402 013737 001174 012250      MOV      $RPVEC,RPVEC ;RH11 VECTOR ADDRESS
854
855      ;DISPLAY DRIVE STATUS AND SET UP THE OTHER UNITS THE
856      ;PROGRAM WILL USE
857
858 002410 004737 005742      SETVEC: JSR      PC,ST.CLK ;START THE CLOCK
859 002414 004737 012264      JSR      PC,RPINIT ;INITIALIZE THE RPO4/5/6 DRIVER
860 002420 012737 177777 012206      MOV      #-1,SAVEFG ;SET THE SAVE REGISTERS FLAG
861 002426 005227 177777      INC      #-1      ;SEE IF FIRST START
862 002432 001404      BEQ      11$      ;BR IF YES
863 002434 032777 000004 176476      BIT      #SW02,JSWR ;TYPEOUT THE DRIVE STATUS TABLE ?
864 002442 001076      BNE      10$      ;BR IF NOT
865 002444 012737 000340 177776 11$:      MOV      #PR7,PS ;SET PRIORITY TO 7
866 002452 005004      CLR      R4      ;DRIVE TABLE POINTER
867 002454 104401 001167      TYPE    .$CRLF ;CR-LF
868 002460 104401 020640      TYPE    .SYSTAT ;TYPE STATUS HEADING
869
870 002464 010446      1$:      MOV      R4,-(SP) ;SAVE R4 FOR TYPEOUT
871
872      ;TYPE DRIVE NUMBER
873      ;GO TYPE--OCTAL ASCII
874      ;TYPE 2 DIGIT(S)
875      ;SUPPRESS LEADING ZEROS
876      ;SPACES
877      ;CHECK DRIVE'S STATUS
878      ;BR IF UNSAFE
879      ;BR IF ONLINE
880      ;SEE IF OFFLINE OR NONEXISTENT
881      ;BR IF NONEXISTENT
882      ;BR IF OFFLINE
883      ;DRIVE NOT AN RPO4/5/6
884      ;CHECK NEXT DRIVE
885      ;DRIVE NOT PRESENT
886      ;CHECK NEXT DRIVE
887      ;DRIVE OFFLINE
888      ;PRINT DRIVE TYPE
889      ;DRIVE UNSAFE
890      ;PRINT DRIVE TYPE
891      ;DRIVE ONLINE
892      ;SPACES
893      ;ADDRESS OF RPO4 MESSAGE
894      ;RPO4 ?
895      ;BR IF YES
896      ;ADDRESS OF RPO5 MESSAGE
897      ;RPO5 ?
898      ;BR IF YES
899      ;ADDRESS OF RPO6 MESSAGE
900      ;TYPE THE DRIVE TYPE MESSAGE
901      ;MESSAGE ADDRESS HERE
902      ;CR-LF
903      ;INCREMENT DRIVE NUMBER/TABLE POINTER
    
```

```

903 002632 020427 000010      CMP      R4,#8.      ;FINISHED ?
904 002636 001312      BNE      1$          ;BR IF NOT
905 002640 104401 001167      10$:    TYPE      ,SCLF      ;CR-LF
906
907
908      ;SEE WHICH MODE THE PROGRAM IS TO BE RUN IN.
909      ;MODES ARE: 'FORMAT & VERIFY' OR 'CHECK FORMAT'
910
911 002644 005037 001220      M1:    CLR      MODE      ;SET MODE TO 'CHECK FORMAT'
912 002650 104401 021121      TYPE      ,MMODE      ;TYPE 'PROGRAM MODE'
913 002654 104411      RDLIN      ;READ THE KEYBOARD
914 002656 012601      MOV      (SP)+,R1      ;GET ADDRESS OF INPUT
915 002660 105711      TSTB     (R1)          ;'CR' ENTERED (DEFAULT)
916 002662 001414      BEQ      2$          ;BR IF YES
917 002664 122711 000103      CMPB     #'C,(R1)      ;'CHECK' ?
918 002670 001406      BEQ      1$          ;BR IF YES
919 002672 122711 000106      CMPB     #'F,(R1)      ;'FORMAT' ?
920 002676 001411      BEQ      3$          ;BR IF YES
921 002700 104401 001166      TYPE      ,SQUES      ;NO CORRECT ENTRY
922 002704 000757      BR      M1          ;TRY AGAIN
923 002706 104401 021174      1$:    TYPE      ,MHECK      ;TYPE REST OF 'CHECK'
924 002712 000410      BR      M1A         ;GET STARTING ADDRESS
925 002714 104401 021153      2$:    TYPE      ,MFORMT      ;TYPE DEFAULT MESSAGE
926 002720 000402      BR      4$          ;SET UP MODE
927 002722 104401 021207      3$:    TYPE      ,MORMAT      ;TYPE REST OF 'FORMAT'
928 002726 012737 177777 001220 4$:    MOV      #-1,MODE      ;SET MODE TO 'FORMAT & VERIFY'
929
930      ;FIND OUT IF FORMAT IS TO BE IN 20 OR 22 SECTOR MODE
931
932 002734 104401 021227      M1A:   TYPE      ,MSIZE      ;TYPE FORMAT MODE REQUEST
933 002740 104411      RDLIN      ;READ THE KEYBOARD
934 002742 012601      MOV      (SP)+,R1      ;ADDRESS OF ENTRY
935 002744 122711 000131      CMPB     #'Y,(R1)      ;IS ENTRY A 'Y' ?
936 002750 001410      BEQ      1$          ;BR IF IT IS
937 002752 122711 000116      CMPB     #'N,(R1)      ;IS ENTRY A 'N' ?
938 002756 001424      BEQ      2$          ;BR IF IT IS
939 002760 105711      TSTB     (R1)          ;DEFAULT MODE ?
940 002762 001403      BEQ      1$          ;BR IF IT IS
941 002764 104401 001156      TYPE      ,SQUES      ;TYPE '?'
942 002770 000761      BR      M1A         ;TRY AGAIN
943 002772 104401 021300      1$:    TYPE      ,MSEC22      ;TYPEOUT MODE SELECTED
944 002776 012737 013130 001256      MOV      #<260.*22.>,WC ;TRACK SIZE IN 22 SECTOR MODE
945 003004 012737 164650 001260      MOV      #-<260.*22.>,MWC ;2'S COMPLEMENT WORD COUNT
946 003012 012737 177777 001264      MOV      #-1,SEC20      ;22 SECTOR INDICATOR
947 003020 012737 000025 001266      MOV      #21.,MAXSEC      ;MAX SECTOR ADDRESS IN 22 SECTOR MODE
948 003026 000415      BR      M1B         ;CONTINUE
949 003030 104401 021357      2$:    TYPE      ,MSEC20      ;TYPE OUT 20 SECTOR MODE SELECTED
950 003034 012737 012120 001256      MOV      #<260.*20.>,WC ;TRACK SIZE IN 20 SECTOR MODE
951 003042 012737 165660 001260      MOV      #-<260.*20.>,MWC ;2'S COMPLEMENT WORD COUNT
952 003050 005037 001264      CLR      SEC20      ;20 SECTOR INDICATOR
953 003054 012737 000023 001266      MOV      #19.,MAXSEC      ;MAX SECTOR ADDRESS IN 20 SECTOR MODE
954 003062 005037 001230      M1B:   CLR      BEGTRK      ;CLEAR STARTING TRACK ADDRESS
955 003066 005037 001224      CLR      BEGCYL      ;CLEAR BEGINNING CYLINDER ADDRESS
956 003072 012737 000022 001226      MOV      #18.,ENDTRK      ;SETUP END TRACK ADDRESS

```



```

957 003100 012737 000002 001240      MOV      #2,PATSEL      ;SETUP FOR WORST CASE PATTERN
959 003106 112737 177777 001354      MOV      #-1,FMTDPB   ;SETUP INVALID DRIVE NUMBER
959 003114 000400                BR        MO           ;BRANCH TO START
960
961                ;GO FIND OUT WHAT DRIVE
962
963 003116 012737 006116 001300  MO:      MOV      #0ENTER,CNTLC ;CONTROL C ABORT ENTRANCE
964 003124 005037 001112                CLR      $ERTTL       ;CLEAR THE ERROR ACCUMULATOR
965 003130 004737 010204                JSR      PC,STKINT    ;INITIALIZE THE TTY KEYBOARD
966 003134 104401 020657                TYPE    ,MUNIT       ;ASK FOR DRIVE NUMBER
967 003140 104411                RDLIN                    ;READ THE KEYBOARD
968 003142 012601                MOV      (SP)+,R1     ;ADDRESS OF ENTRY
969 003144 004537 006426                JSR      R5,CK.CHR   ;CHECK ONE CHARACTER
970 003150 003176                3$
971 003152 003164                2$
972 003154 003176                3$
973 003156 003164                2$
974 003160 003204                4$
975 003162 003176                3$
976 003164 005037 001214                2$:      CLR      DRIVE      ;SELECT DRIVE ZERO
977 003170 104401 020701                TYPE    ,MDRVD       ;GO TYPE DEFAULT DRIVE NUMBER
978 003174 000413                BR        $S          ;GO SEE IF DRIVE ZERO IS THERE
979 003176 104401 001166                3$:      TYPE    ,SQUES    ;TYPE '?'
980 003202 000745                BR        MO           ;ASK FOR DRIVE NUMBER AGAIN
981 003204 010237 001214                4$:      MOV      R2,DRIVE  ;SAVE DRIVE NUMBER
982 003210 005702                TST     R2           ;SEE IF DRIVE 0
983 003212 001004                BNE     $S          ;BR IF NOT
984 003214 122737 000011 000041      CMPB   #11,41       ;PROGRAM LOADED FROM AN RPO4/5/6 ?
985 003222 001431                BEQ     7$          ;BR IF IT WAS, CAN'T FORMAT THE DRIVE
986 003224 004737 005742                5$:      JSR      PC,ST.CLK   ;START THE CLOCK
987 003230 004737 012264                JSR      PC,RPINIT   ;GO SEE WHAT DRIVES ARE AVAILABLE
988 003234 012737 177777 012206      MOV     #-1,SAVEFG  ;SAVE THE REGISTERS
989 003242 012737 177777 012210      MOV     #-1,SEEKFG  ;SET 'NO OPTIMIZATION' FLAG
990 003250 005037 177776                CLR     PS          ;SET PRIORITY BACK TO ZERO
991 003254 105762 012120                TSTB   DRVSTA(R2)  ;LOOK AT DRIVE STATUS
992 003260 003015                BGT     8$          ;BRANCH IF ONLINE
993 003262 001403                BEQ     6$          ;BR IF DRIVE NOT AVAILABLE
994 003264 104401 021067                TYPE    ,MUSDR      ;'DRIVE UNSAFE'
995 003270 000712                BR        MO           ;GO GET DRIVE NUMBER AGAIN
996 003272 105762 012130                6$:      TSTB   DRVSTYP(R2);A DRIVE PRESENT?
997 003276 001003                BNE     7$          ;BR IF SO
998 003300 104401 020762                TYPE    ,MDRNP      ;TYPE 'DRIVE NOT PRESENT'
999 003304 000704                BR        MO           ;GO GET DRIVE NUMBER AGAIN
1000 003306 104401 021007                7$:      TYPE    ,MER11     ;'DRIVE NOT AVAILABLE'
1001 003312 000701                BR        MO           ;GO GET DRIVE NUMBER AGAIN
1002 003314 123737 001214 001354      8$:      CMPB   DRIVE,FMTDPB ;SAME DRIVE AS LAST TIME ?
1003 003322 001430                BEQ     M2          ;BR IF IT IS
1004 003324 113737 001214 001354      MOV     DRIVE,FMTDPB ;SETUP DRIVE ADDRESS
1005 003332 012737 000632 001222      MOV     #410.,ENDCYL ;SETUP FOR RPO4/5
1006 003340 132762 000003 012130      BITB   #BIT00!BIT01,DRVSTYP(R2) ;SEE IF DRIVE RPO4/5
1007 003346 001003                BNE     9$          ;BR IF EITHER
1008 003350 012737 001456 001222      MOV     #814.,ENDCYL ;SETUP ENDING CYLINDER FOR RPO6
1009 003356 005037 001230                9$:      CLR     BEGTRK     ;CLEAR STARTING TRACK ADDRESS
1010 003362 005037 001224                CLR     BEG0YL     ;CLEAR STARTING CYLINDER ADDRESS

```

```

1011 003366 013737 001222 001376      MOV      ENDCYL, TABLE+2 ; ENTRY LIMIT FOR BEGINNING CYLINDER
1012 003374 013737 001222 001412      MOV      ENDCYL, TABLE+16 ; ENTRY LIMIT FOR END CYLINDER
1013 003402 000400                BR        M2 ; GET ADDRESS LIMITS FROM THE OPERATOR
1014
1015 ; GET ADDRESS LIMITS
1016
1017 003404 104401 020710      M2:     TYPE      ENTADR ; 'ENTER ADDRESS LIMITS'
1018 003410 004737 006236      JSR      PC, PARENT ; GET THE ADDRESS LIMITS
1019 003414 023737 001222 001224      CMP      ENDCYL, BEGCYL ; SEE IF ENDING CYLINDER EQ TO GT THAN BEGINNING
1020 003422 101010                BHI      M4 ; BR IF HIGHER
1021 003424 103404                BLO      3$ ; BR IF LESS
1022 003426 023737 001226 001230 2$:     CMP      ENDTRK, BEGTRK ; SEE IF ENDING TRACK EQ OR GT THAN BEGINNING
1023 003434 103003                BHIS     M4 ; BR IF YES
1024 003436 104401 021456      3$:     TYPE      ,MADRER ; INVALID ADDRESS ENTERED
1025 003442 000760                BR        M2 ; TRY AGAIN
1026
1027 ; GO GET DATA PATTERN FOR FORMAT
1028
1029 003444 104401 021560      M4:     TYPE      ,MSELP ; GO TYPE 'SELECT PATTERN'
1030 003450 104411                RDLIN ; READ THE KEYBOARD
1031 003452 012601                MOV      (SP)+, R1 ; ENTRY ADDRESS
1032 003454 004537 006426      JSR      R5, CK.CHR ; CHECK ONE CHARACTER
1033 003460 003514                3$:     ; ILLEGAL CHARACTER
1034 003462 003474                1$:     ; CARRIAGE RETURN
1035 003464 003514                3$:     ; " "
1036 003466 003474                1$:     ; " "
1037 003470 003506                2$:     ; DIGIT 0-7
1038 003472 003514                3$:     ; DIGIT 8-9
1039 003474 104401 021660      1$:     TYPE      ,MPATD ; TYPE DEFAULT PATTERN
1040 003500 012702 000002      MOV      #2, R2 ; WORST CASE PATTERN
1041 003504 000406                BR        4$ ; GO SAVE PATTERN
1042 003506 020227 000002      2$:     CMP      R2, #2 ; IS # LARGER THAN 2
1043 003512 101403                BLOS     4$ ; BRANCH IF NOT
1044 003514 104401 001166      3$:     TYPE      ,SQUES ; TYPE '?'
1045 003520 000751                BR        M4 ; RETYPE LINE
1046 003522 010237 001240      4$:     MOV      R2, PATSEL ; SAVE PATTERN SELECTED
1047
1048 ; GO TYPE 'STARTING FORMAT ON DRIVE N'
1049
1050 003526 012737 006136 001300 M5:     MOV      #TYPADR, CNTLC ; CHANGE IC ENTRANCE
1051 003534 005737 001220                TST      MODE ; 'FORMAT' OR 'CHECK' MODE ?
1052 003540 001403                BEQ      1$ ; BR IF 'CHECK' MODE
1053 003542 104401 021673                TYPE      ,MSFOU ; TYPE 'STARTING FORMAT ON DRIVE N'
1054 003546 000402                BR        2$
1055 003550 104401 021730      1$:     TYPE      ,MSCHK ; TYPE 'STARTING CHECK ON DRIVE N'
1056 003554
1057 003554 013746 001214      2$:     MOV      DRIVE, -(SP) ; SAVE DRIVE FOR TYPEOUT
1058 ; TYPE DRIVE NUMBER
1059 003560 104403                TYPOS ; GO TYPE--OCTAL ASCII
1060 003562 001                .BYTE 1 ; TYPE 1 DIGIT(S)
1061 003563 000                .BYTE 0 ; SUPPRESS LEADING ZEROS
1062 003564 104401 001167                TYPE      ,SCLF ; CR-LF
1063
1064 ; SETUP TOTAL TRACK COUNT FOR FORMAT
    
```



```

1065
1066 003570 023737 001222 001224 CKADRS: CMP ENDCYL,BEGCYL ;STARTING AND ENDING CYLINDERS THE SAME ?
1067 003576 001011 BNE 1$ ;BRANCH IF BEG CYL NOT EQUAL TO ENDCYL
1068 003600 013737 001226 001232 MOV ENDRK,TTRKS ;END TRACK ADDRESS
1069 003606 163737 001230 001232 SUB BEGTRK,TTRKS ;SUBTRACT THE STARTING TRACK ADDRESS
1070 003614 005237 001232 INC TTRKS ;MAKE THE NUMBER OF TRACKS INCLUSIVE
1071 003620 000427 SR SETPAT ;SETUP THE DATA PATTERN
1072 003622 013702 001222 1$: MOV ENDCYL,R2 ;ENDING CYLINDER
1073 003626 163702 001224 SUB BEG CYL,R2 ;SUBTRACT THE STARTING CYLINDER
1074 003632 013737 001226 001232 MOV ENDRK,TTRKS ;CALCULATE THE RESIDUAL TRACKS
1075 003640 163737 001230 001232 SUB BEGTRK,TTRKS ;SUBTRACT THE STARTING TRACK
1076 003646 100004 BPL 2$ ;BR IF ENDING TRACK GREATER
1077 003650 062737 000023 001232 ADD #19.,TTRKS ;CORRECT THE VALUE
1078 003656 005302 DEC R2 ;COUNT THE PARTIAL TRACK
1079 003660 005237 001232 2$: INC TTRKS ;MAKE THE NUMBER INCLUSIVE
1080 003664 005302 3$: DEC R2 ;DECREMENT THE CYLINDER COUNT
1081 003666 100404 BMI SETPAT ;BR IF DONE
1082 003670 062737 000023 001232 ADD #19.,TTRKS ;ADD CYLINDER WORTH OF TRACKS
1083 003676 000772 BR 3$ ;CONTINUE
    
```

;THIS CODE SETS UP THE SELECTED DATA PATTERN TO BE WRITTEN ON EACH SECTOR IMAGE

```

1084
1085
1086
1087 003700 005037 001242 SETPAT: CLR PATA ;CLEAR DATA PATTERN A
1088 003704 005037 001244 CLR PATB ;CLEAR DATA PATTERN B
1089 003710 005737 001240 TST PATSEL ;SEE IF PATTERN OF ONES
1090 003714 001416 BEQ 1$ ;BR IF SO
1091 003716 005137 001242 COM PATA ;SET PATA TO ONES
1092 003722 005137 001244 COM PATB ;SET PATB TO ONES
1093 003726 022737 000001 001240 CMP #1,PATSEL ;SEE IF PATTERN OF ZEROS
1094 003734 001406 BEQ 1$ ;BRANCH IF SO
1095 003736 012737 165555 001242 MOV #165555,PATA ;SET UP WORST CASE
1096 003744 012737 133333 001244 MOV #133333,PATB ;SET UP WORST CASE
1097 003752 013703 001256 1$: MOV WC,R3 ;SET UP COUNTER
1098 003756 012700 025130 MOV #BUFP,R0 ;SET UP MEMORY POINTER
1099 003762 013701 001242 MOV PATA,R1 ;SET UP PATTERN IN R1
1100 003766 013702 001244 MOV PATB,R2 ;SET UP PATTERN IN R2
1101 003772 010120 2$: MOV R1,(R0)+ ;MOV 1ST PAT INTO MEM
1102 003774 010220 MOV R2,(R0)+ ;MOV 2ND PAT INTO MEM
1103 003776 005303 DEC R3 ;KEEP COUNT
1104 004000 005303 DEC R3 ;KEEP COUNT
1105 004002 001373 BNE 2$ ;DO IT AGAIN IF R3 NOT 0
    
```

;THIS CODE SETS UP FOR THE ACTUAL FORMAT

```

1106
1107
1108
1109 004004 004737 005402 WRTRK: JSR PC,SETTBL ;GO SET UP DRIVER TABLE
1110 004010 004737 005602 JSR PC,SETHDR ;GO INITIALIZE HEADERS IN THE SECTOR BUFFER IN CORE
1111 004014 112737 000020 001355 MOVB #20,FMTDPB+1 ;LOAD FMT22 BIT
1112 004022 005737 001264 TST SEC20 ;18 BIT MODE ?
1113 004026 001002 BNE 1$ ;BR IF NOT
1114 004030 105037 001355 CLRB FMTDPB+1 ;CLEAR THE FMT22 BIT
1115 004034 112737 000143 001356 1$: MOVB #SETFMT,FMTDPB+2 ;'LOAD FORMAT' COMMAND
1116 004042 004037 013034 2$: JSR RO,RPO4 ;START THE COMMAND
1117 004046 001354 FMTDPB ;DPB ADDRESS
1118 004050 000774 BR 2$ ;QUEUE FULL RETURN
    
```

```

1119 004052 005737 001372      3$:   TST      FMTDPB+16      ;LOOK FOR DONE
1120 004056 001775                BEQ      3$              ;LOOP UNTIL FINISHED
1121 004060 005037 001216      WRTRK1: CLR      SOFSW        ;CLEAR ERROR COUNTER
1122 004064 005037 001250                CLR      RETRY          ;ZERO THE RETRY COUNTER
1123 004070 105037 001364                CLR      FMTDPB+10     ;RESTORE SECTOR
1124 004074 013737 001260 001360      MOV      MWC,FMTDPB+4   ;RESTORE WC
1125 004102 012737 025130 001362      MOV      #BUFP,FMTDPB+6 ;RESTORE BA
1126 004110 005737 001220                TST      MODE           ;'FORMAT' OR 'CHECK' MODE ?
1127 004114 001450                BEQ      CKTRK          ;BR IF 'CHECK' MODE
1128 004116 112737 000163 001356      WRTRK2: MOV      #WRTHD,FMTDPB+2 ;SET WRITE HEADER & DATA COMMAND IN TBL
1129 004124 012737 004124 001110      MOV      #,$LPERR      ;SETUP LOOP ON ERROR ADDRESS
1130 004132 012706 001100                MOV      #STACK,SP     ;LOAD STACK POINTER
1131 004136 004037 013034      1$:   JSR      RD,RPO4      ;GO FORMAT A TRACK
1132 004142 001354                FMTDPB   ;ADRS OF PARAMETERS - TBL
1133 004144 000774                BR      1$             ;WAIT FOR QUEUE IF FULL
1134 004146 005737 001372      2$:   TST      FMTDPB+16     ;WAIT FOR COMMAND TO COMPLETE
1135 004152 001775                BEQ      2$             ;BRANCH IF NOT DONE
1136 004154 100024                BPL      4$             ;BRANCH IF NO ERROR
1137 004156 012737 004204 001160      MOV      #3$, $ESCAPE   ;ESCAPE TO 3$ ON ERROR
1138 004164 004737 005234                JSR      PC,ERINDX     ;SEE WHICH ERROR
1139 004170 104010                ERROR   10             ;DRIVE OFFLINE
1140 004172 104011                ERROR   11             ;PERSISTENT DRIVE UNSAFE ERROR
1141 004174 104012                ERROR   12             ;UNCORRECTABLE MASSBUS PARITY ERROR
1142 004176 104013                ERROR   13             ;SOFTWARE TIMEOUT
1143 004200 104014                ERROR   14             ;DRIVE UNSAFE ERROR
1144 004202 104015                ERROR   15             ;DRIVE/CONTROLLER ERROR DURING WRITE
1145 004204 004737 005334      3$:   JSR      PC,LOP.CK    ;LOOP ON THE ERROR ?
1146 004210 022737 000003 001250      CMP      #3,RETRY      ;ERROR RETRY LIMIT ?
1147 004216 001403                BEQ      4$             ;BR IF REACHED
1148 004220 005237 001250                INC      RETRY          ;COUNT THE ERROR
1149 004224 000744                BR      1$             ;TRY AGAIN
1150 004226 005037 001160      4$:   CLR      $ESCAPE     ;CLEAR ERROR ESCAPE ADDRESS
1151 004232 005037 001250                CLR      RETRY         ;CLEAR THE RETRY COUNTER
1152
1153                ;CHECK THE TRACK JUST WRITTEN
1154
1155 004236 112737 000153 001356      CKTRK: MOV      #WCKHD,FMTDPB+2 ;SET WRITE CHECK HEADER & DATA COMMAND IN TBL
1156 004244 012737 004244 001110      MOV      #,$LPERR      ;SETUP LOOP ON ERROR ADDRESS
1157 004252 012706 001100                MOV      #STACK,SP     ;LOAD STACK POINTER
1158 004256 004037 013034      1$:   JSR      RD,RPO4      ;GO CHECK THE TRACK JUST FORMATTED
1159 004262 001354                FMTDPB   ;ADRS OF PARAMETERS - TBL
1160 004264 000774                BR      1$             ;WAIT FOR QUEUE IF FULL
1161 004266 005737 001372      2$:   TST      FMTDPB+16     ;WAIT FOR COMMAND TO COMPLETE
1162 004272 001775                BEQ      2$             ;BRANCH IF NOT DONE
1163 004274 100112                BPL      8$             ;BRANCH IF NO ERROR
1164 004276 113737 001312 001252      MOV      RP.REG+RPDA,SAVSEC ;GET THE SECTOR ADDRESS
1165 004304 001005                BNE      3$             ;BRANCH IF NOT SECTOR 0
1166 004306 013737 001266 001252      MOV      MAXSEC,SAVSEC ;RESTORE TO LAST SECTOR +1
1167 004314 005237 001252                INC      SAVSEC         ;INCREMENT TO LAST SECTOR +1
1168 004320 005337 001252      3$:   DEC      SAVSEC         ;ADJUST SECTOR TO THE ONE THAT FAILED
1169 004324 012737 004570 001160      MOV      #10$, $ESCAPE ;ESCAPE TO 10$ ON ERROR
1170 004332 004737 005234                JSR      PC,ERINDX     ;SEE WHICH ERROR
1171 004336 104010                ERROR   10             ;DRIVE OFFLINE
1172 004340 104011                ERROR   11             ;PERSISTENT DRIVE UNSAFE ERROR

```


1173	004342	104012			ERROR	12	;UNCORRECTABLE MASSBUS PARITY ERROR	
1174	004344	104013			ERROR	13	;SOFTWARE TIMEOUT	
1175	004346	104014			ERROR	14	;DRIVE UNSAFE ERROR	
1176	004350	032737	040000	001314	BIT	#WCE,RP.REG+RPCS2	;WRITE CHECK ERROR ?	
1177	004356	001005			BNE	4\$;BR IF SET	
1178	004360	033727	001320		BIT	RP.REG+RPER1,(PC)+	;CHECK FOR DATA ERRORS	
1179	004364	130620			.WORD	DCK!OFI!DTE!HCRC!HCE!FER	;DATA ERROR BITS	
1180	004366	001001			BNE	4\$;BR IF SET	
1181	004370	104016			ERROR	16	;CONTROLLER/DRIVE ERROR DURING WRITE CHECK	
1182	004372	005737	001220	4\$:	TST	MODE	;FORMAT OR CHECK MODE ?	
1183	004376	001407			BEQ	5\$;BR IF CHECK	
1184	004400	005737	001216		TST	SOFSW	;RETRYING THE SECTOR ?	
1185	004404	001404			BEQ	5\$;BR IF NOT	
1186	004406	012737	004612	001160	MOV	#11\$, \$ESCAPE	;ESCAPE TO 11\$ ON ERROR	
1187	004414	104017			ERROR	17	;SECTOR NOT ACCEPTABLE	
1188	004416	005037	001160	5\$:	CLR	\$ESCAPE	;CLEAR THE ERROR ESCAPE ADDRESS	
1189	004422	032737	040000	001316	BIT	#ERR,RP.REG+RPDS1	;DATA ERROR ?	
1190	004430	001002			BNE	6\$;BR IF IT IS	
1191	004432	104024			ERROR	24	;WRITE CHECK ERROR	
1192	004434	000401			BR	7\$;CONTINUE	
1193	004436	104020			ERROR	20	;DATA ERROR DURING WRITE CHECK	
1194	004440	013701	001252	7\$:	MOV	SAVSEC,R1	;FAILING SECTOR ADDRESS	
1195	004444	113737	001252	001364	MOVB	SAVSEC,FMTDPB+10	;SECTOR ADDRESS	
1196	004452	006301			ASL	R1	;SETUP INDEX TO ADDRESS WORDS	
1197	004454	016137	001476	001362	MOV	ADRTBL(R1),FMTDPB+6	;BUFFER ADDRESS FOR FAILING SECTOR	
1198	004462	013746	001260		MOV	MWC,-(SP)	;GET MAXIMUM WORD COUNT VALUE (2'S COMP)	
1199	004466	066116	001552		ADD	WCTBL(R1),(SP)	;ADD WORD COUNT THROUGH FAILING SECTOR	
1200	004472	012637	001254		MOV	(SP)+,SAVWC	;STORE WORD COUNT FOR REMAINDER OF TRACK	
1201	004476	005737	001220		TST	MODE	;FORMAT OR CHECK MODE ?	
1202	004502	001421			BEQ	9\$;BR IF CHECK	
1203	004504	012737	177374	001360	MOV	#-260.,FMTDPB+4	;WORD COUNT FOR 1 SECTOR	
1204	004512	005237	001216		INC	SOFSW	;INDICATE THAT A RETRY IS IN PROGRESS	
1205	004516	000137	004116		JMP	WRTRK2	;REFORMAT ERROR SECTOR	
1206	004522	005737	001216	8\$:	TST	SOFSW	;RETRY IN PROGRESS ?	
1207	004526	001431			BEQ	11\$;BR IF NOT	
1208	004530	022737	000002	001216	CMP	#2,SOFSW	;SEE IF LAST RETRY	
1209	004536	001403			BEQ	9\$;BR IF IT IS	
1210	004540	005237	001216		INC	SOFSW	;INCREMENT RETRY COUNT	
1211	004544	000644			BR	1\$;READ AGAIN	
1212	004546	123737	001266	001364	9\$:	CMPB	MAXSEC,FMTDPB+10	;SEE IF LAST SECTOR ON THE TRACK
1213	004554	001416			BEQ	11\$;BR IF IT IS	
1214	004556	004737	005554		JSR	PC,SCAWC	;SETUP TO CHECK REMAINING SECTORS ON THE TRACK	
1215	004562	005037	001216		CLR	SOFSW	;CLEAR RETRY COUNTER	
1216	004566	000633			BR	1\$;FINISH CHECKING THE TRACK	
1217	004570	004737	005334	10\$:	JSR	PC,LOP.CK	;CHECK FOR LOOP ON ERROR	
1218	004574	022737	000003	001250	CMP	#3,RETRY	;ERROR RETRY REACHED ?	
1219	004602	001403			BEQ	11\$;BR IF IT IS	
1220	004604	005237	001250		INC	RETRY	;COUNT THE RETRY	
1221	004610	000622			BR	1\$;DO THE WRITE CHECK AGAIN	
1222	004612	005037	001160	11\$:	CLR	\$ESCAPE	;CLEAR THE ERROR RETURN ESCAPE ADDRESS	
1223	004616	005037	001250		CLR	RETRY	;CLEAR THE RETRY COUNTER	
1224	004622	032777	000002	174310	BIT	#BIT1,\$SWR	;LOOP ON CURRENT TRACK ?	
1225	004630	001402			BEQ	12\$;BR IF NOT	
1226	004632	000137	004060		JMP	WRTRK1	;START AGAIN ON THE SAME TRACK	


```

1281                                     ;*IF THERES A MONITOR GO TO IT
1282                                     ;*IF THERE ISN'T JUMP TO DONE
1283
1284 005100                               $EOP:
1285 005100 005737 001220                 TST     MODE          ;'FORMAT' OR 'CHECK' MODE ?
1286 005104 001403                       BEQ     3$             ;BR IF 'CHECK'
1287 005106 104401 021764                 TYPE   MFCMPT        ;'FORMAT COMPLETE'
1288 005112 000402                       BR      4$           ;FINISH THE END MESSAGE
1289 005114 104401 022011                 3$:    TYPE   MCCMPT  ;'CHECK COMPLETE'
1290 005120 104401 022035                 4$:    TYPE   NUMERR  ;'TOTAL ERRORS DETECTED: '
1291 005124 013746 001112                 MOV     $ERTTL, -(SP) ;SAVE $ERTTL FOR TYPEOUT
1292 005130 104405                       TYPDS  ;              ;GO TYPE--DECIMAL ASCII WITH SIGN
1293 005132 104401 001167                 TYPE   $CRLF        ;CR-LF
1294 005136 005237 001100                 INC     $PASS        ;INCREMENT THE PASS NUMBER
1295 005142 042737 100000 001100         BIC     #100000, $PASS ;DON'T ALLOW A NEG. NUMBER
1296 005150 005327                       DEC     (PC)+        ;LOOP?
1297 005152 000001                       $EOPCT: .WORD 1
1298 005154 003013                       BGT     $DOAGN       ;: YES
1299 005156 012737                       MOV     (PC)+, a(PC)+ ;: RESTORE COUNTER
1300 005160 000001                       $ENDCT: .WORD 1
1301 005162 005152                       $EOPCT
1302 005164 013700 000042                 $GET42: MOV     a#42, R0 ;: GET MONITOR ADDRESS
1303 005170 001405                       BEQ     $DOAGN       ;: BRANCH IF NO MONITOR
1304 005172 000005                       RESET   ;: CLEAR THE WORLD
1305 005174 004710                       $ENDAD: JSR     PC, (R0) ;: GO TO MONITOR
1306 005176 000240                       NOP     ;: SAVE ROOM
1307 005200 000240                       NOP     ;: FOR
1308 005202 000240                       NOP     ;: ACT11
1309 005204
1310 005204 000137                       $DOAGN: JMP     a(PC)+   ;: RETURN
1311 005206 005210                       $RTNAD: .WORD  DONE
1312 005210 032777 000001 173722         -DONE: .BIT    #SW0, aSWR ;SEE IF SWR 0 SET
1313 005216 001002                       BNE     1$           ;BR IF SET
1314 005220 000137 003116                       JMP     MO            ;ASK FOR NEXT DRIVE
1315 005224 012706 001100                 1$:    MOV     #STACK, SP ;RESET STACK POINTER
1316 005230 000137 003526                       JMP     M5           ;DO THE FORMAT OR CHECK AGAIN
1317
1318                                     ;:*****
1319
1320 .SBTTL  SUPPORT SUBROUTINES
1321
1322                                     ;:*****
1323
1324 ;THIS ROUTINE DETERMINES THE ERROR TYPE
1325
1326 005234 010146                               ERINDX: MOV     R1, -(SP) ;STORE R1
1327 005236 005001                               CLR     R1           ;CLEAR R1
1328 005240 033727 001372                 BIT     FMTDPB+16, (PC)+ ;CHECK ERROR TYPE
1329 005244 060006                               .WORD  BIT14!BIT13!BIT2!BIT1 ;DRIVE OFFLINE
1330 005246 001025                               BNE     5$           ;BR IF OFFLINE
1331 005250 033727 001372                 BIT     FMTDPB+16, (PC)+ ;CHECK ERROR TYPE
1332 005254 010000                               .WORD  BIT12        ;DRIVE PERSISTENTLY UNSAFE
1333 005256 001020                               BNE     4$           ;BR IF UNSAFE
1334 005260 033727 001372                 BIT     FMTDPB+16, (PC)+ ;CHECK ERROR TYPE

```

```

1335 005264 006000 .WORD BIT11!BIT10 :UNCORRECTABLE MASSBUS PARITY ERROR ?
1336 005266 001013 BNE 3$ :BR IF PARITY ERROR
1337 005270 033727 001372 BIT FMTDPB+16,(PC)+ :CHECK ERROR TYPE
1338 005274 001400 .WORD BIT09!BIT08 :SOFTWARE TIMEOUT ?
1339 005276 001006 BNE 2$ :BR IF YES
1340 005300 033727 001372 BIT FMTDPB+16,(PC)+ :CHECK ERROR TYPE
1341 005304 000020 .WORD BIT04 :DRIVE UNSAFE ERROR ?
1342 005306 001001 BNE 1$ :BR IF YES
1343 005310 005201 INC R1 :INCREMENT THE RETURN INDEX
1344 005312 005201 INC R1 :INCREMENT THE RETURN INDEX
1345 005314 005201 INC R1 :INCREMENT THE RETURN INDEX
1346 005316 005201 INC R1 :INCREMENT THE RETURN INDEX
1347 005320 005201 INC R1 :INCREMENT THE RETURN INDEX
1348 005322 006301 5$: ASL R1 :DOUBLE THE INCREMENT
1349 005324 060166 000002 ADD R1,2(SP) :DEVELOP THE RETURN ADDRESS
1350 005330 012601 MOV (SP)+,R1 :RESTORE R1
1351 005332 000207 RTS PC :RETURN
    
```

;ROUTINE TO CHECK FOR LOOP ON ERROR

```

1355 005334 032777 001000 173576 LOP.CK: BIT #SW09,JSWR :LOOP ON ERROR ?
1356 005342 001402 BEQ 1$ :BR IF NOT
1357 005344 000177 173540 JMP JSWPERR :GO TO THE LOOP ON ERROR ADDRESS
1358 005350 005037 001160 1$: CLR $ESCAPE :CLEAR THE ERROR ESCAPE ADDRESS
1359 005354 033727 001372 BIT FMTDPB+16,(PC)+ :CHECK FOR 'FATAL' ERROR
1360 005360 072002 .WORD BIT14!BIT13!BIT12!BIT10!BIT01 :'FATAL' ERROR BITS
1361 005362 001004 BNE 2$ :BR IF NOT
1362 005364 032737 004000 001320 BIT #WLE,RP.REG+RPER1 :WRITE LOCK ERROR ?
1363 005372 001402 BEQ 3$ :BR IF NOT
1364 005374 000137 005100 2$: JMP $EOP :TERMINATE THE FORMAT
1365 005400 000207 3$: RTS PC :RETURN
    
```

;THIS ROUTINE SETS UP THE INPUT TABLE FOR THE DRIVER ROUTINE

```

1369 005402 113737 001214 001354 SETTB:L: MOV B DRIVE,FMTDPB :SET UP DRIVE #
1370 005410 105037 001357 CLR B FMTDPB+3 :CLEAR HIGH ORDER ADRS BITS
1371 005414 013737 001260 001360 MOV MWC,FMTDPB+4 :LOAD UP WORD COUNT
1372 005422 012737 025130 001362 MOV #BUFP,FMTDPB+6 :LOAD UP CURRENT ADRS
1373 005430 105037 001364 CLR B FMTDPB+10 :SET SECTOR TO ZERO
1374 005434 113737 001230 001365 MOV B,BEGTRK,FMTDPB+11 :SET UP STARTING TRK ADRS
1375 005442 013737 001224 001366 MOV B,BEGCYL,FMTDPB+12 :SET UP STARTING CYL
1376 005450 005037 001372 CLR FMTDPB+16 :CLEAR RPO4 STATUS
1377 005454 013737 001230 001236 MOV B,BEGTRK,TRKCNT :SET UP PARTIAL CYL TRACK COUNT
1378 005462 013737 001232 001234 MOV TTRKS,TTRKSC :SET UP TOTAL TRACKS COUNTER
1379 005470 000207 RTS PC :RETURN FROM SETUP
    
```

;THIS ROUTINE CONTROLS THE DISK ADDRESSING AND TOTAL TRK COUNT
;IT IS ENTERED AFTER EVERY TRK OPERATION

```

1384 005472 005337 001234 TRKTST: DEC TTRKSC :SEE IF LAST TRACK
1385 005476 001425 BEQ 3$ :BRANCH IF LAST TRACK
1386 005500 022737 000022 001236 CMP #19.,TRKCNT :IS THIS THE LAST TRACK IN THE CYLINDER ?
1387 005506 001407 BEQ 1$ :BRANCH IF SO
1388 005510 005237 001236 INC TRKCNT :COUNT UP TRACK WITHIN CYLINDER
    
```



```

1389 005514 105237 001365      INCB   FMTDPB+11      ;INCREMENT TRACK NUMBER
1390 005520 004737 005712      JSR    PC,UPDATK    ;UPDATE TRACK ADDRESS IN BUFFER
1391 005524 000410                BR     2$           ;EXIT
1392 005526 005037 001236      1$:   CLR    TRKCNT    ;CLEAR TRACK COUNT FOR NEXT CYLINDER
1393 005532 105037 001365      CLRB   FMTDPB+11    ;RESET TRACK ADDRESS TO 0
1394 005536 005237 001366      INC    FMTDPB+12    ;UPDATE CYLINDER NUMBER
1395 005542 004737 005662      JSR    PC,UPDACY    ;UPDATE CYLINDER NUMBER IN BUFFER
1396 005546 062716 000002      2$:   ADD    #2,(SP)  ;ADD TWO TO RETURN ADRS
1397 005552 000207                3$:   RTS     PC      ;RETURN

;THIS ROUTINE SETS UP WC & BA FOR REMAINING SECTORS
;AFTER A WRITE CHECK ERROR

1400
1401
1402 005554 013737 001254 001360 SCANC: MOV    SAVWC,FMTDPB+4 ;SET UP WC FOR REMAINING SECTORS
1403 005562 062737 001010 001362      ADD    #520,FMTDPB+6 ;SET UP BA FOR REMAINING SECTORS
1404 005570 005237 001364      INC    FMTDPB+10    ;ADVANCE TO NEXT SECTOR IN TBL
1405 005574 005037 001216      CLR    SOFSW        ;RESET RETRY COUNTER
1406 005600 000207      RTS     PC          ;RETURN TO COMPLETE TRK FORMAT

;THIS ROUTINE SETS UP THE CYLINDER ADRS, FORMAT BIT, TRACK AND
;SECTOR ADRS IN MEMORY WITH THE STARTING CYL - TRK INFORMATION

1410
1411
1412 005602 013701 001266      SETHDR: MOV   MAXSEC,R1   ;SET UP SECTOR COUNT
1413 005606 012700 025130      MOV   #BUFP,R0         ;SET UP HEADER POINTER IN R0
1414 005612 013702 001224      MOV   BEGCYL,R2        ;PUT STARTING CYL # IN R2
1415 005616 013703 001230      MOV   BEGTRK,R3        ;PUT STARTING TRK # IN R3
1416 005622 000303                SWAB   R3              ;JUSTIFY TRACK ADRS
1417 005624 005737 001254      TST   SEC20            ;SEE IF 20 OR 22 SECTOR MODE
1418 005630 001402                BEQ   1$              ;BR IF 20 SECTOR MODE
1419 005632 052702 010000      1$:   BIS   #10000,R2    ;SET THE 22 SECTOR FORMAT BIT
1420 005636 010220                MOV   R2,(R0)+        ;WRITE IN HEADER AREA OF CORE THE CYL ADRS
1421 005640 010320                MOV   R3,(R0)+        ;WRITE IN HEADER AREA OF CORE THE TRK ADRS
1422 005642 005020                CLR   (R0)+           ;CLR 1ST KEYWORD
1423 005644 005010                CLR   (R0)            ;CLR 2ND KEYWORD
1424 005646 062700 001002      ADD   #514.,R0        ;SET UP FOR NEXT HEADER
1425 005652 005203                INC   R3              ;UPDATE SECTOR ADRS FOR NEXT HEADER
1426 005654 005301                DEC   R1              ;MAINTAIN COUNT OF SECTORS
1427 005656 002367                BGE   1$              ;BRANCH IF NOT LAST SECTOR
1428 005660 000207      RTS     PC          ;EXIT - HEADERS ARE LOADED INTO CORE

;THIS ROUTINE UPDATES THE CYLINDER ADRS OF THE HEADER WORDS IN CORE

1430
1431
1432 005662 013701 001266      UPDACY: MOV   MAXSEC,R1   ;SET UP SECTOR COUNT
1433 005666 012700 025130      MOV   #BUFP,R0         ;SET UP HEADER POINTER IN R0
1434 005672 005220      1$:   INC   (R0)+         ;INCREMENT FOR NEXT CYLINDER
1435 005674 042710                BIC   #177400,(R0)    ;RESET TRK ADRS TO 0
1436 005700 062700 001006      ADD   #518.,R0        ;SET UP FOR NEXT HEADER
1437 005704 005301                DEC   R1              ;COUNT SECTORS
1438 005706 002371                BGE   1$              ;BRANCH IF NOT LAST SECTOR
1439 005710 000207      RTS     PC          ;EXIT

;THIS ROUTINE UPDATES THE TRACK ADRS OF THE HEADER WORDS IN CORE

```

```

1443 005712 013701 001266 UPDATK: MOV MAXSEC,R1 ;SET UP SECTOR COUNT
1444 005716 012700 025130 MOV #BUFF,R0 ;SET UP HEADER POINTER IN R0
1445 005722 005720 TST (R0)+ ;POINT HEADER POINTER TO TRK - SEC ADRS
1446 005724 062710 000400 IS: ADD #400,(R0) ;INDEX TRK ADRS
1447 005730 062700 001010 ADD #520.,R0 ;SET UP FOR NEXT HEADER
1448 005734 005301 DEC R1 ;COUNT SECTORS
1449 005736 002372 SGE IS ;BRANCH IF NOT LAST SECTOR
1450 005740 000207 RTS PC ;EXIT

```

;ROUTINE TO CHECK FOR KW11-L OR KW11-P CLOCKS

```

1454 005742 012737 006020 000004 ST.CLK: MOV #STCLK1, @#ERRVEC ;SET UP VECTOR FOR P CLK
1455 005750 005037 000006 CLR @#ERRVEC+2 ;NEW PSW
1456 005754 005777 173216 TST @SLKCSR ;CHECK FOR KW11-P
1457 005760 013746 001202 MOV $LPVEC, -(SP) ;VECTOR ADDRESS
1458 005764 012776 006104 000000 MOV #CLOCK, @ (SP) ;SET UP KW11-P VECTOR
1459 005772 062716 000002 ADD #2, (SP) ;POINT TO PSW
1460 005776 012736 000300 MOV #PR6, @ (SP)+ ;PSW - PRI 6
1461 006002 012777 177777 173170 MOV #-1, @SLKCSB ;LOAD COUNTER BUFFER
1462 006010 012777 000135 173160 MOV #135, @SLKCSR ;SET CLK - CNT UP
1463 006016 000426 BR STCLK3
1464 006020 062706 000004 STCLK1: ADD #4, SP ;RESTORE THE STACK POINTER
1465 006024 012737 006070 000004 MOV #STCLK2, @#ERRVEC ;CHANGE ERROR VECTOR
1466 006032 005777 173150 TST @SLKS ;LOOK FOR KW11-L
1467 006036 013746 001210 MOV $LLVEC, -(SP) ;KW11-L VECTOR ADDRESS
1468 006042 012776 006104 000000 MOV #CLOCK, @ (SP) ;SET UP KW11-L VECTOR
1469 006050 062716 000002 ADD #2, (SP) ;INCREMENT VECTOR ADDRESS
1470 006054 012736 000300 MOV #PR6, @ (SP)+ ;PSW - PRI 6
1471 006060 012777 000100 173120 MOV #100, @SLKS ;SET KW11-L INTERRUPT ENABLE
1472 006066 000402 BR STCLK3
1473 006070 062706 000004 STCLK2: ADD #4, SP ;RESTORE THE STACK POINTER
1474 006074 012737 000006 000004 STCLK3: MOV #6, @#ERRVEC ;RESTORE THE ERROR VECTOR
1475 006102 000207 RTS PC

```

;THIS CODE SERVICES A CLOCK INTERRUPT EVERY 16MS

```

1476
1477
1478
1479 006104 012746 000020 CLOCK: MOV #16, -(SP) ;PUT MILLISECONDS ON THE STACK
1480 006110 004737 016466 JSR PC,RPTMR ;GO REPORT TIME
1481 006114 000002 RTI ;RETURN AND CONTINUE

```

;ROUTINE TO INTERCEPT 'CONTROL C' TYPED DURING PARAMETER ENTRY TIME

```

1482
1483
1484
1485 006116 012706 001100 CENTER: MOV #STACK, SP ;INITIALIZE THE STACK
1486 006122 005737 012160 IS: TST TRNSWT ;ALL ACTIVITY STOPPED ?
1487 006126 001375 BNE IS ;BR IF NOT
1488 006130 000005 RESET ;CLEAR THE BUS
1489 006132 000137 003062 JMP MIB ;START AGAIN WITH DRIVE SELECTION

```

;ROUTINE TO TYPE THE PRESENT DISK ADDRESS. ENTERED BY TYPING 'CONTROL C'

```

1490
1491
1492
1493 006136 005037 177776 TYPADR: CLR PSW ;SET PROCESSOR TO PRIORITY 0
1494 006142 012737 006116 001300 MOV #CENTER, CNTLC ;CHANGE 'CONTROL C' RETURN ADDRESS
1495 006150 104401 022065 TYPE ,ADDRIS ;PRESENT ADDRESS IS:
1496 006154 104401 020675 TYPE ,C ;C

```



```

1497 006160 013746 001366      MOV      FMTDPB+12,-(SP)    ;PUT THE CYLINDER ADDRESS ON THE STACK
1498 006164 004737 011470      JSR      PC,$SB2D          ;CONVERT IT TO DECIMAL
1499 006170 004737 011430      JSR      PC,$SUPRS         ;TYPE IT
1500 006174 104401 020705      TYPE    .LINSF            ;SPACES
1501 006200 104401 020677      TYPE    .T                ;T
1502 006204 005046                CLR      -(SP)            ;CLEAR THE STACK
1503 006206 113716 001365      MOVB    FMTDPB+11,(SP)    ;PUT THE TRACK ADDRESS ON THE STACK
1504 006212 004737 011470      JSR      PC,$SB2D          ;CONVERT IT TO DECIMAL
1505 006216 004737 011430      JSR      PC,$SUPRS         ;TYPE IT
1506 006222 104401 001167      TYPE    $CRLF             ;CR-LF
1507 006226 012737 006136      MOV      #TYPADR,CNTLC    ;RESTORE ENTRANCE TO THIS ROUTINE
1508 006234 000002                RTI                       ;RETURN

```

001300

:PARAMETER ENTRY ROUTINE

```

1509 000000                :CALL
1510 000000                :
1511 000000                :
1512 000000                :
1513 000000                :
1514 000000                :
1515 006236 010346                PARENT: MOV      R3,-(SP)    ;SAVE R3
1516 006240 012703 001374      MOV      #TABLE,R3       ;PARAMETER TABLE ADDRESS
1517 006244 012337 006254      JSR      PC,PARENT       ;GET THE PARAMETERS
1518 006250 001436                1$:  MOV      (R3)+,3$     ;ADDRESS OF PARAMETER NAME
1519 006252 104401                BEQ      9$              ;BR IF AT END OF TABLE
1520 006254 000000                3$:  TYPE    .WORD         ;TYPE THE PARAMETER NAME
1521 006256 012302                MOV      (R3)+,R2        ;ADDRESS OF PARAMETER NAME TEXT
1522 006260 012305                MOV      (R3)+,R5        ;MAXIMUM PARAMETER VALUE
1523 006262 011546                MOV      (R5),-(SP)     ;ADDRESS OF PARAMETER
1524 006264 104405                TYPDS    .SLASH         ;CURRENT VALUE OF PARAMETER
1525 006266 104401 020671      TYPE    .SLASH          ;TYPE THE CURRENT VALUE OF THE PARAMETER
1526 006272 104411                RDLIN   ;READ THE KEYBOARD
1527 006274 012601                MOV      (SP)+,R1       ;INPUT ASCII STRING ADDRESS
1528 006276 004537 006500      JSR      R5,CK.DIG       ;CHECK THE DIGIT(S)
1529 006302 006244                1$     ;CARRIAGE RETURN ONLY ENTERED
1530 006304 006346                9$     ;PERIOD ONLY ENTERED
1531 006306 006322                6$     ;ILLEGAL INPUT
1532 006310 006316                5$     ;TERMINATED WITH A CARRIAGE RETURN
1533 006312 006322                5$     ;TERMINATED WITH A "."
1534 006314 006334                7$     ;TERMINATED WITH A ":"
1535 006316 010215                5$:  MOV      R2,(R5)     ;MOVE NEW VALUE TO PARAMETER LOCATION
1536 006320 000751                BR      1$              ;GET MORE PARAMETERS
1537 006322 104401 021436      TYPE    'BADENT         ;'BAD ENTRY'
1538 006326 162703 000006      SUB     #6,R3           ;DECREMENT THE TABLE POINTER
1539 006332 000744                BR      1$              ;TRY AGAIN
1540 006334 010215                7$:  MOV      R2,(R5)     ;NEW VALUE
1541 006336 000403                BR      9$              ;EXIT
1542 006340 012703 001374      8$:  MOV      #TABLE,R3   ;RELOAD THE PARAMETER TABLE ADDRESS
1543 006344 000737                BR      1$              ;TRY AGAIN
1544 006346 012603                9$:  MOV      (SP)+,R3    ;RESTORE R3
1545 006350 000207                RTS      PC              ;RETURN

```

```

:THIS ROUTINE IS USED TO CHECK IF AN
:ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.
:CALL

```

```

1551      :      MOV      #ADR,R1      ;ADDRESS OF ASCII CHARACTER
1552      :      JSR      R5,CK.OCT    ;CHECK THE CHARACTER
1553      :      RETURN1   ;CHARACTER IS NOT BETWEEN 0-7
1554      :      RETURN2   ;CHARACTER IS IN R2 AS A
1555      :      :          ;OCTAL DIGIT
1556
1557 006352 121127 000060 CK.OCT: CMPB   (R1),#'0    ;LESS THAN ZERO?
1558 006356 103407      BLO    1$      ;YES -- BRANCH
1559 006360 121127 000067      CMPB   (R1),#'7    ;GREATER THAN SEVEN?
1560 006364 101004      BHI    1$      ;YES -- BRANCH
1561 006366 111102      MOVB   (R1),R2    ;GET THE CHARACTER
1562 006370 042702 177770      BIC    #'C,R2    ;STRIP AWAY THE ASCII
1563 006374 005725      TST    (R5)+     ;ADJUST FOR RETURN
1564 006376 000205      1$:   RTS     R5      ;RETURN
1565
1566      :      ;THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER
1567      :      ;AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.
1568      :      ;CALL
1569      :      :          MOV      #ADR,R1      ;ADDRESS OF ASCII CHARACTER
1570      :      :          JSR      R5,CK.DEC    ;CHECK THE CHARACTER
1571      :      :          RETURN1   ;NOT BETWEEN 0 AND 9
1572      :      :          RETURN2   ;BETWEEN 0 AND 9
1573      :      :          ;R2 = DIGIT
1574
1575 006400 121127 000060 CK.DEC: CMPB   (R1),#'0    ;LESS THAN ZERO?
1576 006404 103407      BLO    1$      ;YES -- BRANCH
1577 006406 121127 000071      CMPB   (R1),#'9    ;GREATER THAN NINE?
1578 006412 101004      BHI    1$      ;YES -- BRANCH
1579 006414 111102      MOVB   (R1),R2    ;GET THE CHARACTER
1580 006416 042702 000060      BIC    #'0,R2    ;STRIP AWAY THE ASCII
1581 006422 005725      TST    (R5)+     ;ADJUST FOR RETURN
1582 006424 000205      1$:   RTS     R5      ;RETURN
1583
1584      :      ;THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO
1585      :      ;DETERMINE WHAT IT IS.
1586      :      ;CALL
1587      :      :          MOV      #ADR,R1      ;ADDRESS OF ASCII CHARACTER
1588      :      :          JSR      R5,CK.CHR    ;CHECK CHARACTER
1589      :      :          RETURN  ADR1     ;UNKNOWN CHARACTER
1590      :      :          RETURN  ADR2     ;CARRIAGE RETURN * (R1)=ADR+1
1591      :      :          RETURN  ADR3     ;COMMA * (R1)=ADR+1
1592      :      :          RETURN  ADR4     ;PERIOD * (R1)=ADR+1
1593      :      :          RETURN  ADR5     ;DIGIT BETWEEN 0 AND 7.
1594      :      :          RETURN  ADR6     ;DIGIT BETWEEN 8 AND 9.
1595      :      :          ;R2 = DIGIT * (R1)=ADR+1
1596
1597 006426 105711 CK.CHR: TSTB   (R1)      ;"CARRIAGE RETURN"?
1598 006430 001417      BEQ    3$      ;YES -- BRANCH
1599 006432 121127 000054      CMPB   (R1),#',    ;"COMMA"?
1600 006436 001413      BEQ    2$      ;YES -- BRANCH
1601 006440 121127 000056      CMPB   (R1),#'.    ;"PERIOD"?
1602 006444 001407      BEQ    1$      ;YES -- BRANCH
1603 006446 004537 006400      JSR    R5,CK.DEC  ;"DIGIT"?
1604 006452 000410      BR     4$      ;NO -- BRANCH

```



```

1605 006454 004537 006352 JSR R5,CK.OCT ;OCTAL ?
1606 006460 005725 TST (R5)+ ;DIGIT BETWEEN 8-9
1607 006462 005725 TST (R5)+ ;DIGIT BETWEEN 0-7
1608 006464 005725 1$: TST (R5)+ ;PERIOD
1609 006466 005725 2$: TST (R5)+ ;COMMA
1610 006470 005725 3$: TST (R5)+ ;CARRIAGE RETURN
1611 006472 005201 INC R1 ;MOVE POINTER TO NEXT CHARACTER
1612 006474 011505 4$: MOV (R5),R5 ;UNKNOWN CHARACTER
1613 006476 000205 RTS R5 ;RETURN

:THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL
:CHARACTERS AND FORMS A DECIMAL VALUE BINARY NUMBER IN R2.
CALL
:MOV #ADR,R1 ;ADDRESS OF ASCIZ STRING
:MOV #NUM,R2 ;MAX. MAGNITUDE OF INPUT NUMBER
JSR R5,CK.DIG ;CHECK DIGITS
RETURN ADR1 ;"CR" ONLY ENTERED -- R2=0
RETURN ADR2 ;"PERIOD" ONLY ENTERED -- R2=0
RETURN ADR3 ;ILLEGAL CHARACTER OR INPUT TOO LARGE -- R2=?
RETURN ADR4 ;"CR" -- R2 = NUMBER
RETURN ADR5 ;"COMMA" -- R2 = NUMBER
RETURN ADR6 ;"PERIOD" -- R2 = NUMBER

1628 006500 010446 CK.DIG: MOV R4,-(SP) ;SAVE R4
1629 006502 010346 MOV R3,-(SP) ;SAVE R3
1630 006504 010246 MOV R2,-(SP) ;SAVE THE MAX. SIZE ON THE STACK
1631 006506 005002 CLR R2 ;START WITH 0
1632 006510 005003 CLR R3
1633 006512 005004 CLR R4
1634 006514 004537 006426 JSR R5,CK.CHR ;CHECK ONE CHARACTER
1635 006520 006614 6$: ;ILLEGAL CHARACTER
1636 006522 006622 9$: ;CARRIAGE RETURN
1637 006524 006614 6$: ;
1638 006526 006616 7$: ;
1639 006530 006534 1$: ;DIGIT 0-7
1640 006532 006534 1$: ;DIGIT 8-9
1641 006534 062705 000004 1$: ADD #4,R5 ;STEP RETURN POINTER PAST "CR" & "PERIOD" RETURNS
1642 006540 006303 2$: ASL R3 ;INPUT NUMBER *2
1643 006542 010346 MOV R3,-(SP) ;SAVE *2
1644 006544 006303 ASL R3 ;*4
1645 006546 006303 ASL R3 ;*8
1646 006550 062603 ADD (SP)+,R3 ;(*2)+(*8) = *10
1647 006552 060203 ADD R2,R3 ;UPDATE THE INPUT NUMBER
1648 006554 004537 006426 JSR R5,CK.CHR ;CHECK ONE CHARACTER
1649 006560 006620 8$: ;ILLEGAL CHARACTER
1650 006562 006604 5$: ;CARRIAGE RETURN
1651 006564 006602 4$: ;
1652 006566 006574 3$: ;
1653 006570 006540 2$: ;DIGIT 0-7
1654 006572 006540 2$: ;DIGIT 8-9
1655 006574 105711 3$: TSTB (R1) ;DOES A "CR" FOLLOW THE "PERIOD"
1656 006576 001010 BNE 9$ ;BR IF NOT
1657 006600 005724 TST (R4)+ ;INCREMENT THE RETURN
1658 006602 005724 4$: TST (R4)+ ;INCREMENT THE RETURN

```

```

1659 006604 005724 5$: TST (R4)+ ;INCREMENT THE RETURN
1660 006606 020316 CMP R3,(SP) ;CHECK THE MAGNITUDE OF THE NUMBER
1661 006610 101004 BHI 9$ ;BR IF ENTERED NUMBER TOO LARGE
1662 006612 000402 BR 8$ ;BYPASS INCREMENT
1663 006614 005725 6$: TST (R5)+ ;INCREMENT RETURN PAST INVALID RETURN
1664 006616 005725 7$: TST (R5)+ ;INCREMENT RETURN
1665 006620 060405 8$: ADD R4,R5 ;SETUP RETURN POINTER
1666 006622 010302 9$: MOV R3,R2 ;ENTERED VALUE
1667 006624 005726 TST (SP)+ ;CLEAN MAX. SIZE OFF OF STACK
1668 006626 012603 MOV (SP)+,R3 ;RESTORE R3
1669 006630 012604 MOV (SP)+,R4 ;RESTORE R4
1670 006632 011505 MOV (R5),R5 ;GET RETURN ADDRESS
1671 006634 000205 RTS R5 ;RETURN
1672
1673 .SBTTL MACRO ROUTINES
1674
1675
1676 .SBTTL ERROR HANDLER ROUTINE
1677
1678 ::*****
1679 ::THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
1680 ::SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
1681 ::AND GO TO TYPERR ON ERROR
1682 ::THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
1683 ::*SW15=1 HALT ON ERROR
1684 ::*SW13=1 INHIBIT ERROR TYPEOUTS
1685 ::*SW10=1 BELL ON ERROR
1686 ::*SW09=1 LOOP ON ERROR
1687 ::*CALL
1688 ::* ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER
1689
1690 $ERROR:
1691 006636 104407 CKSWR ;:TEST FOR CHANGE IN SOFT-SWR
1692 006636 010137 001274 MOV R1,DDRIVE ;:DRIVE ADDRESS IF DRIVE ERROR CALL
1693 006640 010337 001276 MOV R3,ATTN ;:ATTENTION REGISTER CONTENTS
1694 006650 013737 001366 001270 MOV FMTDPB+12,DS,CYL ;:CURRENT CYLINDER ADDRESS
1695 006656 113737 001365 001272 MOV FMTDPB+11,DS,TRK ;:REQUESTED TRACK ADDRESS
1696 006664 005737 001310 MOVB RP.REG+RPBA ;:NON-ZERO BUFFER ADDRESS ?
1697 006670 001406 BEQ 7$ ;:BR IF NO BUFFER ADDRESS
1698 006672 013746 001310 MOV RP.REG+RPBA,-(SP) ;:BUFFER ADDRESS
1699 006676 162716 000002 SUB #2,(SP) ;:DECREMENT THE ADDRESS
1700 006702 013637 001124 MOV @((SP)+,%GDDAT) ;:GET THE BUFFER WORD WHICH DIDN'T COMPARE
1701 006706 105237 001103 7$: INCB $ERFLG ;:SET THE ERROR FLAG
1702 006712 001775 BEQ 7$ ;:DON'T LET THE FLAG GO TO ZERO
1703 006714 013777 001102 172220 MOV $STSTM,%DISPLAY ;:DISPLAY TEST NUMBER AND ERROR FLAG
1704 006722 032777 002000 172210 BIT #BIT10,%SWR ;:BELL ON ERROR?
1705 006730 001402 BEQ 1$ ;:NO - SKIP
1706 006732 104401 001162 TYPE $BELL ;:RING BELL
1707 006736 005237 001112 1$: INC $ERTTL ;:COUNT THE NUMBER OF ERRORS
1708 006742 011637 001116 MOV (SP),%ERRPC ;:GET ADDRESS OF ERROR INSTRUCTION
1709 006746 162737 000002 001116 SUB #2,%ERRPC
1710 006754 117737 172136 001114 MOVB @%ERRPC,%ITEMB ;:STRIP AND SAVE THE ERROR ITEM CODE
1711 006762 032777 020000 172150 BIT #BIT13,%SWR ;:SKIP TYPEOUT IF SET
1712 006770 001004 BNE 20$ ;:SKIP TYPEOUTS

```



```

1713 006772 004737 007044 JSR PC,TYPERR ;;GO TO USER ERROR ROUTINE
1714 006776 104401 001167 TYPE ,$CRLF
1715 007002 20$: TST $SWR ;;HALT ON ERROR
1716 007002 005777 172132 2$: BPL 3$ ;;SKIP IF CONTINUE
1717 007006 100002 HALT ;;HALT ON ERROR!
1719 007010 000000 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
1719 007012 104407 001000 172116 3$: BIT #BIT09,$SWR ;;LOOP ON ERROR SWITCH SET?
1720 007014 032777 001110 4$: BEQ 4$ BR IF NO
1721 007022 001402 MOV $LPERR,(SP) ;;FUDGE RETURN FOR LOOPING
1722 007024 013716 001160 4$: TST $ESCAPE ;;CHECK FOR AN ESCAPE ADDRESS
1723 007030 005737 001160 5$: BEQ 5$ BR IF NONE
1724 007034 001402 MOV $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
1725 007036 013716 001160 5$: RTI ;;RETURN
1726 007042 000002
1727 007042 000002
1728
1729
1730
1731
1732
1733
1734

```

```

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

```

```

1735 007044 104412 TYPERR: SAVREG ;;SAVE R0-R5
1736 007046 005000 CLR R0 ;;CLEAR R0 FOR ERROR NUMBER
1737 007050 113700 001114 MOV $ITEMB,R0 ;;ERROR NUMBER
1738 007054 005300 DEC R0 ;;FORM INDEX FOR ERROR TABLE
1739 007056 006300 ASL R0
1740 007060 006300 ASL R0
1741 007062 006300 ASL R0
1742 007064 062700 001626 1$: ADD $ERRTB,R0 ;;FORM ADDRESS
1743 007070 012037 007104 MOV (R0)+,2$ ;;GET ERROR MESSAGE (EM) POINTER
1744 007074 001404 BEQ 3$ ;;BRANCH IF THERE ISN'T ONE
1745 007076 104401 001167 TYPE , $CRLF ;;CARRIAGE RETURN - LINE FEED
1746 007102 104401 2$: .WORD 0 ;;"EM" POINTER GOES HERE
1747 007104 000000 3$: MOV (R0)+,4$ ;;PICK UP DATA HEADER (DH) POINTER
1748 007106 012037 007122 BEQ 5$ ;;BRANCH IF NONE
1749 007112 001404 001167 TYPE , $CRLF ;;CARRIAGE RETURN-LINE FEED
1750 007114 104401 4$: .WORD 0 ;;"DH" POINTER GOES HERE
1751 007120 104401 5$: MOV (R0)+,R1 ;;PICKUP DATA TABLE (DT) POINTER
1752 007122 000000 BEQ 20$ ;;BRANCH IF NONE
1753 007124 012001 CLR R5 ;;SET INDENT SWITCH
1754 007126 001460 MOV (R0)+,R0 ;;DATA FORMAT (DF) POINTER
1755 007130 005005 MOV (R0)+,R2 ;;NUMBER OF DH'S TO TYPE
1756 007132 012000 BEQ 17$ ;;BRANCH IF DH NUMBER IS 0
1757 007134 012002 COM R5 ;;NO INDENT
1758 007136 001451 TYPE , $CRLF ;;CARRIAGE RETURN-LINE FEED
1759 007140 005105 001167 MOV (R0)+,R3 ;;NUMBER OF DATA WORDS TO TYPE
1760 007142 104401 10$: MOV (R0)+,R4 ;;AND HOW TO TYPE THEM
1761 007146 112003 ROR R4 ;;OCTAL OR DECIMAL?
1762 007150 112004 BCS 12$ ;;DECIMAL--BRANCH
1763 007152 006004 MOV $J(R1)+,-(SP) ;;SAVE J(R1)+ FOR TYPEOUT
1764 007154 103403 TYPCC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1765 007156 013146
1766 007160 104402

```



```

1821 007314 112046      2$:  MOVB   (RO)+,-(SP)    ;; PUSH CHARACTER TO BE TYPED ONTO STACK
1822 007316 001005      BNE    4$              ;; BR IF IT ISN'T THE TERMINATOR
1823 007320 005726      TST    (SP)+          ;; IF TERMINATOR POP IT OFF THE STACK
1824 007322 012600      60$:  MOV    (SP)+,RO     ;; RESTORE RO
1825 007324 062716 000002  3$:  ADD    #2,(SP)       ;; ADJUST RETURN PC
1826 007330 000002      RTI                    ;; RETURN
1827 007332 122716 000011  4$:  CMPB   #HT,(SP)      ;; BRANCH IF <HT>
1828 007336 001430      BEQ    8$              ;; BRANCH IF NOT <CRLF>
1829 007340 122716 000200      CMPB   #CRLF,(SP)    ;; BRANCH IF NOT <CRLF>
1830 007344 001006      BNE    5$              ;; POP <CR><LF> EQUIV
1831 007346 005726      TST    (SP)+          ;; TYPE A CR AND LF
1832 007350 104401      TYPE                    ;; TYPE A CR AND LF
1833 007352 001167      $CRLF                    ;; TYPE A CR AND LF
1834 007354 105037 007510      CLRB   $CHARCNT       ;; CLEAR CHARACTER COUNT
1835 007360 000755      BR     2$              ;; GET NEXT CHARACTER
1836 007362 004737 007444      5$:  JSR    PC,$TYPEPC     ;; GO TYPE THIS CHARACTER
1837 007366 123726 001156      6$:  CMPB   $FILLC,(SP)+  ;; IS IT TIME FOR FILLER CHARS.?
1838 007372 001350      BNE    2$              ;; IF NO GO GET NEXT CHAR.
1839 007374 013746 001154      MOV    $NULL,-(SP)    ;; GET # OF FILLER CHARS. NEEDED
1840                                     AND THE NULL CHAR.
1841 007400 105366 000001  7$:  DECB   1(SP)          ;; DOES A NULL NEED TO BE TYPED?
1842 007404 002770      BLT    6$              ;; BR IF NO--GO POP THE NULL OFF OF STACK
1843 007406 004737 007444      JSR    PC,$TYPEPC     ;; GO TYPE A NULL
1844 007412 105337 007510      DECB   $CHARCNT       ;; DO NOT COUNT AS A COUNT
1845 007416 000770      BR     7$              ;; LOOP

```

;HORIZONTAL TAB PROCESSOR

```

1846
1847
1848
1849 007420 112716 000040      8$:  MOVB   #' ,(SP)      ;; REPLACE TAB WITH SPACE
1850 007424 004737 007444      9$:  JSR    PC,$TYPEPC     ;; TYPE A SPACE
1851 007430 132737 000007 007510  BITB   #7,$CHARCNT    ;; BRANCH IF NOT AT
1852 007436 001372      BNE    9$              ;; TAB STOP
1853 007440 005726      TST    (SP)+          ;; POP SPACE OFF STACK
1854 007442 000724      BR     2$              ;; GET NEXT CHARACTER
1855 007444 105777 171500  $TYPEPC: TSTB   $STPS        ;; WAIT UNTIL PRINTER IS READY
1856 007450 100375      BPL    $TYPEPC
1857 007452 116677 000002 171472  MOVB   2(SP), $STPB    ;; LOAD CHAR TO BE TYPED INTO DATA REG.
1858 007460 122766 000015 000002  CMPB   #CR,2(SP)      ;; IS CHARACTER A CARRIAGE RETURN?
1859 007466 001003      BNE    1$              ;; BRANCH IF NO
1860 007470 105037 007510      CLRB   $CHARCNT       ;; YES--CLEAR CHARACTER COUNT
1861 007474 000406      BR     $TYPEPC        ;; EXIT
1862 007476 122766 000012 000002  1$:  CMPB   #LF,2(SP)     ;; IS CHARACTER A LINE FEED?
1863 007504 001402      BEQ    $TYPEPC        ;; BRANCH IF YES
1864 007506 105227      INCB   (PC)+          ;; COUNT THE CHARACTER
1865 007510 000000  $CHARCNT: .WORD 0     ;; CHARACTER COUNT STORAGE
1866 007512 000207  $TYPEPC: RTS    PC
1867
1868
1869
1870

```

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

1871 ;;*****
1872 ;;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
1873 ;;*OCTAL (ASCII) NUMBER AND TYPE IT.
1874 ;;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE

```

```

1875      *CALL:
1876      *      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
1877      *      TYPOS      ;;CALL FOR TYPEOUT
1878      *      .BYTE    N      ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
1879      *      .BYTE    M      ;;M=1 OR 0
1880      *      ;;1=TYPE LEADING ZEROS
1881      *      ;;0=SUPPRESS LEADING ZEROS
1882      *
1883      *$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
1884      *$TYPOS OR $TYPOC
1885      *CALL:
1886      *      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
1887      *      TYPON      ;;CALL FOR TYPEOUT
1888      *
1889      *$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
1890      *CALL:
1891      *      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
1892      *      TYPOC      ;;CALL FOR TYPEOUT
1893
1894 007514 017646 000000      $TYPOS: MOV      2(SP),-(SP)      ;;PICKUP THE MODE
1895 007520 116637 000001 007737 MOVVB   1(SP),$OFILL      ;;LOAD ZERO FILL SWITCH
1896 007526 112637 007741 MOVVB   (SP)+,$SOMODE+1      ;;NUMBER OF DIGITS TO TYPE
1897 007532 062716 000002 ADD     #2,(SP)      ;;ADJUST RETURN ADDRESS
1898 007536 000406 BR      $TYPON
1899 007540 112737 000001 007737 $TYPOC: MOVVB  #1,$OFILL      ;;SET THE ZERO FILL SWITCH
1900 007546 112737 000006 007741 MOVVB   #6,$SOMODE+1      ;;SET FOR SIX(6) DIGITS
1901 007554 112737 000005 007736 $TYPON: MOVVB  #5,$OCNT      ;;SET THE ITERATION COUNT
1902 007562 010346 MOV     R3,-(SP)      ;;SAVE R3
1903 007564 010446 MOV     R4,-(SP)      ;;SAVE R4
1904 007566 010546 MOV     R5,-(SP)      ;;SAVE R5
1905 007570 113704 007741 MOVVB   $SOMODE+1,R4      ;;GET THE NUMBER OF DIGITS TO TYPE
1906 007574 005404 NEG     R4
1907 007576 062704 000006 ADD     #6,R4      ;;SUBTRACT IT FOR MAX. ALLOWED
1908 007602 110437 007740 MOVVB   R4,$SOMODE      ;;SAVE IT FOR USE
1909 007606 113704 007737 MOVVB   $OFILL,R4      ;;GET THE ZERO FILL SWITCH
1910 007612 016605 000012 MOV     12(SP),R5      ;;PICKUP THE INPUT NUMBER
1911 007616 005003 CLR     R3      ;;CLEAR THE OUTPUT WORD
1912 007620 006105 1$: ROL    R5      ;;ROTATE MSB INTO "C"
1913 007622 000404 BR      3$      ;;GO DO MSB
1914 007624 006105 2$: ROL    R5      ;;FORM THIS DIGIT
1915 007626 006105 ROL    R5
1916 007630 006105 ROL    R5
1917 007632 010503 MOV     R5,R3
1918 007634 006103 3$: ROL    R3      ;;GET LSB OF THIS DIGIT
1919 007636 105337 007740 DECB   $SOMODE      ;;TYPE THIS DIGIT?
1920 007642 100016 BPL    7$      ;;BR IF NO
1921 007644 042703 177770 BIC    #177770,R3      ;;GET RID OF JUNK
1922 007650 001002 BNE    4$      ;;TEST FOR 0
1923 007652 005704 TST    R4      ;;SUPPRESS THIS 0?
1924 007654 001403 BEQ    5$      ;;BR IF YES
1925 007656 005204 4$: INC    R4      ;;DON'T SUPPRESS ANYMORE 0'S
1926 007660 052703 000060 BIS    #'0,R3      ;;MAKE THIS DIGIT ASCII
1927 007664 052703 000040 5$: BIS    #' ,R3      ;;MAKE ASCII IF NOT ALREADY
1928 007670 110337 007734 MOVVB  R3,9$      ;;SAVE FOR TYP'NG

```


M04

```

1929 007674 104401 007734          TYPE      8$          ;;GO TYPE THIS DIGIT
1930 007700 105337 007736      7$:  DECB      $OCNT          ;;COUNT BY 1
1931 007704 003347          BGT      2$          ;;BR IF MORE TO DO
1932 007706 002402          BLT      6$          ;;BR IF DONE
1933 007710 005204          INC      R4          ;;INSURE LAST DIGIT ISN'T A BLANK
1934 007712 000744          BR       2$          ;;GO DO THE LAST DIGIT
1935 007714 012605          6$:  MOV      (SP)+,R5          ;;RESTORE R5
1936 007716 012604          MOV      (SP)+,R4          ;;RESTORE R4
1937 007720 012603          MOV      (SP)+,R3          ;;RESTORE R3
1938 007722 016666 000002 000004  MOV      2(SP),4(SP)          ;;SET THE STACK FOR RETURNING
1939 007730 012616          MOV      (SP)+,(SP)
1940 007732 000002          RTI
1941 007734          000          8$:  .BYTE     0          ;;RETURN
1942 007735          000          .BYTE     0          ;;STORAGE FOR ASCII DIGIT
1943 007736          000          $OCNT: .BYTE     0          ;;TERMINATOR FOR TYPE ROUTINE
1944 007737          000          $OFILL: .BYTE     0          ;;OCTAL DIGIT COUNTER
1945 007740 000000          $OMODE: .WORD     0          ;;ZERO FILL SWITCH
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959 007742          010046          $TYPDS: MOV      R0,-(SP)          ;;PUT THE BINARY NUMBER ON THE STACK
1960 007742 010046          MOV      R1,-(SP)          ;;GO TO THE ROUTINE
1961 007744 010146          MOV      R2,-(SP)
1962 007746 010246          MOV      R3,-(SP)
1963 007750 010346          MOV      R5,-(SP)
1964 007752 010546          MOV      #20200,-(SP)          ;;PUSH R0 ON STACK
1965 007754 012746 020200          MOV      20(SP),R5          ;;PUSH R1 ON STACK
1966 007760 016605 000020          BPL      1$          ;;PUSH R2 ON STACK
1967 007764 100004          NEG      R5          ;;PUSH R3 ON STACK
1968 007766 005405          MOV#B   #'-,1(SP)          ;;PUSH R5 ON STACK
1969 007770 112766 000055 000001  1$:  CLR      R0          ;;SET BLANK SWITCH AND SIGN
1970 007776 005000          MOV      #SDBLK,R3          ;;GET THE INPUT NUMBER
1971 010000 012703 010156          MOV#B   #' ,(R3)+          ;;BR IF INPUT IS POS.
1972 010004 112723 000040          2$:  CLR      R2          ;;MAKE THE BINARY NUMBER POS.
1973 010010 005002          MOV      $DTBL(R0),R1          ;;MAKE THE ASCII NUMBER NEG.
1974 010012 016001 010146          3$:  SUB      R1,R5          ;;ZERO THE CONSTANTS INDEX
1975 010016 160105          BLT      4$          ;;SETUP THE OUTPUT POINTER
1976 010020 002402          INC      R2          ;;SET THE FIRST CHARACTER TO A BLANK
1977 010022 005202          BR       3$          ;;CLEAR THE BCD NUMBER
1978 010024 000774          4$:  ADD      R1,R5          ;;GET THE CONSTANT
1979 010026 060105          TST     R2          ;;FORM THIS BCD DIGIT
1980 010030 005702          BNE     5$          ;;BR IF DONE
1981 010032 001002          TSTB   (SP)          ;;INCREASE THE BCD DIGIT BY 1
1982 010034 105716

```

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```

*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
*REPLACED WITH SPACES.

```

```

*CALL:
*      MOV      NUM,-(SP)          ;;PUT THE BINARY NUMBER ON THE STACK
*      TYPDS          ;;GO TO THE ROUTINE

```

```

$TYPDS:
MOV      R0,-(SP)          ;;PUSH R0 ON STACK
MOV      R1,-(SP)          ;;PUSH R1 ON STACK
MOV      R2,-(SP)          ;;PUSH R2 ON STACK
MOV      R3,-(SP)          ;;PUSH R3 ON STACK
MOV      R5,-(SP)          ;;PUSH R5 ON STACK
MOV      #20200,-(SP)          ;;SET BLANK SWITCH AND SIGN
MOV      20(SP),R5          ;;GET THE INPUT NUMBER
BPL      1$          ;;BR IF INPUT IS POS.
NEG      R5          ;;MAKE THE BINARY NUMBER POS.
MOV#B   #'-,1(SP)          ;;MAKE THE ASCII NUMBER NEG.
1$:  CLR      R0          ;;ZERO THE CONSTANTS INDEX
MOV      #SDBLK,R3          ;;SETUP THE OUTPUT POINTER
MOV#B   #' ,(R3)+          ;;SET THE FIRST CHARACTER TO A BLANK
2$:  CLR      R2          ;;CLEAR THE BCD NUMBER
MOV      $DTBL(R0),R1          ;;GET THE CONSTANT
3$:  SUB      R1,R5          ;;FORM THIS BCD DIGIT
BLT      4$          ;;BR IF DONE
INC      R2          ;;INCREASE THE BCD DIGIT BY 1
BR       3$
4$:  ADD      R1,R5          ;;ADD BACK THE CONSTANT
TST     R2          ;;CHECK IF BCD DIGIT=0
BNE     5$          ;;FALL THROUGH IF 0
TSTB   (SP)          ;;STILL DOING LEADING 0'S?

```

```

1983 010036 100407          BMI      7$          ;; BR IF YES
1984 010040 106316          5$: ASLB    (SP)      ;; MSD?
1985 010042 103003          BCC     6$          ;; BR IF NO
1986 010044 116663 000001 177777  MOVB   1(SP),-1(R3) ;; YES--SET THE SIGN
1987 010052 052702 000060          6$: BIS    #'0,R2    ;; MAKE THE BCD DIGIT ASCII
1988 010056 052702 000040          7$: BIS    #' ,R2    ;; MAKE IT A SPACE IF NOT ALREADY A DIGIT
1989 010062 110223          MOVB   R2,(R3)+    ;; PUT THIS CHARACTER IN THE OUTPUT BUFFER
1990 010064 005720          TST    (R0)+      ;; JUST INCREMENTING
1991 010066 020027 000010          CMP    R0,#10    ;; CHECK THE TABLE INDEX
1992 010072 002746          BLT    2$          ;; GO DO THE NEXT DIGIT
1993 010074 003002          BGT    8$          ;; GO TO EXIT
1994 010076 010502          MOV    R5,R2     ;; GET THE LSD
1995 010100 000764          BR     6$         ;; GO CHANGE TO ASCII
1996 010102 105726          8$: TSTB  (SP)+    ;; WAS THE LSD THE FIRST NON-ZERO?
1997 010104 100003          BPL    9$         ;; BR IF NO
1998 010106 116663 177777 177776  MOVB   -1(SP),-2(R3) ;; YES--SET THE SIGN FOR TYPING
1999 010114 105013          9$: CLRB  (R3)     ;; SET THE TERMINATOR
2000 010116 012605          MOV    (SP)+,R5  ;; POP STACK INTO R5
2001 010120 012603          MOV    (SP)+,R3  ;; POP STACK INTO R3
2002 010122 012602          MOV    (SP)+,R2  ;; POP STACK INTO R2
2003 010124 012601          MOV    (SP)+,R1  ;; POP STACK INTO R1
2004 010126 012600          MOV    (SP)+,R0  ;; POP STACK INTO R0
2005 010130 104401 010156          TYPE   $DBLK     ;; NOW TYPE THE NUMBER
2006 010134 016666 000002 000004  MOV    2(SP),4(SP) ;; ADJUST THE STACK
2007 010142 012616          MOV    (SP)+,(SP)
2008 010144 000002          RTI                    ;; RETURN TO USER
2009 010146 023420          $DTBL: 10000.
2010 010150 001750          1000.
2011 010152 000144          100.
2012 010154 000012          10.
2013 010156 000004          $DBLK: .BLKW 4
2014
2015          .SBTTL TTY INPUT ROUTINE
2016
2017          ;;*****
2018          .ENABL LSB
2019 010166 000000          $TKCNT: .WORD 0    ;; NUMBER OF ITEMS IN QUEUE
2020 010170 000000          $TKQIN: .WORD 0    ;; INPUT POINTER
2021 010172 000000          $TKQOUT: .WORD 0   ;; OUTPUT POINTER
2022 010174 000007          $TKQSRT: .BLKB 7   ;; TTY KEYBOARD QUEUE
2023          $TKQEND=.
2024          .EVEN
2025
2026          ;*TK INITIALIZE ROUTINE
2027          ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
2028          ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
2029
2030          ;*CALL:
2031          ;* JSR PC,$TKINT
2032          ;* RETURN
2033
2034 010204 005037 010166          $TKINT: CLR    $TKCNT ;; CLEAR COUNT OF ITEMS IN QUEUE
2035 010210 012737 010174 010170  MOV    # $TKQSRT,$TKQIN ;; MOVE THE STARTING ADDRESS OF THE
2036 010216 013737 010170 010172  MOV    $TKQIN,$TKQOUT ;; QUEUE INTO THE INPUT & OUTPUT POINTERS.

```



```

010224 012737 010254 000060      MOV    #STKSRV, @TKVEC  ;; INITIALIZE THE KEYBOARD VECTOR
010226 012737 000200 000062      MOV    #200, @TKVEC+2  ;; "BR" LEVEL 4
010240 005777 170702          TST    @STKB           ;; CLEAR DONE FLAG
010244 012777 000100 170672      MOV    #100, @STKS     ;; ENABLE TTY KEYBOARD INTERRUPT
010252 000207          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" (1C) STKINT IS CALLED AND
; *UPON RETURN EXIT IS MADE TO THE "CONTROL-C" RESTART ADDRESS (@CNTLC)

```

```

010254 117746 170666      $TKSRV: MOV    @STKB, -(SP)    ;; PICKUP THE CHARACTER
010260 042716 177600      BIC    #177, (SP)    ;; STRIP THE JUNK
010264 021627 000003      CMP    (SP), #3      ;; IS IT A CONTROL C?
010270 001007          BNE    1$           ;; BRANCH IF NO
010272 104401 011367      TYPE   @CNTLC        ;; TYPE A CONTROL-C (1C)
010276 004737 010204      JSR    PC, STKINT    ;; INIT THE KEYBOARD
010302 005726          TST    (SP)+         ;; CLEAN UP STACK
010304 000177 170770      JMP    @CNTLC        ;; CONTROL C RESTART
010310 021627 000007      1$:  CMP    (SP), #7      ;; IS IT A CONTROL G?
010314 001004          BNE    2$           ;; BRANCH IF NO
010316 022737 000176 001140      CMP    #SWREG, SWR   ;; IS SOFT-SWR SELECTED?
010324 001500          BEQ    6$           ;; GO TO SWR CHANGE

010326          2$:
010326 022737 000007 010166      CMP    #7, $TKCNT    ;; IS THE QUEUE FULL?
010334 001004          BNE    3$           ;; BRANCH IF NO
010336 104401 001162      TYPE   $BELL         ;; RING THE TTY BELL
010342 005726          TST    (SP)+         ;; CLEAN CHARACTER OFF OF STACK
010344 000451          BR    5$           ;; EXIT
010346 021627 000023      3$:  CMP    (SP), #23     ;; IS IT A CONTROL-S?
010352 001021          BNE    32$          ;; BRANCH IF NO
010354 005077 170564      CLR    @STKS         ;; DISABLE TTY KEYBOARD INTERRUPTS
010360 005726          TST    (SP)+         ;; CLEAN CHAR OFF STACK
010362 105777 170556      31$: TSTB   @STKS         ;; WAIT FOR A CHAR
010366 100375          BPL    31$          ;; LOOP UNTIL ITS THERE
010370 117746 170552      MOV    @STKB, -(SP)  ;; GET THE CHARACTER
010374 042716 177600      BIC    #177, (SP)    ;; MAKE IT 7-BIT ASCII
010400 022627 000021      CMP    (SP)+, #21    ;; IS IT A CONTROL-Q?
010404 001366          BNE    31$          ;; BRANCH IF NO
010406 012777 000100 170530      MOV    #100, @STKS   ;; REENABLE TTY KEYBOARD INTERRUPTS
010414 000002          RTI                ;; RETURN
010416 005237 010166      32$: INC    $TKCNT        ;; COUNT THIS CHARACTER
010422 021627 000140      CMP    (SP), #140    ;; IS IT UPPER CASE?
010426 002405          BLT    4$           ;; BRANCH IF YES
010430 021627 000175      CMP    (SP), #175    ;; IS IT A SPECIAL CHAR?
010434 003002          BGT    4$           ;; BRANCH IF YES
010436 042716 000040      BIC    #40, (SP)     ;; MAKE IT UPPER CASE
010442 112677 177522      4$:  MOV    (SP)+, @STKQIN ;; AND PUT IT IN QUEUE
010446 005237 010170      INC    $TKQIN        ;; UPDATE THE POINTER
010452 023727 010170 010203      CMP    $TKQIN, #STKGEND ;; GO OFF THE END?
010460 001003          BNE    5$           ;; BRANCH IF NO

```

```

0091 010462 012737 010174 010170      MOV    #STKQSR,STKQIN  ;;RESET THE POINTER
0092 010470 000002                5$:   RTI                ;;RETURN
0093
0094
0095
0096
0097
0098
0099
0100
0101
0102
0103
0104
0105
0106
0107
0108
0109
0110
0111
0112
0113
0114
0115
0116
0117
0118
0119
0120
0121
0122
0123
0124
0125
0126
0127
0128
0129
0130
0131
0132
0133
0134
0135
0136
0137
0138
0139
0140
0141
0142
0143
0144
0145
0146
0147
0148
0149
0150
0151
0152
0153
0154
0155
0156
0157
0158
0159
0160
0161
0162
0163
0164
0165
0166
0167
0168
0169
0170
0171
0172
0173
0174
0175
0176
0177
0178
0179
0180
0181
0182
0183
0184
0185
0186
0187
0188
0189
0190
0191
0192
0193
0194
0195
0196
0197
0198
0199
0200
0201
0202
0203
0204
0205
0206
0207
0208
0209
0210
0211
0212
0213
0214
0215
0216
0217
0218
0219
0220
0221
0222
0223
0224
0225
0226
0227
0228
0229
0230
0231
0232
0233
0234
0235
0236
0237
0238
0239
0240
0241
0242
0243
0244
0245
0246
0247
0248
0249
0250
0251
0252
0253
0254
0255
0256
0257
0258
0259
0260
0261
0262
0263
0264
0265
0266
0267
0268
0269
0270
0271
0272
0273
0274
0275
0276
0277
0278
0279
0280
0281
0282
0283
0284
0285
0286
0287
0288
0289
0290
0291
0292
0293
0294
0295
0296
0297
0298
0299
0300

```

```

*****
*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   $STKB,-(SP)   ;;YES
        BIC    #177,(SP)     ;;MAKE IT 7-BIT ASCII
        CMP    (SP),#7      ;;IS IT A CONTROL-G?
        BNE    2$           ;;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.
5$:   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

```

```

010526 123727 001134 000001 5$:   CMPB   $AUTOB,#1      ;;ARE WE RUNNING IN AUTO-MODE?
010534 001674                BEQ   2$           ;;BRANCH IF YES
010536 005726                TST  (SP)+         ;;CLEAR CONTROL-G OFF STACK
010540 004737 010204        JSR  PC,$TKINT     ;;FLUSH THE TTY INPUT QUEUE
010544 005077 170374        CLR  $STKS        ;;DISABLE TTY KEYBOARD INTERRUPTS
010550 112737 000001 001135        MOVB #1,$INTAG     ;;SET INTERRUPT MODE INDICATOR

```

```

010556 104401 011401        $GTSWR: TYPE    , $CNTLG      ;;ECHO THE CONTROL-G (1G)
010562 104401 011406        TYPE    $MSWR        ;;TYPE CURRENT CONTENTS
010566 013746 000176        MOV     $SWREG,-(SP) ;;SAVE SWREG FOR TYPEOUT
010572 104402                TYPCC                ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
010574 104401 011417        TYPE    , $MNEW      ;;PROMPT FOR NEW SWR
010600 005046        19$:   CLR    -(SP)      ;;CLEAR COUNTER
010602 005046        CLR    -(SP)      ;;THE NEW SWR
010604 105777 170334        7$:   TSTB   $STKS        ;;CHAR THERE?
010610 100375                BPL    7$           ;;IF NOT TRY AGAIN

```

```

010612 117746 170330        MOVB   $STKB,-(SP)   ;;PICK UP CHAR
010616 042716 177600        BIC    #177,(SP)    ;;MAKE IT 7-BIT ASCII
010622 021627 000003        CMP    (SP),#3      ;;IS IT A CONTROL-C?
010626 001015                BNE    9$           ;;BRANCH IF NOT
010630 104401 011367        TYPE    , $CNTLC     ;;YES, ECHO CONTROL-C (1C)
010634 062706 000006        ADD    #6,SP        ;;CLEAN UP STACK
010640 123727 001135 000001        CMPB   $INTAG,#1   ;;REENABLE TTY KEYBOARD INTERRUPTS?
010646 001003                BNE    8$           ;;BRANCH IF NO
010650 012777 000100 170266        MOV    #100,$STKS  ;;ALLOW TTY KEYBOARD INTERRUPTS
010656 000177 170416        8$:   JMP    $CNTLC     ;;CONTROL-C RESTART

```

```

010662 021627 000025        9$:   CMP    (SP),#25   ;;IS IT A CONTROL-U?
010666 001005                BNE    10$          ;;BRANCH IF NOT

```



```

010670 104401 011374 TYPE $CNTLU ::YES, ECHO CONTROL-U (IU)
010674 062706 000006 20$: ADD #6,SP ::IGNORE PREVIOUS INPUT
010700 000737 BR 19$ ::LET'S TRY IT AGAIN

010702 021627 000015 10$: CMP (SP),#15 ::IS IT A <CR>?
010706 001022 BNE 16$ ::BRANCH IF NO
010710 005766 000004 TST 4(SP) ::YES, IS IT THE FIRST CHAR?
010714 001403 BEQ 11$ ::BRANCH IF YES
010716 016677 000002 170214 MOV 2(SP),@SWR ::SAVE NEW SWR
010724 062706 000006 11$: ADD #6,SP ::CLEAR UP STACK
010730 104401 001167 14$: TYPE $CRLF ::ECHO <CR> AND <LF>
010734 123727 001135 000001 CMPB $INTAG,#1 ::RE-ENABLE TTY KBD INTERRUPTS?
010742 001003 BNE 15$ ::BRANCH IF NOT
010744 012777 000100 170172 MOV #100,@STKS ::RE-ENABLE TTY KBD INTERRUPTS
010752 000002 15$: RTI ::RETURN
010754 004737 007444 16$: JSR PC,$TYPEC ::ECHO CHAR
010760 021627 000060 CMP (SP),#60 ::CHAR < 0?
010764 002420 BLT 18$ ::BRANCH IF YES
010766 021627 000067 CMP (SP),#67 ::CHAR > 7?
010772 003015 BGT 18$ ::BRANCH IF YES
010774 042726 000060 BIC #60,(SP)+ ::STRIP-OFF ASCII
011000 005766 000002 TST 2(SP) ::IS THIS THE FIRST CHAR
011004 001403 BEQ 17$ ::BRANCH IF YES
011006 006316 ASL (SP) ::NO, SHIFT PRESENT
011010 006316 ASL (SP) :: CHAR OVER TO MAKE
011012 006316 ASL (SP) :: ROOM FOR NEW ONE.
011014 005266 000002 17$: INC 2(SP) ::KEEP COUNT OF CHAR
011020 056616 177776 OYS -2(SP),(SP) ::SET IN NEW CHAR
011024 000667 BR 7$ ::GET THE NEXT ONE
011026 104401 001166 18$: TYPE $QUES ::TYPE ?<CR><LF>
011032 000720 BR 20$ ::SIMULATE CONTROL-U
.DSABL LSB

```

*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY

*CALL:

```

* RDCHR ::GET A CHARACTER FROM THE QUEUE
* RETURN HERE ::CHARACTER IS ON THE STACK
* ::WITH PARITY BIT STRIPPED OFF

```

```

011034 011646 000004 000002 $RDCHR: MOV (SP),-(SP) ::PUSH DOWN THE PC AND
011036 016666 000004 MOV 4(SP),2(SP) ::THE PS
011044 005066 000004 CLR 4(SP) ::GET READY FOR A CHARACTER
011050 005046 CLR -(SP) ::PUT NEW PS ON STACK
011052 012746 011060 MOV #64$,-(SP) ::PUT NEW PC ON STACK
011056 000002 RTI ::POP NEW PC AND PS
011060 011060 64$: TST $TKCNT ::WAIT ON A CHARACTER
011064 001775 1$: BEQ 1$
011066 005337 010166 DEC $TKCNT ::DECREMENT THE COUNTER
011072 117766 177074 000004 MOVEB @TKCNT,4(SP) ::GET ONE CHARACTER

```

```

011100 005237 010172 INC $TKQOUT ;: UPDATE THE POINTER
011104 023727 010172 010203 CMP $TKQOUT, $TKQEND ;: DID IT GO OFF OF THE END?
011112 001003 BNE 2$ ;: BRANCH IF NO
011114 012737 010174 010172 MOV $TKQSRT, $TKQOUT ;: RESET THE POINTER
011122 000002 RTI ;: RETURN
2$:
;: *****
;: *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
;: *CALL:
;: * RDLIN ;: INPUT A STRING FROM THE TTY
;: * RETURN HERE ;: ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
;: * ;: TERMINATOR WILL BE A BYTE OF ALL 0'S

011124 010346 $RDLIN: MOV R3, -(SP) ;: SAVE R3
011126 005046 CLR -(SP) ;: CLEAR THE RUBOUT KEY
011130 012703 011360 1$: MOV $TTYIN, R3 ;: GET ADDRESS
011134 022703 011367 2$: CMP $TTYIN+7, R3 ;: BUFFER FULL?
011140 101456 BLOS 4$ ;: BR IF YES
011142 104410 RDCHR ;: GO READ ONE CHARACTER FROM THE TTY
011144 112613 MOV $B (SP)+, (R3) ;: GET CHARACTER
011146 122713 000177 10$: CMPB #177, (R3) ;: IS IT A RUBOUT
011152 001022 BNE 5$ ;: BR IF NO
011154 005716 TST (SP) ;: IS THIS THE FIRST RUBOUT?
011156 001007 BNE 6$ ;: BR IF NO
011160 112737 000134 011356 MOV $B '\, 9$ ;: TYPE A BACK SLASH
011166 104401 011356 TYPE 9$
011172 012716 177777 MOV $B -1, (SP) ;: SET THE RUBOUT KEY
011176 005303 6$: DEC R3 ;: BACKUP BY ONE
011200 020327 011360 7$: CMP R3, $TTYIN ;: STACK EMPTY?
011204 103434 BLOS 4$ ;: BR IF YES
011206 111337 011356 MOV $B (R3), 9$ ;: SETUP TO TYPEOUT THE DELETED CHAR.
011212 104401 011356 TYPE 9$ ;: GO TYPE
011216 000746 2$: BR 2$ ;: GO READ ANOTHER CHAR.
011220 005716 5$: TST (SP) ;: RUBOUT KEY SET?
011222 001406 7$: BEG 7$ ;: BR IF NO
011224 112737 000134 011356 MOV $B '\, 9$ ;: TYPE A BACK SLASH
011232 104401 011356 TYPE 9$
011236 005016 CLR (SP) ;: CLEAR THE RUBOUT KEY
011240 122713 000025 7$: CMPB #25, (R3) ;: IS CHARACTER A CTRL U?
011244 001003 8$: BNE 8$ ;: BR IF NO
011246 104401 011374 TYPE $CNTLU ;: TYPE A CONTROL "U"
011252 000726 1$: BR 1$ ;: GO START OVER
011254 122713 000022 8$: CMPB #22, (R3) ;: IS CHARACTER A "r"?
011260 001011 3$: BNE 3$ ;: BRANCH IF NO
011262 105013 CLR $B (R3) ;: CLEAR THE CHARACTER
011264 104401 001167 TYPE $CRLF ;: TYPE A "CR" & "LF"
011270 104401 011360 TYPE $TTYIN ;: TYPE THE INPUT STRING
011274 000717 2$: BR 2$ ;: GO PICKUP ANOTHER CHARACTER
011276 104401 001166 4$: TYPE $QUES ;: TYPE A '?'
011302 000712 1$: BR 1$ ;: CLEAR THE BUFFER AND LOOP
011304 111337 011356 3$: MOV $B (R3), 9$ ;: ECHO THE CHARACTER
011310 104401 011356 TYPE 9$
011314 122723 000015 CMPB #15, (R3)+ ;: CHECK FOR RETURN
011320 001305 2$: BNE 2$ ;: LOOP IF NOT RETURN
011322 105063 177777 CLR $B -1(R3) ;: CLEAR RETURN (THE 15)

```



```

2353 011326 104401 001170 TYPE $LF ;;TYPE A LINE FEED
2354 011332 005726 TST (SP)+ ;;CLEAN RUBOUT KEY FROM THE STACK
2355 011334 012603 MOV (SP)+,R3 ;;RESTORE R3
2356 011336 011646 MOV (SP)-,(SP) ;;ADJUST THE STACK AND PUT ADDRESS OF THE
2357 011340 016666 000004 000002 MOV 4(SP),2(SP) ;; FIRST ASCII CHARACTER ON IT
2358 011346 012766 011360 000004 MOV $TTYIN,4(SP)
2359 011354 000002 RTI ;;RETURN
2360 011356 000 9$: .BYTE 0 ;;STORAGE FOR ASCII CHAR. TO TYPE
2361 011357 000 .BYTE 0 ;;TERMINATOR
2362 011360 000007 $TTYIN: .BLKB 7 ;;RESERVE 7 BYTES FOR TTY INPUT
2363 011367 136 006503 000012 $CNTLC: .ASCIZ /C<15><12> ;;CONTROL "C"
2364 011374 352536 005015 000 $CNTLU: .ASCIZ /U<15><12> ;;CONTROL "U"
2365 011401 136 006507 000012 $CNTLG: .ASCIZ /G<15><12> ;;CONTROL "G"
2366 011406 005015 053523 020122 $MSWR: .ASCIZ <15><12>/SWR = /
2367 011414 020075 000
2368 011417 040 047040 053505 $MNEW: .ASCIZ / NEW = /
2369 011424 036440 000040

.SBTL TYPE NUMERICAL ASCIZ STRING SUPPRESS LEADING ZEROS

*****
*THIS ROUTINE IS USED TO TYPE AN ASCIZ NUMBER SUPPRESSING THE
*LEADING NUMBERS.
*CALL
* MOV $NUMADR,-(SP) ;;FIRST ADDRESS OF ASCIZ STRING
* JSR PC,2#$SUPRS

$SUPRS: MOV R0,-(SP) ;;SAVE R0
MOV 4(SP),R0 ;;PICKUP THE POINTER
1$: TSTB (R0) ;;TERMINATOR?
BEQ 2$ ;;BR IF YES
CMPB #'0,(R0)+ ;;IS THIS AN ASCII "0" ?
BEQ 1$ ;;BR IF YES
2$: DEC R0 ;;BACKUP BY "1"
MOV R0,3$ ;;SAVE FOR TYPING
TYPE ;;GO TYPE
3$: .WORD 0 ;;ASCIZ POINTER GOES HERE
MOV (SP)+,R0 ;;RESTORE R0
MOV (SP)+,(SP) ;;RESTORE THE STACK
RTS PC ;;RETURN

.SBTL SINGLE LENGTH BINARY TO DECIMAL ASCIZ ROUTINE

*****
*THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
*UNSIGNED DECIMAL ASCIZ NUMBER.
*CALL
* MOV NUMBER,-(SP) ;;PUT BINARY NUMBER ON THE STACK
* JSR PC,2#$SB2D ;;CALL
* RETURN ;;ADDRESS OF THE 1ST ASCIZ CHAR.IS ON THE STACK

2306 011470 016637 000002 011520 $SB2D: MOV 2(SP),1$ ;;SAVE BINARY NUMBER

```

2307	011476	012746	011520
2308	011502	004737	011524
2309	011506	062716	000005
2310	011512	012666	000002
2311	011516	000207	
2312	011520	000000	000000

```

MOV #1$, -(SP)      ;; SET POINTER
JSR PC, 2#$DB2D     ;; CALL DOUBLE LENGTH CONVERT
ADD #5, (SP)        ;; ONLY ALLOW FIVE CHARACTERS
MOV (SP)+, 2(SP)    ;; PICKUP POINTER
RTS PC              ;; RETURN
1$: .WORD 0,0

```

.SBTTL DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

```

*****
*THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
*DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
*POSITIVE.
*CALL

```

```

* MOV #PNTR, -(SP) ;; POINTER TO LOW WORD OF BINARY NUMBER
* JSR PC, 2#$DB2D
* RETURN           ;; THE FIRST ADDRESS OF ASCII
;; IS ON THE STACK

```

2313	011524	104412	
2314	011526	016602	000002
2315	011532	012700	011704
2316	011536	010066	000002
2317	011542	012201	
2318	011544	012202	
2319	011546	012737	000012 011622
2320	011554	012704	011634
2321	011560	012705	011636
2322	011564	005003	
2323	011566	161401	
2324	011570	005602	
2325	011572	161502	
2326	011574	002402	
2327	011576	005203	
2328	011600	000772	
2329	011602	062401	
2330	011604	005502	
2331	011606	062402	
2332	011610	022525	
2333	011612	052703	000060
2334	011616	110320	
2335	011620	005327	
2336	011622	000000	
2337	011624	001357	
2338	011626	105020	
2339	011630	104413	
2340	011632	000207	
2341	011634	145000	
2342	011636	035632	
2343	011640	160400	
2344	011642	002765	
2345	011644	113200	
2346	011646	000230	

```

$DB2D: SAVREG      ;; SAVE REGISTERS
MOV 2(SP), R2     ;; PICKUP THE DATA POINTER
MOV #SDECVL, R0   ;; GET ADDRESS OF "SDECVL" STRING
MOV R0, 2(SP)     ;; PUT ADDRESS OF ASCII STRING ON STACK
MOV (R2)+, R1     ;; PICKUP THE BINARY NUMBER
MOV (R2)+, R2
MOV #10, 4$      ;; SET UP TO DO 10 CONVERSIONS
MOV #STNPWR, R4   ;; ADDRESS OF TEN POWER
MOV #STNPWR+2, R5
1$: CLR R3        ;; CLEAR PARTIAL
2$: SUB (R4), R1  ;; SUBTRACT TEN POWER
SBC R2
SUB (R5), R2
BLT 3$           ;; BR IF TEN POWER TOO LARGE
INC R3           ;; ADD 1 TO PARTIAL
BR 2$           ;; LOOP
3$: ADD (R4)+, R1 ;; RESTORE SUBTRACTED VALUE
ADC R2
ADD (R4)+, R2
CMP (R5)+, (R5)+ ;; MOVE TO NEXT TEN POWER
BIS #0, R3      ;; CHANGE PARTIAL TO ASCII
MOVB R3, (R0)+  ;; SAVE IT
DEC (PC)+      ;; DONE?
4$: .WORD 0
BNE 1$         ;; BR IF NO
CLRB (R0)+     ;; TERMINATOR
RESREG        ;; RESTORE REGISTERS
RTS PC        ;; RETURN
STNPWR: 145000  ;; 1.0E09
        35632
        160400  ;; 1.0E08
        2765
        113200  ;; 1.0E07
        230

```



```

011650 041100
011652 000017
011654 103240
011656 000001
011660 023420
011662 000000
011664 001750
011666 000000
011670 000144
011672 000000
011674 000012
011676 000000
011700 000001
011702 000000
011704 000014

```

```

041100 ;;1.0E06
17
103240 ;;1.0E05
1
23420 ;;1.0E04
0
1750 ;;1.0E03
0
144 ;;1.0E02
0
12 ;;1.0E01
0
1
0 ;;1.0E00
0
$DECVL: ,BLKB 12. ;;RESERVE STORAGE FOR ASCII STRING

```

.SBTTL SAVE AND RESTORE RO-R5 ROUTINES

```

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

```

```

011720
011722 010046
011724 010146
011726 010246
011730 010346
011732 010446
011734 010546
011734 016646 000022
011740 016646 000022
011744 016646 000022
011750 016646 000022
011754 000002
011756
011756 012666 000022
011762 012666 000022
011766 012666 000022
011772 012666 000022

```

```

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

*RESTORE RO-R5
*CALL:
* RESREG
$RESREG:
MOV (SP)+,22(SP) ;;RESTORE PC OF CALL
MOV (SP)+,22(SP) ;;RESTORE PS OF CALL
MOV (SP)+,22(SP) ;;RESTORE PC OF MAIN FLOW
MOV (SP)+,22(SP) ;;RESTORE PS OF MAIN FLOW

```

```

415 011776 012605      MOV      (SP)+,R5      ;;POP STACK INTO R5
416 012000 012604      MOV      (SP)+,R4      ;;POP STACK INTO R4
417 012002 012603      MOV      (SP)+,R3      ;;POP STACK INTO R3
418 012004 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
419 012006 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
420 012010 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
421 012012 000002      RTI

```

.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.

```

```

422 012014 010046      $TRAP: MOV      RO,-(SP)      ;;SAVE RO
423 012016 016600 000002  MOV      2(SP),RO      ;;GET TRAP ADDRESS
424 012022 005740      TST      -(RO)        ;;BACKUP BY 2
425 012024 111000      MOVSB   (RO),RO      ;;GET RIGHT BYTE OF TRAP
426 012026 006300      ASL     RO            ;;POSITION FOR INDEXING
427 012030 016000 012050  MOV      $TRPAD(RO),RO  ;;INDEX TO TABLE
428 012034 000200      RTS     RO            ;;GO TO ROUTINE

```

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

```

429 012036 011646      $TRAP2: MOV     (SP),-(SP)  ;;MOVE THE PC DOWN
430 012040 016666 000004 000002  MOV     4(SP),2(SP)    ;;MOVE THE PSW DOWN
431 012046 000002      RTI     ;;RESTORE THE PSW

```

.SBTTL TRAP TABLE

```

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

```

```

432 ; ROUTINE
433 ; -----
434 $TRPAD: .WORD  $TRAP2
435 $TYPE   ;;CALL=TYPE   TRAP+1(104401) TTY TYPEOUT ROUTINE
436 $TYPOC  ;;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
437 $TYPOS  ;;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
438 $TYPON  ;;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
439 $TYPDS  ;;CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
440
441 $GTSWR  ;;CALL=GTSWR  TRAP+6(104406) GET SOFT-SWR SETTING
442
443 $CKSWR  ;;CALL=CKSWR  TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
444 $RDCHR  ;;CALL=RDCHR  TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
445 $RDLIN  ;;CALL=RDLIN  TRAP+11(104411) TTY TYPEIN STRING ROUTINE
446 $SAVREG ;;CALL=SAVREG TRAP+12(104412) SAVE RO-R5 ROUTINE
447 $RESREG ;;CALL=RESREG  TRAP+13(104413) RESTORE RO-R5 ROUTINE
448
449 012050 012036
450 012052 007274
451 012054 007540
452 012056 007514
453 012060 007554
454 012062 007742
455
456 012064 010562
457
458 012066 010472
459 012070 011034
460 012072 011124
461 012074 011720
462 012076 011756

```


2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522

.SBTTL SINGLE/DUAL PORT RH11/RPO4/5/6 DRIVER (REV 1.0)

;COPYRIGHT (C) 1976
;DIGITAL EQUIPMENT CORP.
;MAYNARD, MA 01754
;AUTHOR(S): JIM LACEY/CHUCK HESS

;*****

;STORAGE FOR RPDS1, RPER1, RPER2, AND RPER3 ON AN ERROR "2"

;RPERRS = RPDS1
;RPERRS+2 = RPER1
;RPERRS+4 = RPER2
;RPERRS+6 = RPER3

012100 000000 000000 000000 RPERRS: .WORD 0,0,0,0
012106 000000

;TABLE OF DRIVE ACTIVE INDICATORS (DRVACT=8 BYTES)

;DRVACT=0 IF DRIVE IS IDLE
;DRVACT>0 IF DRIVE IS ACTIVE WITH A COMMAND
;DRVACT<0 IF DRIVE IS ACTIVE WITH AN ERROR RECOVERY OPERATION

012110 000 DRVACT: .BYTE 0 ;DRIVE 0
012111 000 .BYTE 0 ;DRIVE 1
012112 000 .BYTE 0 ;DRIVE 2
012113 000 .BYTE 0 ;DRIVE 3
012114 000 .BYTE 0 ;DRIVE 4
012115 000 .BYTE 0 ;DRIVE 5
012116 000 .BYTE 0 ;DRIVE 6
012117 000 .BYTE 0 ;DRIVE 7

;TABLE OF DRIVE STATUS INDICATORS (DRVSTA=8 BYTES)

;DRVSTA=0 IF DRIVE IS OFFLINE OR NONEXISTENT
;DRVSTA>0 IF DRIVE IS ONLINE
;DRVSTA<0 IF DRIVE IS UNSAFE

012120 000 DRVSTA: .BYTE 0 ;DRIVE 0
012121 000 .BYTE 0 ;DRIVE 1
012122 000 .BYTE 0 ;DRIVE 2
012123 000 .BYTE 0 ;DRIVE 3
012124 000 .BYTE 0 ;DRIVE 4
012125 000 .BYTE 0 ;DRIVE 5
012126 000 .BYTE 0 ;DRIVE 6
012127 000 .BYTE 0 ;DRIVE 7

;TABLE OF DRIVE TYPES (DRV TYP=8 BYTES)

;DRV TYP=0 IF DRIVE IS NONEXISTENT (DRVSTA=0, ALSO)
;DRV TYP=1 IF DRIVE IS RPO4
;DRV TYP=2 IF DRIVE IS RPO5
;DRV TYP=4 IF DRIVE IS RPO6
;DRV TYP=-1 IF NOT RPO4/5/6

```

2523 012130 000          DRVTYP: .BYTE 0          ;DRIVE 0
2524 012131 000          .BYTE 0          ;DRIVE 1
2525 012132 000          .BYTE 0          ;DRIVE 2
2526 012133 000          .BYTE 0          ;DRIVE 3
2527 012134 000          .BYTE 0          ;DRIVE 4
2528 012135 000          .BYTE 0          ;DRIVE 5
2529 012136 000          .BYTE 0          ;DRIVE 6
2530 012137 000          .BYTE 0          ;DRIVE 7
2531
2532          ;TABLE OF DUAL PORT INITIALIZATION INDICATORS
2533          ;DPINT=0 IF INITIALIZATION IS NOT ACTIVE ON THE DRIVE
2534          ;DPINT<0 IF INITIALIZATION IS IN PROGRESS
2535
2536 012140 000          DPINT: .BYTE 0          ;DRIVE 0
2537 012141 000          .BYTE 0          ;DRIVE 1
2538 012142 000          .BYTE 0          ;DRIVE 2
2539 012143 000          .BYTE 0          ;DRIVE 3
2540 012144 000          .BYTE 0          ;DRIVE 4
2541 012145 000          .BYTE 0          ;DRIVE 5
2542 012146 000          .BYTE 0          ;DRIVE 6
2543 012147 000          .BYTE 0          ;DRIVE 7
2544
2545          ;TABLE OF PENDING DUAL PORT REQUESTS
2546          ;DPRQS=0 IF THAT A DUAL PORT REQUEST IS NOT PENDING FOR THAT DRIVE
2547          ;DPRQS<0 IF THAT A DUAL PORT REQUEST IS PENDING FOR THAT DRIVE
2548
2549 012150 000          DPRQS: .BYTE 0          ;DRIVE 0
2550 012151 000          .BYTE 0          ;DRIVE 1
2551 012152 000          .BYTE 0          ;DRIVE 2
2552 012153 000          .BYTE 0          ;DRIVE 3
2553 012154 000          .BYTE 0          ;DRIVE 4
2554 012155 000          .BYTE 0          ;DRIVE 5
2555 012156 000          .BYTE 0          ;DRIVE 6
2556 012157 000          .BYTE 0          ;DRIVE 7
2557
2558          ;TRANSFER WAIT FLAG (TRNSWT=1 WORD)
2559          ;THIS IS A ONE WORD QUEUE. IT WILL CONTAIN THE ADDRESS OF
2560          ;"DPB" OF THE I/O OPERATION.
2561
2562 012160 000000          TRNSWT: .WORD 0
2563
2564          ;SEARCH WAIT KEYS (SRCHWT=1 WORD)
2565          ;THIS IS A ONE WORD QUEUE THAT WILL CONTAIN A KEY FOR EACH OF
2566          ;THE DRIVES THAT ARE PERFORMING A SEARCH COMMAND FOR THE I/O
2567          ;REQUEST THAT IS AT THE TOP OF THEIR REQUEST QUEUE.
2568          ;EACH DRIVE IS ASSIGNED ONE BIT, STARTING AT BIT00 FOR DRIVE 0.
2569
2570 012162 000000          SRCHWT: .WORD 0
2571
2572          ;RPO4/5/6 DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
2573          ;ACTDRV=0 IF DRIVER IS INACTIVE
2574          ;ACTDRV>0 IF DRIVER IS ACTIVE
2575
2576 012164 000          ACTDRV: .BYTE 0

```



```

2577
2578 ;SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)
2579 ;ACTSTR=0 IF SOFTWARE TIMER ROUTINE IS INACTIVE
2580 ;ACTSTR>0 IF SOFTWARE TIMER ROUTINE IS ACTIVE
2581
2582 012165 000 ACTSTR: .BYTE 0
2583
2584 ;UNLOAD FLAG (ULDFLG=8 BYTES)
2585 ;ULDFLG=0 IF NO UNLOAD COMMAND
2586 ;ULDFLG>0 IF UNLOAD COMMAND IN PROGRESS
2587 ;ULDFLG<0 IF UNLOAD COMMAND IN WAIT QUEUE
2588
2589 012166 000 ULDFLG: .BYTE 0 ;DRIVE 0
2590 012167 000 .BYTE 0 ;DRIVE 1
2591 012170 000 .BYTE 0 ;DRIVE 2
2592 012171 000 .BYTE 0 ;DRIVE 3
2593 012172 000 .BYTE 0 ;DRIVE 4
2594 012173 000 .BYTE 0 ;DRIVE 5
2595 012174 000 .BYTE 0 ;DRIVE 6
2596 012175 000 .BYTE 0 ;DRIVE 7
2597
2598 ;LOOK AHEAD COUNT (LACNT=8 BYTES)
2599 ;LACNT WILL INDICATE THE NUMBER OF LOOK AHEADS PERFORMED
2600
2601 012176 000 LACNT: .BYTE 0 ;DRIVE 0
2602 012177 000 .BYTE 0 ;DRIVE 1
2603 012200 000 .BYTE 0 ;DRIVE 2
2604 012201 000 .BYTE 0 ;DRIVE 3
2605 012202 000 .BYTE 0 ;DRIVE 4
2606 012203 000 .BYTE 0 ;DRIVE 5
2607 012204 000 .BYTE 0 ;DRIVE 6
2608 012205 000 .BYTE 0 ;DRIVE 7
2609
2610 ;SAVE REGISTERS FLAG (SAVEFG =1 WORD)
2611 ;SAVEFG <0 IF SAVE THE RH11/RPO4/5/6 REGISTERS WHEN THE
2612 ;OPERATION IS COMPLETED AS PER (DPB+14).
2613 ;SAVEFG=0 IF SAVE THE RH11/RPO4/5/6 REGISTERS, AS PER
2614 ;(DPB+14), AFTER AN ERROR.
2615
2616 012206 000000 SAVEFG: .WORD 0
2617
2618 ;SEEK FLAG (SEEKFG=1 WORD)
2619 ;SEEKFG=0 IF WHEN THE DISK ADDRESS ISN'T IN THE WINDOW
2620 ;FOR A DATA TRANSFER START A SEARCH COMMAND
2621 ;SEEKFG<0 IF DATA TRANSFER WILL DO IMPLIED SEEKS,
2622 ;DISREGARD THE WINDOW
2623
2624 012210 000000 SEEKFG: .WORD 0
2625
2626 ;TIMEOUT TABLE (TIMER=8 WORDS)
2627 ;THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION
2628
2629 012212 177777 TIMER: .WORD -1 ;DRIVE 0
2630 012214 177777 .WORD -1 ;DRIVE 1

```

```

2631 012216 177777 .WORD -1 ;DRIVE 2
2632 012220 177777 .WORD -1 ;DRIVE 3
2633 012222 177777 .WORD -1 ;DRIVE 4
2634 012224 177777 .WORD -1 ;DRIVE 5
2635 012226 177777 .WORD -1 ;DRIVE 6
2636 012230 177777 .WORD -1 ;DRIVE 7
2637
2638 ;DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)
2639 ;DTUW<0 IF NO DATA TRANSFER UNDERWAY
2640 ;DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE N
2641
2642 012232 177777 DTUW: .WORD -1
2643
2644 ;ATTENTION BITS TABLE (ATABIT=8 BYTES)
2645 ;THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES
2646 ;ATTENTION BIT
2647
2648 012234 001 ATABIT: .BYTE 1 ;DRIVE 0
2649 012235 002 .BYTE 2 ;DRIVE 1
2650 012236 004 .BYTE 4 ;DRIVE 2
2651 012237 010 .BYTE 10 ;DRIVE 3
2652 012240 020 .BYTE 20 ;DRIVE 4
2653 012241 040 .BYTE 40 ;DRIVE 5
2654 012242 100 .BYTE 100 ;DRIVE 6
2655 012243 200 .BYTE 200 ;DRIVE 7
2656
2657 ;RPO4/5/6 TO RH11 "MASSBUS CONTROL BUS PARITY ERRORS" (MCPE) ALLOWED BEFORE
2658 ;CALLING IT FATAL (MCPEM=1 WORD)
2659
2660 012244 000003 MCPEM: .WORD 3
2661
2662 ;STORAGE FOR RPADR (THE FIRST ADDRESS (776700) OF THE RH11/RPO4/5/6),
2663 ;RPVEC (THE VECTOR ADDRESS (254)), AND RPVEC+2 (THE BR LEVEL (5)).
2664
2665 012246 176700 RPADR: .WORD 176700
2666 012250 000254 000240 RPVEC: .WORD 254,5*32.
2667
2668 ;MAXIMUM NUMBER OF LOOK AHEADS ALLOWED IS 4 (MXLACT=1 WORD)
2669
2670 012254 000004 MXLACT: .WORD 4
2671 ;MAXIMUM DELTA DELAY IS 8 SECTORS (MXDLTA=1 WORD)
2672
2673 012256 001000 MXDLTA: .WORD 8.*64.
2674 ;MINIMUM DELTA DELAY IS 2 SECTORS (MNDLTA=1 WORD)
2675
2676 012260 000200 MNDLTA: .WORD 2*64.
2677 ;MAXIMUM SEARCH FOR I/O WINDOW IS 5 SECTORS (MXWNDW=1 WORD)
2678
2679 012262 000005 MXWNDW: .WORD 5
2680
2681 ;DEFINITIONS OF THE RH11/RPO4/5/6 ADDRESS INDEXES
2682
2683 000000 RPCS1=0 ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00)
2684 000002 RPIC=2 ;WORD COUNT REGISTER (NOT A DRIVE REG)

```



```

2685 000004 RPBA=4
2686 000006 RPDA=6
2687 000010 RPCS2=10
2688 000012 RPDS1=12
2689 000014 RPER1=14
2690 000016 RPAS=16
2691 000020 RPLA=20
2692 000022 RPDB=22
2693 000024 RPMR=24
2694 000026 RPDT=26
2695 000030 RPSN=30
2696 000032 RPOF=32
2697 000034 RPCA=34
2698 000036 RPCC=36
2699 000040 RPER2=40
2700 000042 RPER3=42
2701 000044 RPEC1=44
2702 000046 RPEC2=46

```

```

;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 05)
;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
;DRIVE STATUS REGISTER (DRIVE REG 01)
;ERROR REGISTER #1 (DRIVE REG. 02)
;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 04)
;LOOK AHEAD REGISTER (DRIVE REG. 07)
;DATA BUFFER REGISTER (NOT A DRIVE REG.)
;MAINTAINABILITY REGISTER (DRIVE REG. 03)
;DRIVE TYPE REGISTER (DRIVE REG. 06)
;SERIAL NUMBER REGISTER (DRIVE REG. 10)
;OFFSET REGISTER (DRIVE REG. 11)
;DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
;CURRENT CYLINDER ADDRESS REGISTER (DRIVE REG. 13)
;ERROR REGISTER #2 (DRIVE REG. 14)
;ERROR REGISTER #3 (DRIVE REG. 15)
;ECC POSITION REGISTER (DRIVE REG. 16)
;ECC PATTERN REGISTER (DRIVE REG. 17)

```

```

2703
2704 ;RH11/RP04/5/6 DRIVER INITIALIZATION CODE
2705 ;THIS ROUTINE WILL DETERMINE WHICH RP04/5/6 DRIVES ARE
2706 ;AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
2707 ;TO THE PROPER STATE FOR EACH DRIVE.
2708 ;NOTE: THIS ROUTINE CALLS DRVINT

```

```

2709 ;CALL
2710 ;
2711 ;
2712 ; JSR PC,RPINIT
2713 ; RETURN
2714 ;
2715 ;NOTE: THE 'P' OR 'L' CLOCK MUST BE STARTED
2716 ;

```

```

2717 012264 104412 RPINIT: SAVREG ;SAVE R0 - R5
2718 012266 013746 177776 MOV #PS, -(SP) ;SAVE THE PRESENT PROCESSOR STATUS
2719 012272 012737 000240 177776 MOV #<5*32.>, #PS ;CHANGE THE PRIORITY TO 5
2720 012300 004737 020250 JSR PC, CLRQUE ;CLEAR ALL REQUEST QUEUES
2721 012304 012701 012100 MOV #RPERRS, R1 ;FIRST ADDRESS TO BE CLEARED
2722 012310 012702 012210 MOV #SEEKFG, R2 ;LAST ADDRESS TO BE CLEARED
2723 012314 005021 1$: CLR (R1)+ ;CLEAR
2724 012316 020102 CMP R1, R2 ;ARE WE DONE?
2725 012320 101775 BLOS 1$ ;BRANCH IF NO
2726 012322 012702 012232 MOV #DTUW, R2 ;LAST ADDRESS
2727 012326 012721 177777 2$: MOV #-1, (R1)+ ;INITIALIZE
2728 012332 020102 CMP R1, R2 ;DONE?
2729 012334 101774 BLOS 2$ ;LOOP IF NO
2730 012336 005037 012120 CLR DRVSTA ;SET ALL DRIVES TO OFFLINE
2731 012342 005037 012122 CLR DRVSTA+2
2732 012346 005037 012124 CLR DRVSTA+4
2733 012352 005037 012126 CLR DRVSTA+6
2734 012356 013703 012250 MOV RPVEC, R3 ;SETUP THE RH11/RP04/5/6 VECTOR
2735 012362 012723 015130 MOV #ISR, (R3)+
2736 012366 013713 012252 MOV RPVEC+2, (R3)
2737 012372 013704 012246 MOV RPADR, R4 ;FIRST ADDRESS OF RH11/RP04
2738 012376 012764 000040 000010 MOV #BIT05, RPCS2(R4) ;MASSBUS INIT

```

```

012404 005001 CLR R1 ; START WITH DRIVE 0
012406 004037 012476 3$: JSR R0,DRVINT ; INIT THE DRIVE
012412 000401 BR 4$ ; 'DVA' NOT SET OR PARITY ERROR
012414 000402 BR 5$ ; NORMAL RETURN
012416 105061 012120 4$: CLRB DRVSTA(R1) ; SET DRIVE STATUS TO OFFLINE
012420 005201 5$: INC R1 ; GO TO NEXT DRIVE
012424 042701 177770 BIC #107,R1 ; MASK OUT UNUSED BITS
012428 001366 BNE 3$ ; BR IF MORE DRIVES TO GO
012432 012701 000007 MOV #7,R1 ; START WITH DRIVE 7
012436 005037 177776 CLR J#PS ; CLEAR THE PROCESSOR STATUS
012442 105761 012140 6$: TSTB DPINT(R1) ; WAITING FOR DRIVE TO SWITCH PORTS ?
012446 001405 BEQ 8$ ; BR NOT WAITING
012450 004737 017704 JSR PC,SET.IE ; SET INTERRUPT
012454 105761 012140 7$: TSTB DPINT(R1) ; DRIVE SWITCHED PORTS ?
012460 001375 BNE 7$ ; BR IF NOT
012462 005301 8$: DEC R1 ; GO TO THE NEXT DRIVE
012464 100366 BPL 6$ ; CHECK NEXT DRIVE
012466 012637 177776 MOV (SP)+,J#PS ; RESTORE THE PROCESSOR STATUS
012472 104413 RESREG ; RESTORE R0 - R5
012474 000207 RTS PC ; BYE-BYE

;DRIVE INITIALIZATION ROUTINE
;THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
;AN RPO4/5/6. IF IT IS, A "READ-IN PRESET" IS ISSUED AND FMT22
;IS SET TO A "1". THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
;INSURE THEY ARE ALL ON A "1". AND DEPENDING ON THEIR STATE,
;DRVSTA IS SET TO THE PROPER CONDITION.

;CALL
MOV #DRVNUM,R1 ;DRIVE NUMBER TO R1
MOV RPADR,R4 ;UNIBUS ADDRESS OF RH11/RPO4/5/6 (RPCS1)
JSR R0,DRVINT ;CALLED BY A JSR
RETURN1 ;ERROR OCCURRED (PARITY)
RETURN2 ;NORMAL RETURN

DRVINT: MOV R5,-(SP) ;SAVE R5
CLRB DRVSTA(R1) ;START DRIVE STATUS AS OFFLINE
CLRB DRVSTYP(R1) ;CLEAR THE DRIVE TYPE INDICATOR
CLRB ULDFLG(R1) ;CLEAR THE UNLOAD FLAG
MOV R1,RPCS2(R4) ;SELECT A DRIVE
MOVB #11,RPCS1(R4) ;DO A DRIVE CLEAR COMMAND (& SEIZE DRIVE)
BIT #BIT12,RPCS2(R4) ;NONEXISTENT DRIVE?
IS ;NO---BRANCH
BEQ 1$ ;GO SET "IE" WITHOUT A "TRE"
JSR PC,SET.IE ;LEAVE THIS ROUTINE
BR 6$ ;SET DRIVE STATUS TO OFFLINE
CLRB DRVSTA(R1) ;SET DRIVE STATUS TO OFFLINE
BIT #BIT11,RPCS1(R4) ;SEE IF DRIVE AVAILABLE
BEQ 7$ ;BR IF DRIVE NOT AVAILABLE
JSR R0,RD.RP ;READ THE DRIVE TYPE REG.
RPDT 8$ ;ERROR RETURN ADDRESS
MOV (SP)+,R5 ;PUT DRIVE TYPE IN R5
MOVB #1,DRVSTYP(R1) ;SET RPO4 INDICATOR
CMP #20020,R5 ;IS IT A SINGLE PORT RPO4?

```



```

00000000 012604 001431 BEQ 2$ ;BRANCH IF YES
00000001 012606 022705 024020 CMP #24020,R5 ;IS IT A DUAL PORT RPO4?
00000002 012612 001426 BEQ 2$ ;BR IF YES
00000003 012614 112761 000002 012130 MOVB #2,DRVSTYP(R1) ;SET RPOS INDICATOR
00000004 012622 022705 020021 CMP #20021,R5 ;SINGLE PORT RPOS ?
00000005 012626 001420 BEQ 2$ ;BR IF YES
00000006 012630 022705 024021 CMP #24021,R5 ;DUAL PORT RPOS ?
00000007 012634 001415 BEQ 2$ ;BR IF YES
00000008 012636 112761 000004 012130 MOVB #4,DRVSTYP(R1) ;SET RPO6 INDICATOR
00000009 012644 022705 020022 CMP #20022,R5 ;SINGLE PORT RPO6 ?
00000010 012650 001407 BEQ 2$ ;BR IF YES
00000011 012652 022705 024022 CMP #24022,R5 ;DUAL PORT RPO6 ?
00000012 012656 001404 BEQ 2$ ;BR IF YES
00000013 012660 112761 177777 012130 MOVB #-1,DRVSTYP(R1) ;SET INDICATOR TO 'OTHER'
00000014 012666 000446 BR 6$ ;EXIT
00000015 012670 012746 000121 2$: MOV #121,-(SP) ;DO A "READ-IN PRESET"
00000016 012674 004037 017374 JSR RO,WRT.RP
00000017 012700 000000 RPOSI
00000018 012702 013030 6$
00000019 012704 012746 010000 MOV #BIT12,-(SP) ;SET FMT22=1
00000020 012710 004037 017374 JSR RO,WRT.RP
00000021 012714 000032 RPOF
00000022 012716 013030 6$
00000023 012720 004037 017214 JSR RO,RO.RP ;READ RPOSI
00000024 012724 000012 RPOSI
00000025 012726 013030 6$
00000026 012730 012605 MOV (SP)+,R5 ;AND SAVE IT IN R5
00000027 012732 100015 BPL 4$ ;BRANCH IF ATA=0
00000028 012734 116164 012234 000016 MOVB ATABIT(R1),RPAS(R4) ;CLEAR ATTENTION BIT
00000029 012742 004037 017214 JSR RO,RO.RP ;FIND OUT WHY ATA=1
00000030 012746 000014 RPER1
00000031 012750 013030 6$
00000032 012752 006126 ROL (SP)+ ;IS IT UNSAFE?
00000033 012754 100004 BPL 4$ ;BR IF NOT
00000034 012756 112761 177777 012120 MOVB #-1,DRVSTA(R1) ;SET UNSAFE INDICATOR
00000035 012764 000407 BR 6$ ;EXIT
00000036 012766 005105 4$: COM R5 ;CHECK MOL, DPR, DRY, AND VV
00000037 012770 042705 167077 BIC #10<BIT12!BIT08!BIT07!BIT06>,R5
00000038 012774 001003 BNE 6$ ;BRANCH IF MOL, DPR, DRY, OR VV IS CLEAR
00000039 012776 112761 000001 012120 MOVB #1,DRVSTA(R1) ;SET DRIVE STATUS TO ONLINE
00000040 013004 005720 6$: TST (R0)+ ;STEP OVER THE ERROR RETURN
00000041 013006 000410 BR 8$ ;EXIT
00000042 013010 006301 7$: ASL R1 ;CHANGE INDEX TO ADDRESS WORDS
00000043 013012 012761 003720 012212 MOV #2000.,TIMER(R1) ;START 2 SEC TIMER
00000044 013020 006201 ASR R1 ;RESTORE R1
00000045 013022 112761 177777 012140 MOVB #-1,DPINT(R1) ;SET PORT INITIALIZE INDICATOR
00000046 013030 012605 8$: MOV (SP)+,R5 ;RESTORE R5
00000047 013032 000200 RTS RO ;EXIT

;REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
;CALL
; JSR RO,3#RPO4 ;CALL THE RPO4/5/6 DRIVER

```



```

001 013304 005720          TST      (R0)+      ;SETUP FOR NORMAL RETURN
002 013306 000401          BR       10$        ;FINISH UP, THEN EXIT
003 013310 104413          9$: RESREG        ;RESTORE R0 - R5
004 013312 005720          10$: TST      (R0)+ ;CORRECT THE RETURN ADDRESS
005 013314 105037 012164 CLRB     ACTDRV     ;CLEAR "ACTIVE DRIVER" FLAG
006 013320 012637 177776 MOV      (SP)+, 0#PS ;RETURN "PS" TO USER LEVEL
007 013324 000200          RTS      R0        ;RETURN TO CALLER

;OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
CALL
MOV      #DRVNUM,R1    ;DRIVE NUMBER TO R1
JSR      PC,OPT        ;SETUP A COMMAND

OPT: SAVREG          ;SAVE R0 - R5
MOV      0#PS, -(SP)   ;SAVE PROC. STATUS
BICB    ATABIT(R1),SRCHWT ;CLEAR "SEARCH WAIT" KEY
JSR      PC,GETREQ     ;GET "DPB" POINTER OF REQUEST
TST     R2             ;IS THERE A REQUEST IN QUEUE?
BEQ     7$            ;NO--BRANCH TO EXIT
BIT     #BIT12,RPDS1(R4) ;IS MOL STILL SET ?
BEQ     9$            ;BR IF NOT
BIT     #BIT6,RPDS1(R4) ;IS VV SET ?
BNE     10$          ;BR IF IT IS
9$: JSR      R0,DRVINT  ;SEE IF DRIVE STILL ONLINE ?
BR      6$           ;PARITY OR 'DVA' NOT SET
10$: TSTB   DRVSTA(R1) ;IS DRIVE ONLINE?
BGT     1$           ;YES--BRANCH
JSR     PC,POPQUE     ;NO--REMOVE REQUEST FROM QUEUE
MOV     #BIT15!BIT14,16(R2) ;SET OFFLINE STATUS/ERROR INDICATOR
TSTB   DRVSTA(R1)    ;IS DRIVE UNSAFE ?
BPL     8$           ;BR TO EXIT IF NOT
MOV     #BIT15!BIT12,16(R2) ;SET UNSAFE STATUS/ERROR INDICATOR
BR      9$           ;BRANCH TO EXIT
1$: MOV     #111, -(SP) ;LOAD COMMAND ONTO THE STACK
JSR     R0,WRT.RP     ;LOAD THE REGISTER
RPCS1   ;REGISTER INCREMENT
6$:      ;ERROR RETURN ADDRESS
BIT     #BIT11,(R4)   ;DRIVE AVAILABLE ?
BEQ     5$           ;BR IF NOT
CMPB   #150,2(R2)    ;IS THE REQUEST FOR I/O?
BLT     2$           ;YES--BRANCH
JSR     PC,C14        ;CALL THE COMMAND INITIATOR
BR      8$           ;BRANCH TO EXIT
2$: TST     DTUW       ;DATA TRANSFER UNDERWAY?
BGE     4$           ;YES--GO START A SEARCH
TST     SEEKFG        ;DO IMPLIED SEEKS?
BMI     3$           ;YES---BRANCH
JSR     R0,LA         ;NO--DO LOOK AHEAD
BR      8$           ;RETURN HERE ON A PARITY ERROR
BR      4$           ;GO START A SEARCH
3$: JSR     PC,C11     ;START A DATA TRANSFER
BR      8$           ;
4$: JSR     PC,C13     ;START A SEARCH

```

```

2955 013534 000420 BR B$ ;GO TO THE EXIT
2956 013536 112761 177777 012150 5$: MOVB #-1,DPROS(R1) ;SET PORT REQUEST INDICATOR
2957 013544 010103 MOV R1,R3 ;SET UP TO ADDRESS WORDS
2958 013546 006303 ASL R3 ;CONVERT TO WORD INDEX
2959 013550 012763 023420 012212 MOV #10000.,TIMER(R3) ;START 10 SEC TIMER
2960 013556 000402 BR 7$ ;EXIT
2961 013560 004737 014436 6$: JSR PC,C17 ;PROCESS THE PARITY ERROR
2962 013564 032714 000100 7$: BIT #BIT06,(R4) ;SEE IF 'IE' ALREADY SET
2963 013570 001002 BNE 8$ ;BR IF SET
2964 013572 004737 017704 JSR PC,SET.IE ;SET "IE" WITHOUT A "TRE"
2965 013576 012637 177776 8$: MOV (SP)+,J#PS ;RESTORE PROC. STATUS
2966 013602 104413 RESREG ;RESTORE R0 - R5
2967 013604 000207 RTS PC

:COMMAND INITIATOR
:CALL
: MOV #DRVNUM,R1 ;DRIVE NUMBER
: MOV #DPB,R2 ;ADDRESS OF DPB
: JSR PC,C1? ;C1?= C11,C13, OR C14
: WHERE:
: C11=DATA TRANSFER
: C12=SEARCH REQUESTED BY DATA XFER
: C14=NOT DATA TRANSFER

2980 013606 004737 020444 C11: JSR PC,POPQUE ;REMOVE REQUEST FROM "DRIVES WAIT" QUEUE
2981 013612 010237 012160 MOV R2,TRANSW ;PUT REQ. IN TRANSFER WAIT QUEUE
2982 013616 010203 MOV R2,R3 ;DPB ADDRESS TO R3
2983 013620 013704 012246 MOV RPADR,R4 ;RPCS1 ADDRESS
2984 013624 010164 000010 MOV R1,RPCS2(R4) ;SELECT DRIVE
2985 013630 062703 000004 ADD #4,R3 ;DESIRED WORD COUNT
2986 013634 062704 000002 ADD #2,R4 ;RPWC ADDRESS
2987 013640 012324 MOV (R3)+,(R4)+ ;LOAD WORD COUNT
2988 013642 012324 MOV (R3)+,(R4)+ ;LOAD BUFFER ADDRESS
2989 013644 012346 MOV (R3)+,-(SP) ;LOAD SECTOR AND TRACK
2990 013646 004037 017374 JSR RO,WRT.RP ;CALL THE LOAD(WRITE) ROUTINE
2991 013652 000006 RPDA ;INDEX OF REGISTER TO LOAD
2992 013654 014436 C17 ;ERROR RETURN ADDRESS
2993 013656 012346 MOV (R3)+,-(SP) ;LOAD CYLINDER ADDRESS
2994 013660 004037 017374 JSR RO,WRT.RP
2995 013664 000034 RPCA
2996 013666 014436 C17
2997 013670 016246 000002 MOV 2(R2),-(SP) ;LOAD "COMMAND+GO", "A17SA16", AND "PSEL"
2998 013674 004037 017374 JSR RO,WRT.RP
2999 013700 000000 RPCS1
3000 013702 014436 C17
3001 013704 010137 012232 MOV R1,DTUW ;SET "DATA TRANSFER UNDERWAY"
3002 013710 000137 014400 JMP C15
3003 013714 013704 012246 C13: MOV RPADR,R4 ;RPCS1 ADDRESS
3004 013720 010164 000010 MOV R1,RPCS2(R4) ;SELECT DRIVE
3005 013724 016246 000012 MOV 12(R2),-(SP) ;DESIRED CYLINDER ADDRESS
3006 013730 004037 017374 JSR RO,WRT.RP
3007 013734 000034 RPCA
3008 013736 014436 C17
    
```


3009	013740	116203	000010		MOV B	10(R2),R3	: PICKUP SECTOR ADDRESS
3010	013744	163703	012262		SUB	MXWINDW,R3	: BACKUP BY MAX. SEARCH FOR I/O WINDOW
3011	013750	002002			BGE	1\$	
3012	013752	062703	000026		ADD	#22,R3	
3013	013756	010346		1\$:	MOV	R3,-(SP)	: COMBINE THE ADJUSTED SECTOR WITH
3014	013760	116266	000011	000001	MOV B	11(R2),1(SP)	: THE DESIRED TRACK
3015	013766	004037	017374		JSR	RD,WRT.RP	: LOAD DESIRED TRACK & SECTOR
3016	013772	000006			RPDA		
3017	013774	014436			CI7		
3018	013776	012746	000131		MOV	#131,-(SP)	: START A SEARCH
3019	014002	004037	017374		JSR	RD,WRT.RP	
3020	014006	000000			RPCS1		
3021	014010	014436			CI7		
3022	014012	156137	012234	012162	BISB	ATABIT(R1),SRCHWT	: SET "SEARCH WAIT" KEY
3023	014020	000567			BR	CI5	
3024	014022	013704	012246	CI4:	MOV	RPADR,R4	: RPCS1 ADDRESS
3025	014026	010164	000010		MOV	R1,RPCS2(R4)	: SELECT DRIVE
3026	014032	116203	000002		MOV B	2(R2),R3	: PICKUP THE REQUESTED COMMAND
3027	014036	122703	000131		CMPB	#131,R3	: IS IT A SEARCH COMMAND?
3028	014042	001007			BNE	1\$: BRANCH IF NO
3029	014044	016246	000010		MOV	10(R2),-(SP)	: LOAD DESIRED TRACK & SECTOR
3030	014050	004037	017374		JSR	RD,WRT.RP	
3031	014054	000006			RPDA		
3032	014056	014436			CI7		
3033	014060	000403			BR	2\$: GO LOAD CYLINDER
3034	014062	122703	000105	1\$:	CMPB	#105,R3	: IS IT A SEEK COMMAND
3035	014066	001007			BNE	3\$: BRANCH IF NO
3036	014070	016246	000012	2\$:	MOV	12(R2),-(SP)	: LOAD DESIRED CYLINDER
3037	014074	004037	017374		JSR	RD,WRT.RP	
3038	014100	000034			RPCA		
3039	014102	014436			CI7		
3040	014104	000546			BR	CI6	
3041	014106	122703	000115	3\$:	CMPB	#115,R3	: IS IT AN "OFFSET" COMMAND?
3042	014112	001013			BNE	4\$: BR IF NO
3043	014114	004037	017214		JSR	RD,RD.RP	: MERGE THE OFFSET VALUE INTO RPOF
3044	014120	000032			RPOF		: BUT DON'T CHANGE THE UPPER
3045	014122	014436			CI7		
3046	014124	116216	000001		MOV B	1(R2),(SP)	: BYTE WHEN LOADING THE
3047	014130	004037	017374		JSR	RD,WRT.RP	: REGISTER (RPOF)
3048	014134	000032			RPOF		
3049	014136	014436			CI7		
3050	014140	000530			BR	CI6	: GO START THE COMMAND
3051	014142	122703	000107	4\$:	CMPB	#107,R3	: IS IT A "RECALIBRATE" COMMAND?
3052	014146	001525			BEQ	CI6	: BRANCH IF YES
3053	014150	122703	000117		CMPB	#117,R3	: IS IT A RETURN TO CENTER?
3054	014154	001522			BEQ	CI6	: BRANCH IF YES
3055	014156	122703	000103		CMPB	#103,R3	: IS IT AN "UNLOAD" COMMAND?
3056	014162	001016			BNE	5\$: BRANCH IF NO
3057	014164	112761	000001	012110	MOV B	#1,DRVACT(R1)	: SET THE DRIVE ACTIVE INDICATOR
3058	014172	105061	012120		CLRB	DRVSTA(R1)	: PUT DRIVE STATUS TO OFFLINE
3059	014176	112761	000001	012166	MOV B	#1,ULDFLG(R1)	: SET "UNLOAD IN PROGRESS" FLAG
3060	014204	010346			MOV	R3,-(SP)	: START THE "UNLOAD" COMMAND
3061	014206	004037	017374		JSR	RD,WRT.RP	
3062	014212	000000			RPCS1		

H06

3063	014214	014436			CI7			
3064	014216	000207			RTS	PC		;RETURN TO USER
3065	014220	122703	000143		5\$: CMPB	#143,R3		;IS IT A "SET FORMAT" COMMAND?
3066	014224	001014			BNE	6\$;BRANCH IF NO
3067	014226	004037	017214		JSR	RD, RD.RP		;READ THE OFFSET REGISTER
3068	014232	000032			RPOF			
3069	014234	014436			CI7			
3070	014236	116266	000001	000001	MOVb	1(R2), 1(SP)		;COMBINE "FMT22" "ECI" AND "HCI"
3071	014244	004037	017374		JSR	RD, WRT.RP		;LOAD "FMT22", "ECI", AND/OR "HCI".
3072	014250	000032			RPOF			
3073	014252	014436			CI7			
3074	014254	000436			BR	12\$		
3075	014256	122703	000141		6\$: CMPB	#141,R3		;IS IT A "GET REGISTER" COMMAND?
3076	014262	001023			BNE	10\$;BRANCH IF NO
3077	014264	016203	000006		7\$: MOV	6(R2), R3		;POINTS TO 1ST ADDRESS OF WHERE
3078								;TO PUT THE REGISTER(S)
3079	014270	116237	000010	014306	MOVb	10(R2), 9\$;INIT. THE INDEX FOR THE FIRST REG.
3080	014276	116205	000011		MOVb	11(R2), R5		;INDEX OF LAST REG. TO MOVE
3081	014302	004037	017214		9\$: JSR	RD, RD.RP		;READ RPO4/5/6 REGISTER
3082	014306	000000			9\$: RPCS1			;INDEX OF REG. TO READ
3083	014310	014436			CI7			
3084	014312	012623			MOV	(SP)+, (R3)+		;GET THE CONTENTS OF RH11/RPO4/5/6 REG.
3085	014314	023705	014306		CMP	9\$, R5		;LAST REG. BEEN READ?
3086	014320	001414			BEQ	12\$;GET OUT IF YES
3087	014322	062737	000002	014306	ADD	#2, 9\$;INCREASE THE INDEX BY 2
3088	014330	000764			BR	8\$;LOOP--MORE TO READ
3089	014332	122703	000145		10\$: CMPB	#145,R3		;IS IT A "SELECT DRIVE" COMMAND?
3090	014336	001405			BEQ	12\$;BRANCH IF YES
3091	014340	010346			11\$: MOV	R3, -(SP)		;LOAD THE COMMAND
3092	014342	004037	017374		JSR	RD, WRT.RP		
3093	014346	000000			RPCS1			
3094	014350	014436			CI7			
3095	014352	004737	020444		12\$: JSR	PC, POPQUE		;REMOVE REQ. FROM QUEUE
3096	014356	052762	000200	000016	BIS	#BIT07, 16(R2)		;SET THE "DONE" BIT
3097	014364	005737	012206		TST	SAVEFG		;SAVE THE RH11/RPO4/5/6 REGISTERS?
3098	014370	100002			BPL	13\$;BRANCH IF NO
3099	014372	004737	017566		JSR	PC, SVRH11		;YES--GO SAVE THE REGISTERS
3100	014376	000207			13\$: RTS	PC		;RETURN TO USER
3101	014400	006301			CI5:	R1		
3102	014402	012761	001750	012212	MOV	#1000., TIMER(R1)		;SET A ONE SECOND TIMER
3103	014410	006201			ASR	R1		
3104	014412	112761	000001	012110	MOVb	#1, DRVACT(R1)		;SET THE DRIVE ACTIVE
3105	014420	000207			RTS	PC		;RETURN TO THE USER
3106	014422	010346			CI6:	MOV	R3, -(SP)	;LOAD THE COMMAND
3107	014424	004037	017374		JSR	RD, WRT.RP		
3108	014430	000000			RPCS1			
3109	014432	014436			CI7			
3110	014434	000761			BR	CI5		
3111	014436	032764	010000	000010	CI7:	BIT	#BIT12, RPCS2(R4)	;DRIVE NON-EXISTENT ?
3112	014444	001034			BNE	CI8		;BR IF YES
3113	014446	005702			1\$: TST	R2		;ANYTHING IN QUEUE ?
3114	014450	001405			BEQ	CI7B		;BR IF NOT
3115	014452	012762	104000	000016	MOV	#BIT15!BIT11, 16(R2)		;SET "PARITY" ERROR INDICATOR
3116	014460	004737	017566		JSR	PC, SVRH11		;GO SAVE THE RH11/RPO4/5/6 REGISTERS


```

3117 014464 012746 000111          CI7B:  MOV    #111,-(SP)      ;DO A "DRIVE CLEAR"
3118 014470 004037 017374          JSR    RO,WRT.RP
3119 014474 000000          RPCS1
3120 014476 014536          CIB
3121 014500 004737 020326          JSR    PC,EMPTYQ      ;EMPTY THE QUEUE
3122 014504 105061 012166          CLRB  ULDFLG(R1)      ;CLEAR THE UNLOAD IN QUEUE FLAG
3123 014510 105061 012110          CLRB  DRVACT(R1)      ;DRIVE IS IDLE
3124 014514 020137 012232          CMP   R1,DTUW        ;IF THIS DRIVE HAD AN I/O REQUEST
3125 014520 001005          BNE   1$              ;IN PROGRESS CLEAR ALL OF THE FLAGS
3126 014522 005037 012160          CLR   TRNSWT
3127 014526 012737 177777 012232          MOV   #-1,DTUW
3128 014534 000207          RTS   PC
3129 014536 104412          CIB:  SAVREG          ;SAVE R0 - R5
3130 014540 032764 010000 000010          BIT   #BIT12,RPCS2(R4) ;IS 'NED' SET ?
3131 014546 001002          BNE   1$              ;BR IF YES
3132 014550 005001          CLR   R1
3133 014552 005003          CLR   R3
3134 014554 105761 012110          1$:  TSTB  DRVACT(R1)    ;DRIVE ACTIVE?
3135 014560 001443          BEQ   5$              ;BRANCH IF NO
3136 014562 013702 012160          MOV   TRNSWT,R2      ;GET THE "TRANSFER WAIT" QUEUE
3137 014566 020137 012232          CMP   R1,DTUW        ;DID THIS DRIVE HAVE AN I/O IN PROGRESS?
3138 014572 001402          BEQ   2$              ;BRANCH IF YES
3139 014574 004737 020422          JSR   PC,GETREQ      ;GET THE DPB POINTER
3140 014600 005702          2$:  TST   R2              ;QUEUE ENTRY FOR DRIVE ?
3141 014602 001415          BEQ   4$              ;BR IF NOT
3142 014604 032764 010000 000010          BIT   #BIT12,RPCS2(R4) ;'NED' SET ?
3143 014612 001404          BEQ   3$              ;BR IF NOT
3144 014614 012762 100002 000016          MOV   #BIT15:BIT01,16(R2) ;SET 'DRIVE NON-EXISTENT' INDICATOR
3145 014622 000405          BR    4$              ;CONTINUE
3146 014624 012762 102000 000016          3$:  MOV   #BIT15:BIT10,16(R2) ;SET "NON-CLEARABLE PARITY" ERROR INDICATOR
3147 014632 004737 017566          JSR   PC,SVRH11      ;SAVE RH11/RPO4/5/6 REGISTERS
3148 014636 012763 177777 012212          4$:  MOV   #-1,TIMER(R3)   ;STOP THE TIMER
3149 014644 105061 012110          CLRB  DRVACT(R1)      ;SET "DRIVE ACTIVE" TO IDLE
3150 014650 020137 012232          CMP   R1,DTUW        ;IS THIS DRIVE SETUP FOR A TRANSFER
3151 014654 001005          BNE   5$              ;BR IF NOT
3152 014656 012737 177777 012232          MOV   #-1,DTUW      ;RESET THE INDICATOR
3153 014664 005037 012160          CLR   TRNSWT         ;CLEAR THE TRANSFER QUEUE
3154 014670 105061 012166          5$:  CLRB  ULDFLG(R1)    ;CLEAR UNLOAD FLAG
3155 014674 032764 010000 000010          BIT   #BIT12,RPCS2(R4) ;'NED' SET ?
3156 014702 001021          BNE   6$              ;BR IF YES
3157 014704 005201          INC   R1              ;MOVE TO THE NEXT DRIVE
3158 014706 062703 000002          ADD   #2,R3
3159 014712 042701 177770          BIC   #1<7,R1
3160 014716 001316          BNE   1$              ;BRANCH IF MORE DRIVES
3161 014720 012737 177777 012232          MOV   #-1,DTUW      ;NO DATA TRANSFERS UNDERWAY
3162 014726 005037 012160          CLR   TRNSWT         ;CLEAR THE 'TRANSFER WAIT' QUEUE
3163 014732 004737 020250          JSR   PC,CLRQUE      ;CLEAR ALL OF THE REQUEST QUEUES
3164 014736 012764 000040 000010          MOV   #BIT05,RPCS2(R4) ;DO A MASSBUS INIT.
3165 014744 000406          BR    7$              ;CONTINUE
3166 014746 004737 020326          6$:  JSR   PC,EMPTYQ    ;CLEAR THE DRIVE'S QUEUE
3167 014752 105061 012120          CLRB  DRVSTA(R1)     ;SET DRIVE TO OFFLINE
3168 014756 105061 012130          CLRB  DRVTP(R1)      ;CLEAR THE DRIVE TYPE INDICATOR
3169 014762 004737 017704          7$:  JSR   PC,SET.IE    ;SET "IE" WITHOUT "TRE"
3170 014766 104413          RESREG          ;RESTORE R0 - R5
    
```

```

3171 014770 000207          RTS      PC          ;RETURN
3172
3173          ;LOOK AHEAD ROUTINE
3174          ;CALL
3175          ;
3176          ;
3177          ;
3178          ;
3179          ;
3180          ;
3181          ;
3182          ;
3183 014772 013704 012246    LA:     MOV      RPADR,R4      ;GET RPCS1'S ADDRESS
3184 014776 010164 000010    MOV      R1,RPCS2(R4)     ;SELECT DRIVE
3185 015002 004037 017214    JSR      RO,LD.RP        ;READ CURRENT CYLINDER
3186 015006 000036          RPCC          ;
3187 015010 015122          4$          ;ERROR RETURN ADDRESS
3188 015012 022662 000012    CMP      (SP)+,12(R2)     ;IS CURRENT CYLINDER=DESIRED
3189          ;CYLINDER?
3190 015016 001037          BNE      3$              ;EXIT IF NO
3191 015020 105261 012176    INCB    LACNT(R1)         ;INCREMENT THE LOOK AHEAD COUNT
3192 015024 126137 012176 012254    CMPB   LACNT(R1),MXLACT  ;EXCEED MAX?
3193 015032 003026          BGT      2$              ;BRANCH IF YES
3194 015034 116203 000010    MOVB   10(R2),R3         ;GET DESIRED SECTOR ADDRESS AND
3195 015040 000303          SWAB    R3               ;MULT. BY 64--ALIGN WITH
3196 015042 006203          ASR     R3               ;LOOK AHEAD REGISTER
3197 015044 006203          ASR     R3               ;
3198 015046 012737 000340 177776    MOV      #340,3#PS       ;PRIORITY LEVEL "7"
3199 015054 004037 017214    JSR      RO,LD.RP        ;READ LOOK AHEAD REGISTER
3200 015060 000020          RPLA    4$              ;
3201 015062 015122          4$          ;
3202 015064 162603          SUB     (SP)+,R3         ;CALCULATE THE DELTA
3203 015066 002002          BGE     1$              ;
3204 015070 062703 002600    ADD     #(<22.*64.>),R3  ;MAKE THE DELTA POSITIVE
3205 015074 023703 012256    1$:    CMP     MXDLTA,R3      ;CHECK THE DELTA TO SEE
3206 015100 002406          BLT     3$              ;IF IT IS WITHIN THE
3207 015102 023703 012260    CMP     MNDLTA,R3       ;WINDOW---IF YES, ZERO
3208 015106 002003          BGE     3$              ;THE LOOK AHEAD COUNT
3209 015110 105061 012176    2$:    CLRB   LACNT(R1)     ;AND TAKE THE I/O EXIT
3210 015114 005720          TST    (R0)+            ;
3211 015116 005720          3$:    TST    (R0)+            ;ADJUST THE RETURN ADDRESS
3212 015120 000402          BR     5$              ;EXIT
3213 015122 004737 014436    4$:    JSR     PC,C17      ;PROCESS THE ERROR
3214 015126 000200          5$:    RTS     RO          ;RETURN
3215
3216          ;INTERRUPT SERVICE ROUTINE
3217
3218 015130 112737 000001 012164    ISR:   MOVB   #1,ACTDRV   ;SET "ACTIVE DRIVER" FLAG
3219 015136 104412          SAVREG          ;SAVE R0 - R5
3220 015140 013704 012246    MOV     RPADR,R4        ;ADDRESS OF RHSCS1
3221 015144 013701 012232    MOV     DTUW,R1         ;GET "DATA TRANSFER UNDERWAY" INDICATOR
3222 015150 002403          BLT     1$            ;BRANCH IF NO DATA TRANSFER UNDERWAY
3223 015152 004737 015174    JSR     PC,TD          ;CALL TRANSFER DONE
3224 015156 000402          BR     2$            ;EXIT
    
```



```

3225 015160 004737 015334      1$:   JSR    PC,SC      ;CALL SPECIAL CONDITIONS
3226 015164 104413      2$:   RESREG          ;RESTORE R0 - R5
3227 015166 105037 012164      CLRB   ACTDRV      ;CLEAR "ACTIVE DRIVER" FLAG
3228 015172 000002      RTI              ;RETURN
3229
3230      ;TRANSFER DONE ROUTINE
3231
3232 015174 105061 012110      TD:   CLRB   DRVACT(R1) ;SET DRIVE ACTIVE INDICATOR TO IDLE
3233 015200 012737 177777 012232      MOV    #-1,DTUW      ;NO DATA TRANSFERS UNDERWAY
3234 015206 006301      ASL    R1
3235 015210 012761 177777 012212      MOV    #-1,TIMER(R1) ;CANCEL TIMEOUT
3236 015216 006201      ASR    R1
3237 015220 013702 012160      MOV    TRNSWT,R2     ;GET "DPB" ADDRESS FROM THE
3238 015224 005037 012160      CLR    TRNSWT        ;TRANSFER WAIT QUEUE--CLEAR QUEUE
3239 015230 052762 000200 000016      BIS    #BIT07,16(R2) ;SET DONE
3240 015236 010164 000010      MOV    R1,RPC52(R4)  ;SELECT THE DRIVE
3241 015242 004037 017214      JSR    R0,RD.RP      ;TRANSFER ERROR(TRE=1)?
3242 015246 000000      RPCS1
3243 015250 014436      C17
3244 015252 006126      ROL    (SP)+
3245 015254 100413      BMI    3$           ;BR IF YES
3246 015256 005737 012206      TST    SAVEFG        ;SAVE THE RH11/RPO4/5/6 REGISTERS?
3247 015262 100002      BPL    1$           ;BRANCH IF NO
3248 015264 004737 017566      JSR    PC,SVRH11     ;YES--SAVE THE REGISTERS
3249 015270 004737 013326      1$:   JSR    PC,OPT      ;CALL OPTIMIZER
3250 015274 000417      BR     SC           ;CHECK OTHER DRIVES
3251 015276 012714 000113      2$:   MOV    #113,(R4)  ;RELEASE THE DRIVE
3252 015302 000414      BR     SC           ;CHECK FOR OTHER DRIVES
3253 015304 052762 100100 000016 3$:   BIS    #BIT15:BIT06,16(R2) ;SET DATA ERROR FLAG
3254 015312 004737 020326      JSR    PC,EMPTYQ     ;EMPTY THE "DRIVE'S WAIT" QUEUE
3255 015316 004737 017566      JSR    PC,SVRH11     ;SAVE THE RH11/RPO4/5/6 REGISTERS
3256 015322 012714 040111      MOV    #40111,(R4)  ;ISSUE A "DRIVE CLEAR"
3257 015326 012714 000113      MOV    #113,(R4)    ;ISSUE A RELEASE TO THE DRIVE
3258 015332 000400      BR     SC           ;CHECK FOR OTHER DRIVES
3259
3260      ;SPECIAL CONDITION ROUTINE
3261
3262 015334 116403 000016      SC:   MOVB   RPAS(R4),R3 ;READ "RPAS"
3263 015340 001014      BNE    2$           ;BRANCH IF ANY 'ATA' BITS SET
3264 015342 004037 017214      JSR    R0,RD.RP      ;READ CONTROL AND STATUS REGISTER
3265 015346 000000      RPCS1
3266 015350 014536      C18
3267 015352 106126      ROLB   (SP)+
3268 015354 100405      BMI    1$           ;IS "IE"=1?
3269 015356 004037 020506      JSR    R0,ES.SAV     ;YES, NO DRIVES TO CHECK
3270 015362 104001      ERROR  1            ;SAVE THE ADDRESS IN 'SESCAPE'
3271 015364 004737 017704      JSR    PC,SET.IE     ;REPORT AN ILLEGAL INTERRUPT
3272 015370 000207      1$:   RTS    PC         ;SET INTERRUPT ENABLE
3273 015372 005046      2$:   CLR    -(SP)      ;RETURN
3274 015374 110316      MOVB   R3,(SP)      ;PROCESS ALL DRIVES THAT HAVE
3275 015376 012703 000001      MOV    #1,R3        ;AN "ATA"=1
3276 015402 005001      CLR    R1
3277 015404 030316      SC3:  BIT    R3,(SP)   ;ATA=1?
3278 015406 001005      BNE    SC5          ;YES--BRANCH

```

```

3279 015410 005201          SC4:  INC      R1          ;MOVE TO THE NEXT DRIVE
3280 015412 106303          ASLB     R3
3281 015414 001373          BNE     SC3          ;BRANCH IF MORE TO CHECK?
3282 015416 005726          TST     (SP)+        ;CLEAN OFF THE STACK
3283 015420 000207          RTS     PC          ;RETURN TO USER
3284 015422 105761 012140        SC5:  TSTB    DPINT(R1)   ;INITIALIZING THE DRIVE ?
3285 015426 001402          BEQ     1$          ;BR IF NOT
3286 015430 000137 016326        JMP     SC13        ;PROCESS THE DRIVE
3287 015434 105761 012150        1$:  TSTB    DPRQS(R1)  ;PORT REQUEST OUTSTANDING ?
3288 015440 001402          BEQ     2$          ;BR IF NOT
3289 015442 000137 016326        JMP     SC13        ;START THE OUTSTANDING COMMAND
3290 015446 105761 012120        2$:  TSTB    DRVSTA(R1) ;CHECK THE DRIVE STATUS
3291 015452 003025          BGT     5$          ;BRANCH IF ONLINE
3292 015454 105761 012166        TSTB    ULDFLG(R1) ;UNLOAD IN PROGRESS?
3293 015460 003422          BLE     5$          ;BRANCH IF NOT
3294 015462 004737 020422        JSR     PC,GETREQ   ;GET DPB POINTER
3295 015466 004737 017566        JSR     PC,SVRH11  ;SAVE THE RH11/RPO4/5/6 REGISTERS
3296 015472 004737 016256        JSR     PC,SC12    ;SAVE RPDS1, RPER1, RPER2, AND RPER3
3297                                     ;ALSO DO A DRIVE INIT (DRVINT)
3298 015476 105761 012120        TSTB    DRVSTA(R1) ;DID DRIVE COME ONLINE?
3299 015502 003416          BLE     6$          ;NO---BRANCH
3300 015504 032737 040000 012100        BIT     #BIT14,RPERRS ;WAS THERE AN ERROR?
3301 015512 001002          BNE     3$          ;BR IF ERROR
3302 015514 000137 016166        JMP     SC11        ;NO ERROR
3303 015520 013705 012102        3$:  MOV     RPERRS+2,R5 ;YES -- PICKUP RPER1 AND
3304 015524 000502          BR      SC6A        ;GO PROCESS THE ERROR
3305 015526 105761 012110        5$:  TSTB    DRVACT(R1) ;DRIVE ACTIVE WITH COMMAND OR ERROR RECOVERY ?
3306 015532 001033          BNE     SC6          ;BR IF EITHER
3307 015534 004737 016256        JSR     PC,SC12    ;SAVE RPDS1, RPER1, RPER2, AND RPER3
3308                                     ;ALSO DO A DRVINT
3309 015540 105761 012140        6$:  TSTB    DPINT(R1)   ;TRYING TO INIT THE DRIVE ?
3310 015544 001321          BNE     SC4          ;BR IF YES, CHECK ON MORE DRIVES
3311 015546 105761 012120        TSTB    DRVSTA(R1) ;CHECK ON DRIVE'S STATUS
3312 015552 100412          BMI     7$          ;BR IF UNSAFE
3313 015554 032737 020000 012106        BIT     #BIT13,RPERRS+6 ;ADDRESS PLUG CHANGED ?
3314 015562 001013          BNE     8$          ;BR IF YES
3315 015564 012746 000113        MOV     #113,-(SP) ;RELEASE COMMAND
3316 015570 004037 017374        JSR     RO,WRT.RP  ;WRITE THE COMMAND INTO RPCS1
3317 015574 000000          RPCS1 ;REGISTER INDEX
3318 015576 016136          SC8     ;PARITY EXIT ADDRESS
3319 015600 011605        7$:  MOV     (SP),R5    ;PICKUP (RPAS) BEFORE THE ERROR CALL
3320 015602 004037 020506        JSR     RO,ES.SAV  ;SAVE THE ADDRESS IN 'SESCAPE'
3321 015606 104002          ERROR  2          ;REPORT THE UNEXPECTED ATTENTION
3322 015610 000677          BR      SC4          ;GO CHECK FOR MORE ATA'S
3323 015612                                     ;
3324 015612 004037 020506        8$:  JSR     RO,ES.SAV  ;SAVE THE ADDRESS IN 'SESCAPE'
3325 015616 104005          ERROR  5          ;REPORT THE ADDRESS PLUG CHANGE
3326 015620 000673          BR      SC4          ;CHECK FOR MORE DRIVES
3327 015622 006301          ASL     R1          ;SETUP TO ADDRESS WORDS
3328 015624 012761 177777 012212        MOV     #-1,TIMER(R1) ;STOP THE TIMER
3329 015632 006201          ASR     R1          ;RESTORE THE DRIVE ADDRESS
3330 015634 004737 020422        JSR     PC,GETREQ  ;GET THE DPB POINTER FROM THE QUEUE
3331 015640 010164 000010        MOV     R1,RPCS2(R4) ;SELECT DRIVE
3332 015644 004037 017214        JSR     RO,RO.RP   ;READ THE RPO4'S STATUS REG.

```



```

3333 015650 000012          RPDS1
3334 015652 016136          SC8
3335 015654 011605          MOV      (SP),R5          ;AND PUT IT IN R5
3336 015656 006126          ROL      (SP)+          ;WAS THERE AN ERROR?
3337 015660 100407          BMI      1$            ;BR IF ERROR
3338 015662 105761 012110      TSTB    DRVACT(R1)      ;CHECK DRIVE'S STATE
3339 015666 003137          BGT      SC11          ;BR IF DRIVE ACTIVE WITH ORDER
3340 015670 052762 100210 000016  BIS      #BIT15!BIT07!BIT03,16(R2) ;INFORM USER OF ERROR RECOVER COMPLETION
3341 015676 000470          BR      SC7
3342 015700 004037 017214  1$:      JSR      RO,RD.RP      ;READ ERROR REGISTER #1
3343 015704 000014          RPER1
3344 015706 016136          SC8
3345 015710 012605          MOV      (SP)+,R5      ;AND SAVE IT IN R5
3346 015712 004737 017566      JSR      PC,SVRH11     ;SAVE RH11/RPO4/5/6 REGISTERS
3347 015716 012746 000111      MOV      #111,-(SP)    ;ISSUE A DRIVE CLEAR
3348 015722 004037 017374      JSR      RO,WRT.RP
3349 015726 000000          RPCS1
3350 015730 016136          SC8
3351 015732 006105          SC6A:  ROL      R5          ;WAS "UNSAFE" CONDITION =1?
3352 015734 100406          BMI      1$            ;BRANCH IF YES
3353 015736 005702          TST      R2            ;ANYTHING IN QUEUE ?
3354 015740 001447          BEQ      SC7           ;BR IF NOT
3355 015742 052762 100240 000016  BIS      #BIT15!BIT07!BIT05,16(R2) ;INFORM USER OF ERROR
3356 015750 000443          BR      SC7
3357 015752 004037 017214  1$:      JSR      RO,RD.RP      ;READ DRIVE STATUS REG. #1
3358 015756 000012          RPDS1
3359 015760 016136          SC8
3360 015762 011605          MOV      (SP),R5      ;SAVE RPDS1 IN R5
3361 015764 006126          ROL      (SP)+          ;"ERR"=1?
3362 015766 100011          BPL      2$            ;BR IF NO--UNSAFE CLEARED
3363 015770 112761 177777 012120  MOVB    #-1,DRVSTA(R1) ;DRIVE IS UNSAFE
3364 015776 004737 017566      JSR      PC,SVRH11     ;SAVE RH11/RPO4/5/6 REGISTERS
3365 016002 052762 110000 000016  BIS      #BIT15!BIT12,16(R2) ;INFORM USER OF UNSAFE ERROR
3366 016010 000423          BR      SC7
3367 016012 032705 010000  2$:      BIT      #BIT12,R5      ;"MOL" = 1 ?
3368 016016 001015          BNE      3$            ;BR IF YES
3369 016020 112761 177777 012110  MOVB    #-1,DRVACT(R1) ;ACTIVE ERROR RECOVER
3370 016026 112761 000001 012120  MOVB    #1,DRVSTA(R1)  ;ONLINE
3371 016034 006301          ASL      R1
3372 016036 012761 072460 012212  MOV      #30000.,TIMER(R1) ;START 30 SECOND TIMER
3373 016044 006201          ASR      R1
3374 016046 000137 015410          JMP      SC4
3375 016052 052762 100220 000016  3$:      BIS      #BIT15!BIT07!BIT04,16(R2) ;INFORM USER OF ERROR
3376 016060 105061 012110  SC7:  CLRB    DRVACT(R1)      ;DRIVE IS IDLE
3377 016064 004737 020326      JSR      PC,EMPTYQ     ;DUMP THE QUEUE
3378 016070 105761 012166      TSTB    ULDFLG(R1)     ;UNLOAD IN PROGRESS OR QUEUE?
3379 016074 003002          BGT      1$            ;BR IF NOT
3380 016076 105061 012166      CLRB    ULDFLG(R1)     ;CLEAR UNLOAD FLAG
3381 016102 116164 012234 000016  1$:      MOVB    ATABIT(R1),RPAS(R4) ;CLEAR ATTENTION BIT
3382 016110 105761 012120      TSTB    DRVSTA(R1)     ;IS THE DRIVE UNSAFE ?
3383 016114 100406          BMI      2$            ;BR IF IT IS
3384 016116 012746 000113      MOV      #113,-(SP)    ;RELEASE COMMAND
3385 016122 004037 017374      JSR      RO,WRT.RP      ;WRITE THE COMMAND INTO RPCS1
3386 016126 000000          RPCS1
    
```



```

016441 016426 004737 020422          JSR    PC,GETREQ      ;GET DPB ADDRESS
016442 016432 052762 140000 000016    BIS    #BIT15:BIT14,16(R2) ;INFORM USER THAT DRIVE OFFLINE
016443 016440 004737 017566          JSR    PC,SVRH11     ;SAVE THE REGISTERS
016444 016444 004737 020326          JSR    PC,EMPTYQ    ;EMPTY THE REQUEST QUEUE
016445 016450 000404          BR     3$           ;
016446 016452 105061 012150 2$:    CLRB  DPRQS(R1)     ;CLEAR THE PORT REQUEST INDICATOR
016447 016456 004737 013326          JSR    PC,OPT       ;START THE PENDING REQUEST
016448 016462 000137 015410 3$:    JMP   SC4          ;PROCESS OTHER DRIVES

;RPO4/5/6 TIMER ROUTINE
:CALL
:    MOV    #TIME,-(SP)      ;ELAPSED TIME IN MILLISECONDS ON THE STACK
:    JSR    PC,RPTMR       ;CALL RPO4/5/6 TIME ROUTINE

RPTMR: TST    ACTDRV        ;CHECK "ACTDRV" & "ACTSTR"
        BNE    4$          ;IF NON ZERO EXIT
        MOVB  #1,ACTSTR    ;SET "ACTSTR"
        SAVREG             ;SAVE R0 - R5
        CLR   R1           ;START WITH DRIVE 0
        CLR   R3
1$:    TST    TIMER(R3)    ;IS THE TIMER RUNNING?
        BLT   2$          ;BRANCH IF NO
        SUB   2(SP),TIMER(R3) ;COUNT THE INTERVAL
        BGT   2$          ;BR IF NO SOFTWARE TIMEOUT
        JSR   PC,STO       ;CALL SOFTWARE TIMEOUT ROUTINE
        BR   3$          ;GO TO THE EXIT
2$:    INC   R1           ;MOVE TO NEXT DRIVE
        TST   (R3)+
        CMP   #8,R1       ;OUT OF DRIVES?
        BGT   1$          ;BRANCH IF NO
3$:    RESREG             ;RESTORE R0 - R5
        CLRB  ACTSTR      ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
4$:    MOV   (SP)+,(SP)    ;ADJUST THE STACK
        RTS   PC          ;RETURN

;SOFTWARE TIMEOUT ROUTINE
:NOTE: THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
:OR GREATER
:CALL:  STO
:        MOV   #DRVNUM,R1 ;DRIVE NUMBER
:        JSR   PC,STO     ;CALL
:        RETURN

STO:   MOV   R1,-(SP)     ;SAVE R1
        MOV   R3,-(SP)     ;SAVE R3
        MOV   RPADR,R4    ;GET ADDRESS OF "RPCS1"
        MOV   R1,RPCS2(R4) ;SELECT THE DRIVE
        JSR   RD,RD.RP    ;READ "DRIVE STATUS REG"
        RPDS1
        STOS
        TSTB  (SP)+
        BMI  ST02        ;IS "DRY"=1?
                        ;BR IF YES
    
```

3495	016610	105761	012140		ST01:	TSTB	DPINT(R1)	: TRYING TO INITIALIZE THE DRIVE ?
3496	016614	001074				BNE	ST02	: BR IF YES
3497	016616	105761	012150			TSTB	DPRQS(R1)	: OUTSTANDING PORT REQUEST FOR THE DRIVE ?
3498	016622	001071				BNE	ST02	: BR IF YES
3499	016624	013702	012160			MOV	TRNSWT,R2	: PICKUP TRANSFER WAIT QUEUE
3500	016630	020137	012232			CMP	R1,DTUW	: TRANSFER UNDERWAY ON THIS DRIVE?
3501	016634	001402				BEQ	1\$: BRANCH IF YES
3502	016636	004737	020422			JSR	PC,GETREQ	: GET DPB ADDRESS
3503	016642	052762	101000	000016	1\$:	BIS	#BIT15!BIT09,16(R2)	: SET THE ERROR FLAGS
3504	016650	004737	017566			JSR	PC,SVRH11	: SAVE RH11/RPO4/5/6 REGISTERS
3505	016654	012764	000040	000010		MOV	#BIT05,RPCS2(R4)	: "INIT" THE MASS BUS
3506	016662	105061	012110			CLRB	DRVACT(R1)	: DRIVE IS IDLE
3507	016666	105061	012166			CLRB	ULDFLG(R1)	: CLEAR THE UNLOAD FLAG
3508	016672	005001				CLR	R1	: START WITH DRIVE 0
3509	016674	005003				CLR	R3	
3510	016676	004037	012476		2\$:	JSR	RD,DRVINT	: INIT. THIS DRIVE
3511	016702	000477				BR	ST05	: PARITY ERROR RETURN
3512	016704	105761	012110			TSTB	DRVACT(R1)	: DRIVE IDLE BEFORE THE INIT.?
3513	016710	001414				BEQ	4\$: YES--BRANCH
3514	016712	013702	012160			MOV	TRNSWT,R2	: GET TRANSFER WAIT QUEUE
3515	016716	023701	012232			CMP	DTUW,R1	: WAS THERE I/O ON THIS DRIVE?
3516	016722	001402				BEQ	3\$: YES--BRANCH
3517	016724	004737	020422			JSR	PC,GETREQ	: GET THE DPB POINTER FROM QUEUE
3518	016730	052762	100400	000016	3\$:	BIS	#BIT15!BIT08,16(R2)	: INFORM USER OF INIT.
3519	016736	105061	012110			CLRB	DRVACT(R1)	: SET DRIVE ACTIVE TO IDLE
3520	016742	105061	012166		4\$:	CLRB	ULDFLG(R1)	: NO UNLOAD
3521	016746	012763	177777	012212		MOV	#-1,TIMER(R3)	: STOP THE TIMER
3522	016754	005723				TST	(R3)+	: UPDATE THE INDEX
3523	016756	005201				INC	R1	: INCREMENT THE DRIVE NUMBER
3524	016760	022701	000010			CMP	#8.,R1	: LAST DRIVE BEEN CHECKED?
3525	016764	003344				BGT	2\$: NO--LOOP
3526	016766	012737	177777	012232		MOV	#-1,DTUW	: NO DATA TRANSFERS UNDERWAY
3527	016774	005037	012160			CLR	TRNSWT	: CLEAR TRANSFER WAIT QUEUE
3528	017000	004737	020250			JSR	PC,CLRQUE	: CLEAR ALL REQUEST QUEUES
3529	017004	000500				BR	ST09	: EXIT
3530	017006	116405	000016		ST02:	MOVB	RPAS(R4),R5	: READ ATTENTION REG
3531	017012	136105	012234			BITB	ATABIT(R1),R5	: IS ATTENTION FOR THIS DRIVE UP ?
3532	017016	001017				BNE	ST03	: YES--BRANCH
3533	017020	105761	012140			TSTB	DPINT(R1)	: TRYING TO INITIALIZE THE DRIVE ?
3534	017024	001031				BNE	ST06	: BR IF YES - DRIVE NOT ONLINE
3535	017026	105761	012150			TSTB	DPRQS(R1)	: OUTSTANDING PORT REQUEST FOR THE DRIVE ?
3536	017032	001045				BNE	ST07	: BR IF YES - NO RESPONSE TO REQUEST
3537	017034	020137	012232			CMP	R1,DTUW	: DATA TRANSFER UNDERWAY FOR THIS DRIVE
3538	017040	001263				BNE	ST01	: BR IF NO
3539	017042	004037	017214			JSR	RD,RD.RP	: YES--CHECK "RDY"
3540	017046	000000				RPCS1		
3541	017050	017102				ST05		
3542	017052	105726				TSTB	(SP)+	
3543	017054	100255				BPL	ST01	: BR IF "RDY"=0
3544	017056	105761	012140		ST03:	TSTB	DPINT(R1)	: INITIALIZING THE DRIVE ?
3545	017062	001003				BNE	1\$: BR IF INIT PENDING
3546	017064	105761	012150			TSTB	DPRQS(R1)	: PORT REQUEST PENDING ?
3547	017070	001446				BEQ	ST09	: BR IF NOT
3548	017072	012763	177777	012212	1\$:	MOV	#-1,TIMER(R3)	: STOP THE TIMER


```

3549 017100 000442          BR      ST09          ;EXIT
3550 017102 004737 014536 ST05: JSR      PC,C18      ;GO HANDLE THE PARITY ERROR
3551 017106 000437          BR      ST09
3552 017110 105061 012140 ST06: CLRB   DPINT(R1)   ;CLEAR THE INITIALIZE INDICATOR
3553 017114 105061 012120 CLRB   DRVSTA(R1)   ;SET UNIT OFFLINE
3554 017120 012763 177777 012212 MOV    #-1,TIMER(R3) ;STOP THE TIMER
3555 017126 004737 020422 JSR    PC,GETREQ    ;GET THE DPB ADDRESS
3556 017132 005702          TST    R2           ;REQUEST IN QUEUE ?
3557 017134 001424          BEQ    ST09         ;BR IF NOT
3558 017136 052762 140000 000016 BIS    #BIT15:BIT14,16(R2) ;INFORM THE USER DRIVE NOT AVAILABLE
3559 017144 000414          BR      ST08
3560 017146 012763 177777 012212 ST07: MOV    #-1,TIMER(R3) ;STOP THE TIMER
3561 017154 105061 012150 CLRB   DPRQS(R1)   ;CLEAR PORT REQUEST INDICATOR
3562 017160 004737 020422 JSR    PC,GETREQ    ;GET DPB ADDRESS
3563 017164 005702          TST    R2           ;QUEUE ENTRY FOR DRIVE ?
3564 017166 001407          BEQ    ST09         ;BR IF NONE
3565 017170 012762 100004 000016 MOV    #BIT15:BIT2,16(R2) ;INFORM USER OF PORT REQUEST ERROR
3566 017176 004737 020326 ST08: JSR    PC,EMPTYQ ;CLEAR THE QUEUE FOR THE DRIVE
3567 017202 004737 017566 JSR    PC,SVRH11    ;SAVE THE REGISTERS
3568 017206 012603 ST09: MOV    (SP)+,R3 ;RESTORE R3
3569 017210 012601 MOV    (SP)+,R1    ;RESTORE R1
3570 017212 000207          RTS     PC         ;RETURN

```

:ROUTINE TO READ A RH11/RPO4/5/6 REGISTER

```

3571          :CALL
3572          :JSR    RD, RD.RP ;GO READ A REGISTER
3573          :INDEX ;REG. INDEX FROM BASE
3574          :ERRADR ;ERROR ADDRESS--PROCESS ERROR STARTING
3575          :RETURN ;AT THIS ADDRESS
3576          :CONTENTS OF REG. IS ON THE STACK
3577
3578 RD.RP: MOV    MCPEMX, RD.RP2 ;MAX. RETRYS ALLOWED
3579          MOV    (SP), -(SP) ;SAVE RD FOR RETURN
3580          MOV    RPADR, RD.ADR ;FORM THE DESIRED ADDRESS
3581          ADD    (RD)+, RD.ADR ;USING THE BASE AND THE INDEX
3582 RD.RP1: MOV    @((PC)+), (PC)+ ;READ THE DESIRED REGISTER OF THE RPO4
3583 RD.ADR: .WORD 0 ;ADDRESS IS FORMED HERE
3584 RD.WRD: .WORD 0 ;REG. CONTENTS PUT HERE
3585          MOV    RD.WRD, 2(SP) ;RETURN IT TO THE USER
3586          MOV    RPADR, -(SP) ;PUT THE ADDRESS ON THE STACK
3587          ADD    #RPCS2, (SP) ;FORM THE ADDRESS OF RPCS2
3588          BIT    #BIT12, @((SP)+) ;CHECK THE 'NED' BIT
3589          BNE   RD.RP3 ;BR IF DRIVE NON-EXISTENT
3590          MOV    @RPADR, -(SP) ;READ RPCS1
3591          BIT    #BIT13, (SP) ;DID MCPE SET?
3592          BNE   IS ;BRANCH IF YES
3593          CMP    (SP)+, (RD)+ ;ADJUST FOR RETURN
3594          BR    RD.RP4 ;EXIT
3595
3596 IS: JSR    RD, ES.SAV ;SAVE THE ADDRESS IN '$ESCAPE'
3597          ERROR 3 ;REPORT "MCPE" ERROR
3598          TST   DTUW ;DATA TRANSFER UNDERWAY?
3599          BMI  2$ ;NO--BRANCH

```

E07

3603	017322	032716	040000			BIT	#BIT14,(SP)	:NO--"TRE"=1?
3604	017326	001402				BEQ	2\$:NO--BRANCH
3605	017330	005726				TST	(SP)+	:YES--CLEAN OFF THE STACK AND
3606	017332	000415				BR	RD.RP3	:TAKE THE FATAL ERROR EXIT
3607	017334	052716	040000		2\$:	BIS	#BIT14,(SP)	:CLEAR "MCPE" BY SENDING A "1" TO "TRE"
3608	017340	000316				SWAB	(SP)	:POSITION BEFORE WRITING
3609	017342	013737	012246	017356		MOV	RPADR,3\$:FORM ADDRESS OF HIGH BYTE
3610	017350	005237	017356			INC	3\$	
3611	017354	112637				MOVB	(SP)+,2(PC)+	:WRITE THE HIGH BYTE OF RPCS1
3612	017356	000000			3\$:	.WORD	0	:ADDRESS STORAGE
3613	017360	005327				DEC	(PC)+	:EXCEEDED MAX. RETRYS
3614	017362	000003			RD.RP2:	.WORD	3	
3615	017364	002324				BGE	RD.RP1	:BRANCH IF NO
3616	017366	011000			RD.RP3:	MOV	(RD),RD	:FATAL ERROR EXIT
3617	017370	012616				MOV	(SP)+,(SP)	
3618	017372	000200			RD.RP4:	RTS	RD	
3619								:ROUTINE TO WRITE A REGISTER
3620								:CALL
3621						MOV	DATA,-(SP)	:DATA TO BE LOADED ON THE STACK
3622						JSR	RD,WRT.RP	:CALL THE ROUTINE TO LOAD(WRITE) THE REG.
3623						INDEX		:INDEX OF THE REGISTER TO BE LOADED
3624						ERRADR		:ADDRESS TO RETURN TO ON AN ERROR
3625						RETURN		:ERROR FREE RETURN
3626								
3627								
3628								
3629	017374	013737	012244	017550	WRT.RP:	MOV	MCPEMX,WRT.R2	:MAX RETRYS ALLOWED
3630	017402	016637	000002	017462		MOV	2(SP),WRT.WD	:SAVE THE WORD TO WRITE
3631	017410	012616				MOV	(SP)+,(SP)	:ADJUST THE STACK
3632	017412	012037	017464			MOV	(RD)+,WRT.AD	:GET INDEX OF REGISTER TO BE WRITTEN
3633	017416	001015				BNE	1\$:BRANCH IF NOT RPCS1
3634	017420	122737	000150	017462		CMPEB	#150,WRT.WD	:IS THE COMMAND FOR DATA TRANSFERS?
3635	017426	002411				BLT	1\$:YES--DON'T GET THE OLD A16 & A17, & PSEL
3636	017430	004037	017214			JSR	RD,RD.RP	:NO---COMBINE A16&A17, & PSEL WITH
3637	017434	000000				RPCS1		:THE COMMAND BEFORE SENDING IT TO
3638	017436	017556				WRT.R3		:THE RH11/RPO4
3639	017440	000316				SWAB	(SP)	
3640	017442	042716	177770			BIC	#1C7,(SP)	
3641	017446	112637	017463			MOVB	(SP)+,WRT.WD+1	
3642	017452	063737	012246	017464	1\$:	ADD	RPADR,WRT.AD	:FORM THE ADDRESS OF THE DISK REG.
3643	017460	012737			WRT.R1:	MOV	(PC)+,2(PC)+	:LOAD THE DESIRED REG.
3644	017462	000000			WRT.WD:	.WORD	0	:WORD TO WRITE GOES HERE
3645	017464	000000			WRT.AD:	.WORD	0	:ADDRESS IS FORMED HERE
3646	017466	013746	012246			MOV	RPADR,-(SP)	:PUT THE ADDRESS ON THE STACK
3647	017472	062716	000010			ADD	#RPCS2,(SP)	:FORM THE ADDRESS OF RPCS2
3648	017476	032736	010000			BIT	#BIT12,2(SP)+	:CHECK THE 'NED' BIT
3649	017502	001025				BNE	WRT.R3	:BR IF DRIVE NON-EXISTENT
3650	017504	004037	017214			JSR	RD,RD.RP	:CHECK FOR PARITY ERROR ON WRITE
3651	017510	000014				RPER1		
3652	017512	017556				WRT.R3		
3653	017514	032726	000010			BIT	#BIT03,(SP)+	
3654	017520	001420				BEQ	WRT.R4	:BRANCH IF "PAR=0"
3655	017522	016037	177776	017534		MOV	-2(RD),1\$:PICKUP THE INDEX
3656	017530	004037	017214			JSR	RD,RD.RP	:READ THE REG.


```

3657 017534 000000 1$: .WORD 0 ;REG. INDEX
3658 017536 017556 WRT.R3 ;RETURN TO THIS ADDRESS ON ERROR
3659 017540 004037 020506 JSR RO,ES.SAV ;SAVE THE ADDRESS IN 'SESCAPE'
3660 017544 104004 ERROR 4 ;REPORT THE PARITY ON WRITE ERROR
3661 017546 005327 DEC (PC)+ ;DECREMENT THE ERROR COUNT
3662 017550 000003 WRT.R2: .WORD 3 ;RETRY COUNTER
3663 017552 002342 BGE WRT.R1 ;TRY AGAIN IF NOT FINISHED
3664 017554 005726 TST (SP)+ ;CLEAN OFF THE STACK
3665 017556 011000 WRT.R3: MOV (RO),RO ;TAKE THE "PARITY ON WRITE" ERROR EXIT
3666 017560 000401 BR WRT.R5 ;EXIT
3667 017562 005720 WRT.R4: TST (RO)+ ;ADJUST FOR ERROR FREE EXIT
3668 017564 000200 WRT.R5: RTS RO
3669
3670 ;ROUTINE TO $AVE THE RH11/RPO4/5/6 REGISTERS AS PER DPB+14
3671
3672 ;CALL
3673
3674 ; MOV #DPBNUM,R2 ;DPB POINTER TO R2
3675 ; JSR PC,SVRH11 ;SAVE THE DRIVES REG'S
3676
3677 SVRH11: SAVREG ;SAVE RO - R5
3678 TST R2 ;QUEUE ENTRY FOR THE DRIVE ?
3679 BEQ 4$ ;BR IF NONE
3680 017572 001430 MOV RPADR,R4
3681 017574 013704 012246 MOVB (R2),RPCS2(R4) ;SELECT DRIVE
3682 017600 111264 000010 MOV #4(R2),R3 ;GET THE ERROR TABLE POINTER
3683 017604 016203 000014 BEQ 6$ ;EXIT IF NO ADDRESS
3684 017610 001433 CLR 3$ ;COUNTER & POINTER
3685 017612 005037 017646 1$: CMP 3$,#RPDB ;REACHED THE BUFFER REGISTER ?
3686 017616 023727 017646 000022 BNE 2$ ;BR IF NOT
3687 017624 001006 BIT #BIT07,RPCS2(R4) ;'OR' SET ?
3688 017626 032764 000200 000010 BNE 2$ ;BR IF SET
3689 017634 001002 CLR (R3)+ ;STORE RPDB AS ZEROES
3690 017636 005023 BR 4$ ;CONTINUE
3691 017640 000405 JSR RO,RO.RP ;READ THE SELECTED REGISTER
3692 017642 004037 017214 2$: .WORD 0 ;REGISTER INDEX
3693 017644 000000 3$: 5$ ;ERROR RETURN ADDRESS
3694 017650 017674 MOV (SP)+,(R3)+ ;STORE THE REGISTER CONTENTS
3695 017652 012623 4$: CMP 3$,#RPEC2 ;REACHED THE END ?
3696 017654 023727 017646 000046 BEQ 6$ ;BR IF YES
3697 017662 001406 ADD #2,3$ ;INCREMENT THE REGISTER INDEX
3698 017664 062737 000002 017646 BR 1$ ;CONTINUE READING THE REGISTERS
3699 017672 000751 5$: JSR PC,C17 ;PROCESS THE UNCORRECTABLE PARITY ERROR
3700 017674 004737 014436 6$: RESREG ;RESTORE RO - R5
3701 017700 104413 RTS PC ;RETURN
3702
3703 ;ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A "TRE"
3704 ;CALL
3705
3706 ; MOV #DRVNUM,R1 ;DRIVE NUMBER TO R1
3707 ; JSR PC,SET.IE ;SET "IE"
3708 ; RETURN
3709
3710 SET.IE: MOV R4,-(SP) ;SAVE R4
3711 MOV RPADR,R4 ;PICKUP ADDRESS OF RPCS1
3712 MOV R1,RPCS2(R4) ;SELECT DRIVE
    
```

```

3711 017716 011446      MOV      (R4),-(SP)      ;READ RPCS1
3712 017720 052716 040000  BIS      #BIT14,(SP)    ;SET THE "TRE" BIT OF THE WORD READ
3713 017724 000316      SWAB     (SP)           ;ADJUST FOR DATO
3714 017726 112714 000100  MOVB    #BIT06,(R4)    ;SET "IE"
3715 017732 032764 010000 000010  BIT     #BIT12,RPCS2(R4) ;IS "NED"=1?
3716 017740 001002      BNE     1$             ;YES--CLEAR "TRE"
3717 017742 005726      TST     (SP)+          ;CLEAN OFF THE STACK
3718 017744 000402      BR      2$
3719 017746 112664 000001 1$:  MOVB   (SP)+,1(R4)    ;CLEAR "TRE"
3720 017752 012604 2$:  MOV    (SP)+,R4       ;RESTORE R4
3721 017754 000207      RTS     PC             ;RETURN TO CALLER

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

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

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

QSTART: .WORD QDRV0    ;DRIVE 0 START ADDRESS
QSTOP:  .WORD QDRV1    ;DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
      .WORD QDRV2    ;STOP DRIVE 1--START DRIVE 2
      .WORD QDRV3    ;STOP DRIVE 2--START DRIVE 3
      .WORD QDRV4    ;STOP DRIVE 3--START DRIVE 4
      .WORD QDRV5    ;STOP DRIVE 4--START DRIVE 5
      .WORD QDRV6    ;STOP DRIVE 5--START DRIVE 6
      .WORD QDRV7    ;STOP DRIVE 6--START DRIVE 7
      .WORD QTERM

```



```

3765          :DRIVE REQUEST QUEUES
3766          QDRV0: .BLKW  10
3767          QDRV1: .BLKW  10
3768          QDRV2: .BLKW  10
3769          QDRV3: .BLKW  10
3770          QDRV4: .BLKW  10
3771          QDRV5: .BLKW  10
3772          QDRV6: .BLKW  10
3773          QDRV7: .BLKW  10
3774          QTERM=.
3775
3776          :ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
3777          :CALL
3778          :CALL      JSR      PC,CLRQUE
3779
3780          CLRQUE: SAVREG          :SAVE R0 - R5
3781          MOV      #QCNT,R2      :ZERO THE QUEUE COUNTS
3782          CLR      (R2)+         :DRIVES 0 & 1
3783          CLR      (R2)+         :DRIVES 2 & 3
3784          CLR      (R2)+         :DRIVES 4 & 5
3785          CLR      (R2)+         :DRIVES 6 & 7
3786          MOV      #8,R3        :MOVE THE STARTING
3787          MOV      #QSTART,R1    :ADDRESS OF THE QUEUE INTO
3788          1$: MOV      (R1)+,(R2)+ :THE QUEUE INPUT POINTER
3789          DEC      R3
3790          BNE     1$
3791          MOV      #8,R3        :MOVE THE STARTING ADDRESS
3792          MOV      #QSTART,R1    :OF THE QUEUE INTO THE
3793          2$: MOV      (R1)+,(R2)+ :QUEUE OUTPUT POINTER
3794          DEC      R3
3795          BNE     2$
3796          RESREG          :RESTORE R0 - R5
3797          RTS      PC
3798
3799          :EMPTY THE QUEUE SPECIFIED BY R1:
3800          :CALL
3801          :CALL      MOV      DRVNUM,R1      :DRIVE NUMBER TO R1
3802          :CALL      JSR      PC,EMPTYQ
3803
3804          EMPTYQ: CLRB      QCNT(R1)      :CLEAR NUMBER OF ITEMS IN QUEUE
3805          ASL      R1
3806          MOV      QINPT(R1),QOUTPT(R1) :SET OUTPUT QUEUE POINTER=INPUT POINTER
3807          ASR      R1
3808          RTS      PC
3809
3810          :ROUTINE TO PUT A REQUEST IN QUEUE
3811          :CALL
3812          :CALL      MOV      #DRVNUM,R1    :DRIVE NUMBER
3813          :CALL      MOV      #DPB,R2      :ADDRESS OF PARAMETER BLOCK
3814          :CALL      JSR      R0,DRVQUE    :GO PUT REQUEST IN QUEUE
3815
3816
3817
3818

```

```

3819          : RETURN1          ;RETURN HERE IF QUEUE IS FULL
3820          : RETURN2          ;RETURN HERE IF REQUEST IS IN QUEUE
3821
3822 020346 122761 000010 017756 DRVQUE: CMPB #10,QCNT(R1) ;IS QUEUE FULL?
3823 020354 001421          BEQ 2$          ;BR IF YES-TAKE RETURN1
3824 020356 105261 017756          INCB QCNT(R1)    ;INCREMENT QUEUE COUNT
3825 020362 006301          ASL R1
3826 020364 010271 017766          MOV R2,QINPT(R1) ;PUT THIS REQUEST IN QUEUE
3827 020370 062761 000002 017766          ADD #2,QINPT(R1) ;UPDATE THE QUEUE POINTER
3828 020376 026161 017766 020030          CMP QINPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER
3829 020404 001003          BNE 1$          ;BRANCH IF NO
3830 020406 016161 020026 017766          MOV QSTART(R1),QINPT(R1) ;YES--RESET POINTER
3831 020414 006201          1$: ASR R1
3832 020416 005720          TST (R0)+      ;TAKE RETURN 2
3833 020420 000200          2$: RTS R0    ;RETURN TO USER
3834
3835          ;ROUTINE TO GET THE "DPB" ADDRESS OF NEXT REQUEST IN QUEUE
3836          :CALL
3837          :
3838          :   MOV #DRVNUM,R1      ;DRIVE NUMBER TO R1
3839          :   JSR PC,GETREQ      ;GO GET THE REQUEST
3840          :   :
3841          :   RETURN           ;R2="DPB" ADDRESS OF THE REQUEST
3842          :                   ;R2=0 IF NO REQUEST IN QUEUE
3843
3843 020422 005002          GETREQ: CLR R2
3844 020424 105761 017756          TSTB QCNT(R1)  ;IS THERE ANY REQUEST IN QUEUE?
3845 020430 001404          BEQ 2$          ;NO---BRANCH
3846 020432 006301          1$: ASL R1
3847 020434 017102 020006          MOV #QOUTPT(R1),R2 ;PICKUP "DPB" POINTER FOR THIS DRIVE
3848 020440 006201          ASR R1
3849 020442 000207          2$: RTS PC    ;RETURN TO USER
3850
3851          ;ROUTINE TO "POP" THE REQUEST FROM QUEUE
3852          :CALL
3853          :
3854          :   MOV #DRVNUM,R1      ;DRIVE NUMBER TO R1
3855          :   JSR PC,POPQUE      ;CALL TO REMOVE REQUEST
3856          :   :
3857          :   RETURN           ;R2=ADDRESS OF DPB REMOVED
3858
3858 020444 105361 017756          POPQUE: DECB QCNT(R1) ;DECREMENT QUEUE COUNT
3859 020450 006301          ASL R1
3860 020452 017102 020006          MOV #QOUTPT(R1),R2 ;GET THE "DPB" POINTER
3861 020456 062761 000002 020006          ADD #2,QOUTPT(R1) ;UPDATE THE QUEUE POINTER
3862 020464 026161 020006 020030          CMP QOUTPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER?
3863 020472 001003          BNE 1$          ;NO--BRANCH TO EXIT
3864 020474 016161 020026 020006          MOV QSTART(R1),QOUTPT(R1) ;YES--RESET THE POINTER
3865 020502 006201          1$: ASR R1
3866 020504 000207          RTS PC        ;RETURN TO USER
3867
3868          ;ROUTINE TO SAVE THE CONTENTS OF 'SESCAPE' WHEN THE DRIVER
3869          ;REPORTS AN ERROR DIRECTLY.
3870          :CALL
3871          :
3872          :   JSR R0,ES.SAV
    
```



```

3873      :      ERROR  N      ;:THE ERROR CALL
3874      :      RETURN      ;:THE RETURN IS PAST THE ERROR CALL
3875
3876 020506 012037 020522 ES.SAV: MOV (R0)+,1$ ;GET THE ERROR CALL
3877 020512 013746 001160      MOV $ESCAPE,-(SP) ;SAVE THE ADDRESS IN '$ESCAPE'
3878 020516 005037 001160      CLR $ESCAPE ;CLEAR THE ESCAPE RETURN
3879 020522 000000      IS: .WORD 0 ;THE ERROR CALL IS MOVED HERE
3880 020524 012637 001160      MOV (SP)+,$ESCAPE ;RESTORE THE ESCAPE ADDRESS
3881 020530 000200      RTS R0 ;RETURN

```

;;*****

.SBTTL TELETYPE MESSAGES

```

3888 020532 047440 043106 044514 UNTOFF: .ASCIZ / OFFLINE/
3889 020540 042516 000      UNTON: .ASCIZ / ONLINE/
3890 020543 040 047117 044514
3891 020550 042516 000      NOTRP: .ASCIZ a NOT AN RPO4/5/6a
3892 020553 040 047516 020124
3893 020560 047101 051040 030120
3894 020566 027464 027465 000066
3895 020574 047040 052117 050040 NOTPRS: .ASCIZ / NOT PRESENT/
3896 020602 042522 042523 052116
3897 020610 000
3898 020611 040 047125 040523 NOTSAF: .ASCIZ / UNSAFE/
3899 020616 042506 000
3900 020621 122 030120 000064 RPO4B: .ASCIZ /RPO4/
3901 020626 050122 032460 000 RPO5: .ASCIZ /RPO5/
3902 020633 122 030120 000066 RPO6: .ASCIZ /RPO6/
3903 020640 047125 052111 051440 SYSTAT: .ASCIZ /UNIT STATUS/<CR><LF><LF>
3904 020646 040524 052524 006523
3905 020654 005012 000
3906 020657 015 042012 044522 MUNIT: .ASCIZ <CR><LF>/DRIVE: /
3907 020664 042526 020072 000
3908 020671 040 020057 000 SLASH: .ASCIZ a / a
3909 020675 103 000 C: .ASCIZ /C/
3910 020677 124 000 T: .ASCIZ /T/
3911 020701 060 000 MDRVD: .ASCIZ /D/
3912 020703 040 040 LIN4SP: .ASCIZ / /
3913 020705 040 000040 LINSR: .ASCIZ / /
3914 020710 047105 042524 020122 ENTADR: .ASCIZ /ENTER ADDRESS LIMITS:/<CR><LF>
3915 020716 042101 051104 051505
3916 020724 020123 044514 044515
3917 020732 051524 006472 000012
3918 020740 020040 051104 053111 MOFFLN: .ASCIZ / DRIVE OFFLINE/<CR><LF>
3919 020746 020105 043117 046106
3920 020754 047111 006505 000012
3921 020762 042040 044522 042526 MDRNP: .ASCIZ / DRIVE NOT PRESENT/<CR><LF>
3922 020770 047040 052117 050040
3923 020776 042522 042523 052116
3924 021004 005015 000
3925 021007 040 051104 053111 MER11: .ASCIZ / DRIVE NOT AVAILABLE/<CR><LF>
3926 021014 020105 047516 020124

```

3927	021022	053101	044501	040514	
3928	021030	046102	006505	000012	
3929	021036	042040	044522	042526	MNRPO4: .ASCIZ 3 DRIVE NOT AN RPO4/5/63<CR><LF>
3930	021044	047040	052117	040440	
3931	021052	020116	050122	032060	
3932	021060	032457	033057	005015	
3933	021066	000			
3934	021067	040	051104	053111	MUSDR: .ASCIZ / DRIVE UNSAFE/<CR><LF>
3935	021074	020105	047125	040523	
3936	021102	042506	005015	000	
3937	021107	040	042523	042514	MSELD: .ASCIZ / SELECTED/
3938	021114	052103	042105	000	
3939	021121	015	050012	047522	MMODE: .ASCIZ <CR><LF>/PROGRAM MODE (C OR F): /
3940	021126	051107	046501	046440	
3941	021134	042117	020105	041450	
3942	021142	047440	020122	024506	
3943	021150	020072	000		
3944	021153	040	047506	046522	MFORMAT: .ASCIZ / FORMAT & VERIFY/
3945	021160	052101	023040	053040	
3946	021166	051105	043111	000131	
3947	021174	044103	041505	020113	MHECK: .ASCIZ /CHECK ONLY/
3948	021202	047117	054514	000	
3949	021207	106	051117	040515	MORMAT: .ASCIZ /FORMAT & VERIFY/
3950	021214	020124	020046	042526	
3951	021222	044522	054506	000	
3952	021227	015	005012	050117	MSIZE: .ASCIZ <CR><LF><LF>/OPERATE IN 22 SECTOR MODE (Y OR N) ? /
3953	021234	051105	052101	020105	
3954	021242	047111	031040	020062	
3955	021250	042523	052103	051117	
3956	021256	046440	042117	020105	
3957	021264	054450	047440	020122	
3958	021272	024516	037440	000040	
3959	021300	050117	051105	052101	MSEC22: .ASCIZ /OPERATION WILL BE IN 22 SECTOR (16 BIT) MODE/<CR><LF>
3960	021306	047511	020116	044527	
3961	021314	046114	041040	020105	
3962	021322	047111	031040	020062	
3963	021330	042523	052103	051117	
3964	021336	024040	033061	041040	
3965	021344	052111	020051	047515	
3966	021352	042504	005015	000	
3967	021357	117	042520	040522	MSEC20: .ASCIZ /OPERATION WILL BE IN 20 SECTOR (18 BIT) MODE/<CR><LF>
3968	021364	044524	047117	053440	
3969	021372	046111	020114	042502	
3970	021400	044440	020116	030062	
3971	021406	051440	041505	047524	
3972	021414	020122	030450	020070	
3973	021422	044502	024524	046440	
3974	021430	042117	006505	000012	
3975	021436	047111	040526	044514	BADENT: .ASCIZ /INVALID ENTRY/<CR><LF>
3976	021444	020104	047105	051124	
3977	021452	006531	000012		
3978	021456	047105	044504	043516	MADRER: .ASCII /ENDING DSK ADRS MUST BE EQUAL TO OR GREATER/<CR><LF>
3979	021464	042040	045523	040440	
3980	021472	051104	020123	052515	

3981 021500 052123 041040 020105
3982 021506 050505 040525 020114
3983 021514 047524 047440 020122
3984 021522 051107 040505 042524
3985 021530 006522 012
3986 021533 124 040510 020116
3987 021540 052123 051101 044524
3988 021546 043516 040440 051104
3989 021554 006523 000012
3990 021560 042523 042514 052103
3991 021566 042040 052101 020101
3992 021574 040520 052124 051105
3993 021602 006516 012
3994 021605 040 030050 020051
3995 021612 042532 047522 051447
3996 021620 005015
3997 021622 024040 024461 047440
3998 021630 042516 051447 005015
3999 021636 024040 024462 053440
4000 021644 051117 052123 041440
4001 021652 051501 035105 000040
4002
4003 021660 047527 051522 020124
4004 021666 040503 042523 000
4005 021673 015 005012 052123
4006 021700 051101 044524 043516
4007 021706 043040 051117 040515
4008 021714 020124 047117 042040
4009 021722 044522 042526 000040
4010 021730 005015 051412 040524
4011 021736 052122 047111 020107
4012 021744 044103 041505 020113
4013 021752 047117 042040 044522
4014 021760 042526 000040
4015 021764 005015 043012 051117
4016 021772 040515 020124 047503
4017 022000 050115 042514 042524
4018 022006 020054 000
4019 022011 015 005012 044103
4020 022016 041505 020113 047503
4021 022024 050115 042514 042524
4022 022032 020054 000
4023 022035 124 052117 046101
4024 022042 042440 051122 051117
4025 022050 020123 042504 042524
4026 022056 052103 042105 020072
4027 022064 000
4028 022065 120 042522 042523
4029 022072 052116 040440 042104
4030 022100 042522 051523 044440
4031 022106 035123 000040
4032
4033
4034

.ASCIZ /THAN STARTING ADRS/<CR><LF>
MSELP: .ASCII /SELECT DATA PATTERN/<CR><LF>
.ASCII / (0) ZERO'S/<CR><LF>
.ASCII / (1) ONE'S/<CR><LF>
.ASCIZ / (2) WORST CASE: /
MPATD: .ASCIZ /WORST CASE/
MSFOU: .ASCIZ <CR><LF><LF>/STARTING FORMAT ON DRIVE /
MSCHK: .ASCIZ <CR><LF><LF>/STARTING CHECK ON DRIVE /
MFCMPT: .ASCIZ <CR><LF><LF>/FORMAT COMPLETE, /
MCCMPT: .ASCIZ <CR><LF><LF>/CHECK COMPLETE, /
NUMERR: .ASCIZ /TOTAL ERRORS DETECTED: /
ADDRIS: .ASCIZ /PRESENT ADDRESS IS: /

::*****

4035
4036
4037
4038
4039 022112 044122 030461 044440
4040 022120 052116 051105 052522
4041 022126 052120 047440 041503
4042 022134 051125 042522 020104
4043 022142 051050 040520 036523
4044 022150 024460 000
4045 022153 125 042516 050130
4046 022160 041505 042524 020104
4047 022166 052101 042524 052116
4048 022174 047511 020116 041517
4049 022202 052503 051122 042105
4050 022210 000
4051 022211 115 051501 041123
4052 022216 051525 050040 051101
4053 022224 052111 020131 051105
4054 022232 047522 020122 046450
4055 022240 050103 036505 024461
4056 022246 000
4057 022247 115 051501 041123
4058 022254 051525 050040 051101
4059 022262 052111 020131 051105
4060 022270 047522 020122 050050
4061 022276 051101 030475 000051
4062 022304 042101 051104 051505
4063 022312 020123 046120 043525
4064 022320 041440 040510 043516
4065 022326 020105 044502 020124
4066 022334 042523 000124
4067 022340 044122 030461 042040
4068 022346 042111 023516 020124
4069 022354 042522 050123 047117
4070 022362 020104 047524 040440
4071 022370 042104 042522 051523
4072 022376 047111 000107
4073 022402 051104 053111 020105
4074 022410 043117 046106 047111
4075 022416 000105
4076 022420 042520 051522 051511
4077 022426 042524 052116 042040
4078 022434 044522 042526 052440
4079 022442 051516 043101 020105
4080 022450 051105 047522 000122
4081 022456 047125 047503 051122
4082 022464 041505 040524 046102
4083 022472 020105 040515 051523
4084 022500 052502 020123 040520
4085 022506 044522 054524 042440
4086 022514 051122 051117 000
4087 022521 123 043117 053524
4088 022526 051101 020105 044524

.SBTTL ERROR MESSAGES

;*****

EM1: .ASCIZ /RH11 INTERRUPT OCCURRED (RPAS=0)/

EM2: .ASCIZ /UNEXPECTED ATTENTION OCCURRED/

EM3: .ASCIZ /MASSBUS PARITY ERROR (MCPE=1)/

EM4: .ASCIZ /MASSBUS PARITY ERROR (PAR=1)/

EM5: .ASCIZ /ADDRESS PLUG CHANGE BIT SET/

EM6: .ASCIZ /RH11 DIDN'T RESPOND TO ADDRESSING/

EM10: .ASCIZ /DRIVE OFFLINE/

EM11: .ASCIZ /PERSISTENT DRIVE UNSAFE ERROR/

EM12: .ASCIZ /UNCORRECTABLE MASSBUS PARITY ERROR/

EM13: .ASCIZ /SOFTWARE TIMEOUT/

4089	022534	042515	052517	000124		
4090	022542	051104	053111	020105	EM14:	.ASCIZ /DRIVE UNSAFE ERROR/
4091	022550	047125	040523	042506		
4092	022556	042440	051122	051117		
4093	022564	000				
4094	022565	103	047117	051124	EM15:	.ASCIZ @CONTROLLER/DRIVE ERROR DURING WRITE@
4095	022572	046117	042514	027522		
4096	022600	051104	053111	020105		
4097	022606	051105	047522	020122		
4098	022614	052504	044522	043516		
4099	022622	053440	044522	042524		
4100	022630	000				
4101	022631	103	047117	051124	EM16:	.ASCIZ @CONTROLLER/DRIVE ERROR DURING WRITE CHECK@
4102	022636	046117	042514	027522		
4103	022644	051104	053111	020105		
4104	022652	051105	047522	020122		
4105	022660	052504	044522	043516		
4106	022666	053440	044522	042524		
4107	022674	041440	042510	045503		
4108	022702	000				
4109	022703	122	052105	044522	EM17:	.ASCIZ @RETRIES NOT SUCCESSFUL - SECTOR NOT ACCEPTABLE@
4110	022710	051505	047040	052117		
4111	022716	051440	041525	051505		
4112	022724	043123	046125	026440		
4113	022732	051440	041505	047524		
4114	022740	020122	047516	020124		
4115	022746	041501	042503	052120		
4116	022754	041101	042514	000		
4117	022761	104	052101	020101	EM20:	.ASCIZ @DATA ERROR DURING WRITE CHECK@
4118	022766	051105	047522	020122		
4119	022774	052504	044522	043516		
4120	023002	053440	044522	042524		
4121	023010	041440	042510	045503		
4122	023016	000				
4123	023017	103	047117	051124	EM21:	.ASCIZ @CONTROLLER/DRIVE ERROR VERIFYING HEADERS@
4124	023024	046117	042514	027522		
4125	023032	051104	053111	020105		
4126	023040	051105	047522	020122		
4127	023046	042526	044522	054506		
4128	023054	047111	020107	042510		
4129	023062	042101	051105	000123		
4130	023070	042510	042101	051105	EM22:	.ASCIZ @HEADER COMPARE ERROR VERIFYING HEADERS@
4131	023076	041440	046517	040520		
4132	023104	042522	042440	051122		
4133	023112	051117	053040	051105		
4134	023120	043111	044531	043516		
4135	023126	044040	040505	042504		
4136	023134	051522	000			
4137	023137	103	046131	047111	EM23:	.ASCIZ @CYLINDER FIELD IN HEADER IS NOT CORRECT@
4138	023144	042504	020122	044506		
4139	023152	046105	020104	047111		
4140	023160	044040	040505	042504		
4141	023166	020122	051511	047040		
4142	023174	052117	041440	051117		

4143	023202	042522	052103	000
4144	023207	127	044522	042524
4145	023214	041440	042510	045503
4146	023222	042440	051122	051117
4147	023230	000		
4148				
4149				
4150				
4151	023231	122	040520	000123
4152	023236	051104	053111	020105
4153	023244	020040	050122	051504
4154	023252	020061	020040	050122
4155	023260	051105	020061	020040
4156	023266	050122	051105	020062
4157	023274	020040	050122	051105
4158	023302	020063	020040	050122
4159	023310	051501	000	
4160	023313	104	044522	042526
4161	023320	020040	051040	043505
4162	023326	040440	051104	020040
4163	023334	040504	040524	000
4164	023341	104	044522	042526
4165	023346	020040	051040	043505
4166	023354	040440	051104	020040
4167	023362	020040	047507	042117
4168	023370	020040	020040	040502
4169	023376	000104		
4170	023400	044122	040440	042104
4171	023406	042522	051523	000
4172	023413	104	044522	042526
4173	023420	020040	042440	051122
4174	023426	050040	020103	051040
4175	023434	041520	030523	020040
4176	023442	051040	041520	031123
4177	023450	020040	051040	042120
4178	023456	030523	020040	051040
4179	023464	042520	030522	020040
4180	023472	051040	042520	031122
4181	023500	020040	051040	042520
4182	023506	031522	000	
4183	023511	122	042520	030503
4184	023516	020040	051040	042520
4185	023524	031103	020040	051040
4186	023532	053520	020103	020040
4187	023540	051040	041120	020101
4188	023546	020040	051040	042120
4189	023554	020101	020040	051040
4190	023562	040520	020123	020040
4191	023570	051040	046120	000101
4192	023576	050122	041104	020040
4193	023604	020040	050122	051115
4194	023612	020040	020040	050122
4195	023620	052104	020040	020040
4196	023626	050122	047123	020040

EM24: .ASCIZ SWRITE CHECK ERROR3

::*****

DH1: .ASCIZ /RPAS/
DH2: .ASCIZ /DRIVE RPOS1 RPER1 RPER2 RPER3 RPAS/

DH3: .ASCIZ /DRIVE REG ADR DATA/

DH4: .ASCIZ /DRIVE REG ADR GOOD BAD/

DH6: .ASCIZ /RH ADDRESS/

DH10: .ASCIZ /DRIVE ERR PC RPOS1 RPOS2 RPOS1 RPER1 RPER2 RPER3/

DH10A: .ASCIZ /RPEC1 RPEC2 RPWC RPSA RPSA RPAS RPLA/

DH10B: .ASCIZ /RPOB RPMR RPDT RPSN RPOF RPOA RPCC/

4251	024310	040440	052103	040525
4252	024316	020114	042510	042101
4253	024324	051105	000	
4254	024327	040	020040	020040
4255	024334	020040	020040	020040
4256	024342	020040	020040	020040
4257	024350	020040	020040	020040
4258	024356	020040	020040	020040
4259	024364	020040	020040	020040
4260	024372	020040	020040	046440
4261	024400	046505	051117	020131
4262	024406	042040	051511	006513
4263	024414	012		
4264	024415	104	044522	042526
4265	024422	020040	042440	051122
4266	024430	050040	020103	041440
4267	024436	046131	042116	020122
4268	024444	052040	040522	045503
4269	024452	020040	051440	041505
4270	024460	047524	020122	042040
4271	024466	052101	020101	020040
4272	024474	042040	052101	000101
4273				
4274				
4275	024502	001276		
4276	024504	001274	012100	012102
4277	024512	012104	012106	001276
4278	024520	001274	017240	017242
4279	024526	001274	017464	017462
4280	024534	017242		
4281	024536	001172		
4282	024540	001214	001116	001304
4283	024546	001314	001316	001320
4284	024554	001344	001346	
4285	024560	001350	001352	001306
4286	024566	001310	001312	001322
4287	024574	001324		
4288	024576	001326	001330	001332
4289	024604	001334	001336	001340
4290	024612	001342		
4291	024614	001214	001116	001270
4292	024622	001272	001252	
4293	024626	001214	001116	001270
4294	024634	001272	001252	
4295	024640	001304	001314	001316
4296	024646	001320	001344	001346
4297	024654	001350	001352	
4298	024660	001306	001310	001312
4299	024666	001322	001324	001326
4300	024674	001330	001332	
4301	024700	001334	001336	001340
4302	024706	001342		
4303	024710	001214	001116	025130
4304	024716	025120	025122	025124

DH24: .ASCII MEMORY DISK<<CR><LF>

.ASCIZ /DRIVE /ERR PC CYLNR TRACK SECTOR DATA DATA/

DT1: .WORD ATTN
DT2: .WORD DDRIVE, RPERRS, RPERRS+2, RPERRS+4, RPERRS+6, ATTN
DT3: .WORD DDRIVE, RD, ADR, RD, WRD
DT4: .WORD DDRIVE, WRT, AD, WRT, WD, RD, WRD
DT6: .WORD \$RPADR
DT10: .WORD DRIVE, \$ERRPC, RP, REG, RP, REG+10, RP, REG+12, RP, REG+14, RP, REG+40, RP, REG+42
.WORD RP, REG+44, RP, REG+46, RP, REG+2, RP, REG+4, RP, REG+6, RP, REG+16, RP, REG+20
.WORD RP, REG+22, RP, REG+24, RP, REG+26, RP, REG+30, RP, REG+32, RP, REG+34, RP, REG+36
DT17: .WORD DRIVE, \$ERRPC, DS, CYL, DS, TRK, SAVSEC
DT20: .WORD DRIVE, \$ERRPC, DS, CYL, DS, TRK, SAVSEC
.WORD RP, REG, RP, REG+10, RP, REG+12, RP, REG+14, RP, REG+40, RP, REG+42, RP, REG+44, RP, RE
.WORD RP, REG+2, RP, REG+4, RP, REG+6, RP, REG+16, RP, REG+20, RP, REG+22, RP, REG+24, RP, RE
DT23: .WORD DRIVE, \$ERRPC, BUFP, RBUF, RBUF+2, RBUF+4, RBUF+6

E08

4305	024724	025126					
4306	024726	001214	001116	001270	DT24:	.WORD	*DRIVE,\$ERRPC,DS.CYL,DS.TRK,SAVSEC,\$GDDAT,RP.REG+22
4307	024734	001272	001252	001124			
4308	024742	001326					
4309	024744	001304	001314	001316		.WORD	RP.REG,RP.REG+10,RP.REG+12,RP.REG+14,RP.REG+40,RP.REG+42,RP.REG+44,RP.RE
4310	024752	001320	001344	001346			
4311	024750	001350	001352				
4312	024764	001306	001310	001312		.WORD	RP.REG+2,RP.REG+4,RP.REG+6,RP.REG+16,RP.REG+20,RP.REG+22,RP.REG+24,RP.RE
4313	024772	001322	001324	001326			
4314	025000	001330	001332				
4315	025004	001334	001336	001340		.WORD	RP.REG+30,RP.REG+32,RP.REG+34,RP.REG+36
4316	025012	001342					
4317							
4318	025014	000001			DF1:	.WORD	1
4319	025016	001	000			.BYTE	1,0
4320	025020	000001			DF2:	.WORD	1
4321	025022	006	000			.BYTE	6,0
4322	025024	000001			DF3:	.WORD	1
4323	025026	003	000			.BYTE	3,0
4324	025030	000001			DF4:	.WORD	1
4325	025032	004	000			.BYTE	4,0
4326	025034	000003			DF10:	.WORD	3
4327	025036	010	000			.BYTE	8,0
4328	025040	023511				.WORD	DH10A
4329	025042	007	000			.BYTE	7,0
4330	025044	023576				.WORD	DH10B
4331	025046	007	000			.BYTE	7,0
4332	025050	000001			DF17:	.WORD	1
4333	025052	005	034			.BYTE	5,34
4334	025054	000004			DF20:	.WORD	4
4335	025056	005	034			.BYTE	5,34
4336	025060	024001				.WORD	DH20A
4337	025062	010	000			.BYTE	8,0
4338	025064	024077				.WORD	DH20B
4339	025066	010	000			.BYTE	8,0
4340	025070	024174				.WORD	DH20C
4341	025072	004	000			.BYTE	4,0
4342	025074	000001			DF23:	.WORD	1
4343	025076	007	000			.BYTE	7,0
4344	025100	000004			DF24:	.WORD	4
4345	025102	007	034			.BYTE	7,34
4346	025104	024001				.WORD	DH20A
4347	025106	010	000			.BYTE	8,0
4348	025110	024077				.WORD	DH20B
4349	025112	010	000			.BYTE	8,0
4350	025114	024174				.WORD	DH20C
4351	025116	004	000			.BYTE	4,0

:BUFFER STARTS HERE

4352
4353
4354
4355
4356
4357
4358

4359	025120	000000	000000	000000
4360	025126	000000		
4361				
4362	025130			
4363				
4364	025130	005015	046412	044501
4365	025136	042116	041505	030455
4366	025144	026461	055104	045122
4367	025152	026502	006501	012
4368	025157	122	030120	027464
4369	025164	027465	020066	047506
4370	025172	046522	052101	042524
4371	025200	020122	051120	043517
4372	025206	040522	006515	005012
4373	025214	000		
4374	025215	015	052012	020117
4375	025222	043047	051117	040515
4376	025230	023524	047440	020122
4377	025236	041447	042510	045503
4378	025244	020047	051104	053111
4379	025252	020105	026060	051040
4380	025260	050105	040514	042503
4381	025266	052040	042510	023440
4382	025274	054130	050104	006447
4383	025302	012		
4384	025303	120	041501	020113
4385	025310	047117	042040	044522
4386	025316	042526	030040	053440
4387	025324	052111	020110	047101
4388	025332	052117	042510	020122
4389	025340	040520	045503	020054
4390	025346	046103	040505	020122
4391	025354	042515	047515	054522
4392	025362	046040	041517	052101
4393	025370	047511	020116	030064
4394	025376	006454	012	
4395	025401	101	042116	051040
4396	025406	051505	040524	052122
4397	025414	052040	042510	050040
4398	025422	047522	051107	046501
4399	025430	005015	000	

RBUF: .WORD 0,0,0,0 ;BUFFER FOR HEADER CHECK

BUF: ;FORMAT AND CHECK BUFFER STARTS HERE

TITLE: .ASCII <CR><LF><LF>/MAINDEC-11-DZRJB-A/<CR><LF>

.ASCIZ RPO4/5/6 FORMATTER PROGRAM<CR><LF><LF>

LOADRV: .ASCII <CR><LF>/TO 'FORMAT' OR 'CHECK' DRIVE D, REPLACE THE 'XXDP'<CR><LF>

.ASCII /PACK ON DRIVE D WITH ANOTHER PACK, CLEAR MEMORY LOCATION 40,<CR><LF>

.ASCIZ /AND RESTART THE PROGRAM/<CR><LF>

```

.SBTL BUSADR - GET BUS ADDRESS AND VECTOR ADDRESS FOR RH11
:THIS ROUTINE IS USED TO INSURE THE BUS ADDRESS
:OF THE RH11 IS SETUP FOR THE PROPER ADDRESS.
:IT WILL ALSO READ THE ADDRESS FROM THE TTY IF
:REQUIRED.
:NOTE: THIS ROUTINE DESTROYS R0-R4
:CALL
:
:       JSR     PC,BUSADR
:       RETURN

```

4400
4401
4402
4403
4404
4405
4406
4407
4408
4409
4410
4411
4412
4413
4414
4415
4416
4417
4418
4419
4420
4421
4422
4423
4424
4425
4426
4427
4428
4429
4430
4431
4432
4433
4434
4435
4436
4437
4438
4439
4440
4441
4442
4443
4444
4445
4446
4447
4448
4449
4450
4451
4452
4453
4454
4455
4456
4457
4458
4459
4460
4461
4462
4463
4464
4465
4466
4467
4468
4469
4470
4471
4472
4473
4474
4475
4476
4477
4478
4479
4480
4481
4482
4483
4484
4485
4486
4487
4488
4489
4490
4491
4492
4493
4494
4495
4496
4497
4498
4499
4500


```

4413 025434 005737 001302      BUSADR: TST      CHGADR      :INPUT FROM TTY REQUESTED?
4414 025440 001450              BEQ        7$          :NO--BRANCH
4415 025442 005037 001302      CLR        CHGADR      :YES--CLEAR THE REQUEST FLAG
4416 025446 012700 001172      1$: MOV      #SRPADR,R0 :FIRST ADDRESS
4417 025452 104401 025634      TYPE      MRPCS1      :"RPCS1="
4418 025456 012046              MOV      (R0)+,-(SP)  :PRESENT RPCS1 ADDRESS
4419 025460 104402              TYP0C     :TYPE IT
4420 025462 104401 020705      TYPE      ,LINSF      :2 SPACES
4421 025466 104411              RDLIN     :GET THE ENTRY
4422 025470 012601              MOV      (SP)+,R1     :ADDRESS OF ASCII TEXT
4423 025472 004537 025656      JSR      R5,CK.NUM    :CHECK THE NUMBER
4424 025476 025516              3$       :CARRIAGE RETURN ONLY ENTERED
4425 025500 025562              7$       :PERIOD ONLY ENTERED
4426 025502 025446              1$       :ILLEGAL INPUT
4427 025504 025512              2$       :TERMINATED WITH A CARRIAGE RETURN
4428 025506 025446              1$       :TERMINATED WITH A "."
4429 025510 025556              4$       :TERMINATED WITH A ":"
4430 025512 010260 177776      2$: MOV      R2,-2(R0)  :SAVE NEW RPCS1
4431 025516 104401 025645      3$: TYPE      ,MRHVEC   :"RHVEC="
4432 025522 012046              MOV      (R0)+,-(SP)  :PRESENT RH11 VECTOR ADDRESS ON THE STACK
4433 025524 104402              TYP0C     :TYPE IT
4434 025526 104401 020705      TYPE      ,LINSF      :2 SPACES
4435 025532 104411              RDLIN     :READ THE ENTRY
4436 025534 012601              MOV      (SP)+,R1     :ASCII TEXT ADDRESS
4437 025536 004537 025656      JSR      R5,CK.NUM    :CHECK THE NUMBER
4438 025542 025562              7$       :CARRIAGE RETURN ONLY ENTERED
4439 025544 025562              7$       :PERIOD ONLY ENTERED
4440 025546 025516              3$       :ILLEGAL INPUT
4441 025550 025556              4$       :TERMINATED WITH A CARRIAGE RETURN
4442 025552 025516              3$       :TERMINATED WITH A "."
4443 025554 025556              4$       :TERMINATED WITH A ":"
4444 025556 010260 177776      4$: MOV      R2,-2(R0)  :SAVE INPUT
4445 025562 013701 000004      7$: MOV      ERVVEC,R1  :SAVE THE ERROR VECTOR
4446 025566 012737 025622 000004      MOV      #8$,ERVVEC  :SETUP FOR TRAP
4447 025574 005777 153372      TST      JSRPADR      :CHECK FOR RH11
4448 025600 010137 000004      MOV      R1,ERVVEC    :RESTORE ERROR VECTOR
4449 025604 012700 001172      MOV      #SRPADR,R0  :FIRST ADDRESS OF NEW PARAMETERS
4450 025610 012701 012246      MOV      #RPADR,R1   :FIRST ADDRESS OF WHERE TO PUT THEM
4451 025614 012021              MOV      (R0)+,(R1)+ :BUS ADDRESS
4452 025616 012021              MOV      (R0)+,(R1)+ :VECTOR ADDRESS
4453 025620 000207              RTS      PC          :RETURN
4454 025622 010137 000004      8$: MOV      R1,ERVVEC :RESTORE ERROR VECTOR
4455 025626 022626              CMP      (SP)+,(SP)+ :CLEAN OFF THE STACK
4456 025630 104006              ERROR   6           :REPORT THE ERROR
4457 025632 000705              BR      1$          :ASK FOR BUS ADDRESS
4458
4459 025634 050122 051503 020061 MRPCS1: .ASCIZ 2RPCS1 = 2
4460 025642 020075 000          000
4461 025646 122 053110 041505 MRHVEC: .ASCIZ 2RHVEC = 2
4462 025652 036440 000040
4463
4464
4465
4466
.SBTL CK.NUM - CHECK NUMBER (OCTAL)
;THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL CHARACTERS
;AND FORMS AN OCTAL NUMBER IN R2
    
```


ABS = 000200	ACK = 000123	ACL = 000040	ACTDRV 012154
ACTSTR 012165	ACU = 100000	ADDRIS 022065	ADRTBL 001476
AOE = 001000	ATA = 100000	ATABIT 012234	ATTN 001276
ATO = 000001	AT1 = 000002	AT2 = 000004	AT3 = 000010
AT4 = 000020	ATS = 000040	AT6 = 000100	AT7 = 000200
A16 = 000400	A17 = 001000	BADENT 021436	BAI = 000010
BEGCYL 001224	BEGIN = 002066	BEGIN1 002076	BEGIN2 002102
BEGTRK 001230	BIT0 = 000001	BIT00 = 000001	BIT01 = 000002
BIT02 = 000004	BIT03 = 000010	BIT04 = 000020	BIT05 = 000040
BIT06 = 000100	BIT07 = 000200	BIT08 = 000400	BIT09 = 001000
BIT1 = 000002	BIT10 = 002000	BIT11 = 004000	BIT12 = 010000
BIT13 = 020000	BIT14 = 040000	BIT15 = 100000	BIT2 = 000004
BIT3 = 000010	BIT4 = 000020	SIT5 = 000040	BIT6 = 000100
BIT7 = 000200	BIT8 = 000400	BIT9 = 001000	BPTVEC= 000014
BUFP 025130	BUSADR 025434	C 020675	CHGADR 001302
CI1 013606	CI3 013714	CI4 014022	CI5 014400
CI6 014422	CI7 014436	CI7B 014464	CI8 014536
CKADRS 003570	CKSWR = 104407	CKTRK 004236	CK.CHR 006426
CK.DEC 006400	CK.DIG 006500	CK.NUM 025656	CK.OCT 006352
CLOCK 006104	CLR = 000040	CLRQUE 020250	CNTLC 001300
CR = 000015	CRLF = 000200	CSF = 000002	CSU = 000010
CYLCK 001246	DCK = 100000	DCL = 000100	DCU = 000001
DDISP = 177570	DDRIVE 001274	DE1 = 000040	DF20 = 000002
DF1 025014	DF10 025034	DF17 025050	DF2 025020
DF20 025054	DF23 025074	DF24 025100	DF3 025024
DF4 025030	DH1 023231	DH10 023413	DH10A 023511
DH10B 023576	DH17 023663	DH2 023236	DH20 023732
DH20A 024001	DH20B 024077	DH20C 024174	DH23 024231
DH24 024327	DH3 023313	DH4 023341	DH6 023400
DIGB = 000004	DISPLA 001142	DISPRE 000174	DLT = 100000
DL64 = 000020	DMD = 000001	DONE 005210	DPINT = 012140
DPR = 000400	DPRQS 012150	DRIVE 001214	DRQ = 004000
DRVACT 012110	DRVCLR= 000111	DRVINT 012476	DRVQUE 020346
DRVSTA 012120	DRVTyp 012130	DRY = 000200	DSWR = 177570
DS.CYL 001270	DS.TRK 001272	DTE = 010000	DTSY = 000200
DTUW 012232	DT00 = 000001	DT01 = 000002	DT02 = 000004
DT03 = 000010	DT04 = 000020	DT05 = 000040	DT06 = 000100
DT07 = 000200	DT08 = 000400	DT1 024502	DT10 024540
DT17 024614	DT2 024504	DT20 024626	DT23 024710
DT24 024726	DT3 024520	DT4 024526	DT6 024536
DVA = 004000	ECH = 000100	ECI = 004000	EMPTYQ 020326
EMTVEC= 000030	EM1 022112	EM10 022402	EM11 022420
EM12 022456	EM13 022521	EM14 022542	EM15 022565
EM16 022631	EM17 022703	EM2 022153	EM20 022761
EM21 023017	EM22 023070	EM23 023137	EM24 023207
EM3 022211	EM4 022247	EM5 022304	EM6 022340
ENDCYL 001222	ENDTRK 001226	ENTADR 020710	ERINDX 005234
ERR = 040000	ERRVEC= 000004	ES.SAV 020506	EXT1 = 000001
EXT10 = 000010	EXT2 = 000002	EXT20 = 000020	EXT4 = 000004
EXT40 = 000040	FEN = 000200	FER = 000020	FMTDPB 001354
FMT22 = 010000	F1 = 000002	F2 = 000004	F3 = 000010
F4 = 000020	F5 = 000040	GETREG= 000141	GETREQ 020422
GC = 000001	GRV = 000010	GTSWR = 104406	HCE = 000200
HCI = 002000	HCRC = 000400	HDREAD 004650	HEDERR 001262

HT	=	000011	IAE	=	002000	IE	=	000100	ILF	=	000001
ILR	=	000002	IOTVEC	=	000020	IR	=	000100	ISR	=	015130
IXE	=	004000	LA	=	014772	LACNT	=	012176	LF	=	000012
LINSP	=	020705	LIN4SP	=	020703	LOADRV	=	025215	LOP.CK	=	005334
LST	=	002000	MADRER	=	021456	MAXSEC	=	001266	MCCMPT	=	022011
MCLK	=	000002	MCPE	=	020000	MCPEMX	=	012244	MDRNP	=	020762
MDRVD	=	020701	MER11	=	021007	MFCMPT	=	021764	MFORMT	=	021153
MHECK	=	021174	MHS	=	001000	MINX	=	000004	MMODE	=	021121
MNDLTA	=	012260	MNRPO4	=	021036	MODE	=	001220	MOFFLN	=	020740
MOH	=	020000	MOL	=	010000	MORMAT	=	021207	MPATD	=	021660
MPE	=	000400	MRD	=	000020	MRHVEC	=	025645	MRPCS1	=	025634
MSCHK	=	021730	MSE	=	000020	MSEC20	=	021357	MSEC22	=	021300
MSELD	=	021107	MSELP	=	021560	MSFOU	=	021673	MSIZE	=	021227
MSTCK	=	000010	MUNIT	=	020657	MUSDR	=	021067	MWC	=	001260
MWR	=	000040	MXDLTA	=	012256	MXF	=	001000	MXLACT	=	012254
MXWIND	=	012262	MO	=	003116	M1	=	002644	M1A	=	002734
M1B	=	003062	M2	=	003404	M4	=	003444	M5	=	003526
NBA	=	100000	NED	=	010000	NEM	=	004000	NHS	=	002000
NOTPRS	=	020574	NOTRP	=	020553	NOTSAF	=	020611	NUMERR	=	022035
OCYL	=	100000	OENTER	=	006116	OFFSET	=	000115	OFREV	=	000200
OF100	=	000004	OF200	=	000010	OF25	=	000001	OF400	=	000020
OF50	=	000002	OF800	=	000040	OPE	=	020000	OPI	=	020000
OFT	=	013326	OR	=	000200	PAR	=	000010	PARENT	=	006236
PAR1	=	001426	PAR2	=	001441	PAR3	=	001454	PAR4	=	001465
PAT	=	000020	PATA	=	001242	PATB	=	001244	PATSEL	=	001240
PC	=	%000007	PGE	=	002000	PGM	=	001000	PIP	=	020000
PIRQ	=	177772	PIRQVE	=	000240	PLU	=	020000	POPQUE	=	020444
PRO	=	000000	PR1	=	000040	PR2	=	000100	PR3	=	000140
PR4	=	000200	PR5	=	000240	PR6	=	000300	PR7	=	000340
PS	=	177776	PSEL	=	002000	PSU	=	000001	PSW	=	177776
PWRVEC	=	000024	QCNT	=	017756	QDRV0	=	020050	QDRV1	=	020070
QDRV2	=	020110	QDRV3	=	020130	QDRV4	=	020150	QDRV5	=	020170
QDRV6	=	020210	QDRV7	=	020230	QINPT	=	017766	QOUTPT	=	020006
QSTART	=	020026	QSTOP	=	020030	QTERM	=	020250	RAW	=	000020
RBUF	=	025120	RDCHR	=	104410	RDDAT	=	000171	RDHD	=	000173
RDLIN	=	104411	RDY	=	000200	RD.ADR	=	017240	RD.RP	=	017214
RD.RP1	=	017236	RD.RP2	=	017362	RD.RP3	=	017366	RD.RP4	=	017372
RD.WRD	=	017242	READIN	=	000121	RECAL	=	000107	RELSE	=	000113
RESREG	=	104413	RESVEC	=	000010	RETRY	=	001250	RMR	=	000004
RNOP	=	000101	RPADR	=	012246	RPAS	=	000016	RPBA	=	000004
RPCA	=	000034	RPCC	=	000036	RPCS1	=	000000	RPCS2	=	000010
RPDA	=	000006	RPDB	=	000022	RPDS1	=	000012	RPDT	=	000026
RPEC1	=	000044	RPEC2	=	000046	RPERRS	=	012100	RPER1	=	000014
RPER2	=	000040	RPER3	=	000042	RPINIT	=	012264	RPLA	=	000020
RPMR	=	000024	RPOF	=	000032	RPSN	=	000030	RPTMR	=	016466
RPVEC	=	012250	RPWC	=	000002	RP.REG	=	001304	RPO4	=	013034
RPO4B	=	020621	RPOS	=	020626	RPO6	=	020633	RTC	=	000117
RO	=	%000000	R1	=	%000001	R2	=	%000002	R3	=	%000003
R4	=	%000004	R5	=	%000005	R6	=	%000006	R7	=	%000007
SAVEFG	=	012206	SAVREG	=	104412	SAVSEC	=	001252	SAVWC	=	001254
SC	=	015334	SCAWC	=	005554	SC1	=	000100	SC10	=	001000
SC11	=	016166	SC12	=	016256	SC13	=	016326	SC2	=	000200
SC20	=	002000	SC3	=	015404	SC4	=	015410	SC5	=	015422
SC6	=	015622	SC6A	=	015732	SC7	=	016060	SC8	=	016136

SEARCH= 000131	SEC20 001264	SEEK = 000105	SEEKFG 012210
SELDIV= 000145	SETFMT= 000143	SETHDR 0056C2	SETPAT 003700
SETTBL 005402	SETVEC 002410	SET.IE 017704	SKI = 040000
SLASH 020671	SOFSW 001216	SP =%000006	SRCHWT 012162
STACK = 001100	STCLK1 006020	STCLK2 006070	STCLK3 006074
STKLMT= 177774	ST0 016560	ST01 016610	ST02 017006
ST03 017056	ST05 017102	ST06 017110	ST07 017146
ST08 017176	ST09 017206	ST.CLK 005742	SVRH11 017566
SWR 001140	SWREG 000176	SW0 = 000001	SW00 = 000001
SW01 = 000002	SW02 = 000004	SW03 = 000010	SW04 = 000020
SW05 = 000040	SW06 = 000100	SW07 = 000200	SW08 = 000400
SW09 = 001000	SW1 = 000002	SW10 = 002000	SW11 = 004000
SW12 = 010000	SW13 = 020000	SW14 = 040000	SW15 = 100000
SW2 = 000004	SW3 = 000010	SW4 = 000020	SW5 = 000040
SW6 = 000100	SW7 = 000200	SW8 = 000400	SW9 = 001000
SYSTAT 020640	T 020677	TABLE 001374	TAP = 040000
TBITVE= 000014	TD 015174	TDF = 000040	TIMER 012212
TITLE 025130	TKVEC = 000060	TPVEC = 000064	TRAPVE= 000034
TRE = 040000	TRKCNT 001236	TRKTST 005472	TRK1 = 004000
TRK10 = 040000	TRK2 = 010000	TRK20 = 100000	TRK4 = 020000
TRNSWT 012160	TRTVEC= 000014	TTRKS 001232	TTRKSC 001234
TUF = 000100	TYPADR 006136	TYPDS = 104405	TYPE = 104401
TYPERR 007044	TYPOC = 104402	TYPON = 104404	TYPOS = 104403
ULDFLG 012166	UNLOAD= 000103	UNS = 040000	UNTOFF 020532
UNTON 020543	UPDACY 005662	UPDATK 005712	UPE = 020000
US1 = 000001	US2 = 000002	US4 = 000004	UWR = 000010
VUF = 000002	VU30 = 010000	VV = 000100	WAO = 000002
WC 001256	WCE = 040000	WCF = 000040	WCKD = 000151
WCKHD = 000153	WCTBL 001552	WCU = 000001	WLE = 004000
WRL = 004000	WRDAT= 000161	WRTHD = 000163	WRTRK 004004
WRTRK1 004060	WRTRK2 004116	WRT.AD 017464	WRT.RP 017374
WRT.R1 017460	WRT.R2 017550	WRT.R3 017556	WRT.R4 017562
WRT.R5 017564	WRT.WD 017462	WRU = 000400	WSU = 000004
SAUTOB 001134	\$BDADR 001122	\$BDAT 001126	\$BELL 001162
\$CHARC 007510	\$CKSWR 010472	\$CMTAG 001100	\$CM3 = 000000
\$CNTLC 011367	\$CNTLG 011401	\$CNTLU 011374	\$CRLF 001167
\$DBLK 010156	\$DB2D 011524	\$DECVL 011704	\$DOAGN 005204
\$DTBL 010146	\$ENDAD 005174	\$ENDCT 005160	\$EOP 005100
\$EOPCT 005152	\$ERFLG 001103	\$ERMAX 001115	\$ERROR 006636
\$ERRPC 001116	\$ERRTB 001626	\$ERTTL 001112	\$ESCAP 001160
\$FILLC 001156	\$FILLS 001155	\$GDADR 001120	\$GDDAT 001124
\$GET42 005164	\$GTSWR 010562	\$HD = 000000	\$ICNT 001104
\$INTAG 001135	\$ITEMB 001114	\$LF = 001170	\$LKCSB 001200
\$LKCSR 001176	\$LKS 001206	\$LLVEC 001210	\$LPADR 001106
\$LPERR 001110	\$LPVEC 001202	\$MNEW 011417	\$MSWR 011406
\$NULL 001154	\$OCNT 007736	\$OMODE 007740	\$PASS 001100
\$QUES 001166	\$RDCHR 011034	\$RDLIN 011124	\$RDSZ = 000007
\$RESRE 011756	\$RPADR 001172	\$RPVEC 001174	\$RTNAD 005206
\$SAVRE 011720	\$SB2D 011470	\$SETUP= 000106	\$STUP = 177777
\$SUPRS 011430	\$SVPC = 000200	\$SWR = 123000	\$TKB 001146
\$TKCNT 010166	\$TKINT 010204	\$TKQEN= 010203	\$TKQIN 010170
\$TKGOU 010172	\$TKQSR 010174	\$TKS 001144	\$TKSRV 010254
\$TN = 000000	\$TNPWR 011634	\$TPB 001152	\$TPFLG 001157
\$TPS 001150	\$TRAP 012014	\$TRAP2 012036	\$TRP = 000014

\$TRPAD 012050
\$TYPE 007274
\$TYPON 007554
= 026016

\$STNM 001102
\$TYPEC 007444
\$TYPOS 007514

\$ITYIN 011360
\$TYPEX 007512
\$\$GET4= 000000

\$TYPDS 007742
\$TYPC 007540
\$OFILL 007737

ERRORS DETECTED: 0

*DZRJBA,DZRJB/SOL/LI:ME/NL:MC:MD:CND=RPO456.010,DZRJB.013
RUN-TIME: 72 49 0 SECONDS
CORE USED: 35K

