

RP04/5/6

FORMATTER
CZRJBBO

AH-9187B-MC

JAN 1978

COPYRIGHT © 74-77

digital

FICHE 1 OF 1

MADE IN USA

This microfiche card contains a grid of frames. The frames on the left side of the card contain data, while the right side is mostly blank. The data in the frames is organized into columns and rows, with some frames containing what appears to be a header or title. The data is too small to read clearly but seems to be a structured list or table of information.

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

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.

198
199
000
001
002
003
004
005
006
007
008
009
010
011
012
013
014
015
016
017
018
019
020
021
022
023
024
025
026
027
028
029
030
031
032
033
034
035
036
037
038
039
040
041
042
043
044
045
046
047
048
049
050
051
052
053
054
055
056
057
058
059
060
061
062
063
064
065
066
067
068
069
070
071
072
073
074
075
076
077
078
079
080
081
082
083
084
085
086
087
088
089
090
091
092
093
094
095
096
097
098
099
100

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(8) 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.

25
24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0

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 RPO4/5/6 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.

010
011
012
013
014
015
016
017
018
019
020
021
022
023
024
025
026
027
028
029
030
031
032
033
034
035
036
037
038
039
040
041
042
043
044
045
046
047
048
049
050
051
052
053
054
055
056
057
058
059
060
061
062
063
064
065
066
067
068
069
070
071
072
073
074
075
076
077
078
079
080
081
082
083
084
085
086
087
088
089
090
091
092
093
094
095
096
097
098
099
100
101
102
103
104
105
106
107
108
109
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

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 SUCESSFUL - 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 EPROR OCCURRED DURING

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

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 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.

9. PROGRAM LISTING

```

-----
          2
.TITLE  CZRJ880, RPO4/5/6 FMTR
;*COPYRIGHT (C) 1976,1977
;*DIGITAL EQUIPMENT CORP.
;*MAYNARD, MASS. 01754
;*
;*PROGRAM BY C. HESS
;*
;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
;*PACKAGE (MAINDEC-11-DZQAC-C3). JAN 19, 1977.
;*
    
```

478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
000
001
002
003
004
005
006
007
008
009
010
011
012
013
014
015
016
017
018
019
020
021
022
023
024
025
026
027
028
029
030
031
032
033
034
035
036
037
038
039
040
041
042
043
044
045
046
047
048
049
050
051
052
053
054
055
056
057
058
059
060
061
062
063
064
065
066
067
068
069
070
071
072
073
074
075
076
077
078
079
080
081
082
083
084
085
086
087
088
089
090
091
092
093
094
095
096
097
098
099
100
101
102
103
104
105
106
107
108
109
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
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 OPERATIONAL SWITCH SETTINGS
;*
;* SWITCH USE
;* -----
;* 15 HALT ON ERROR
;* 13 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
.=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
DISPRG: .WORD 0 ;; SOFTWARE DISPLAY REGISTER
SWREG: .WORD 0 ;; SOFTWARE SWITCH REGISTER

.SBTTL ACT11 HOOKS
;*****
;HOOKS REQUIRED BY ACT11
$SVPC=.;SAVE PC
.=46
$ENDAD ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
.=52
.WORD 20000 ;;2)SET LOC.52 TO 20000
.=$SVPC ;; RESTORE PC

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

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

.SBTTL BASIC DEFINITIONS
;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
STACK= 1100
.EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL
.EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL

;*MISCELLANEOUS DEFINITIONS
HT= 11 ;;CODE FOR HORIZONTAL TAB
LF= 12 ;;CODE FOR LINE FEED
CR= 15 ;;CODE FOR CARRIAGE RETURN
CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED
PS= 177776 ;;PROCESSOR STATUS WORD
.EQUIV PS,PSW
STKLMT= 177774 ;;STACK LIMIT REGISTER

```

000000

000174 000000
000176 000000

000046 000200
000052 000052
000200 000200

000200 000137 002076

000204 000137 002066

001100

000011
000012
000015
000200
177776
177774

534	177772	PIRQ=	177772	:::	PROGRAM INTERRUPT REQUEST REGISTER
535	177570	DSWR=	177570	:::	HARDWARE SWITCH REGISTER
536	177570	DDISP=	177570	:::	HARDWARE DISPLAY REGISTER

537		:*GENERAL PURPOSE REGISTER DEFINITIONS			
538	000000	R0=	%0	:::	GENERAL REGISTER
539	000001	R1=	%1	:::	GENERAL REGISTER
540	000002	R2=	%2	:::	GENERAL REGISTER
541	000003	R3=	%3	:::	GENERAL REGISTER
542	000004	R4=	%4	:::	GENERAL REGISTER
543	000005	R5=	%5	:::	GENERAL REGISTER
544	000006	R6=	%6	:::	GENERAL REGISTER
545	000007	R7=	%7	:::	GENERAL REGISTER
546	000006	SP=	%6	:::	STACK POINTER
547	000007	PC=	%7	:::	PROGRAM COUNTER

548		:*PRIORITY LEVEL DEFINITIONS			
549	000000	PR0=	0	:::	PRIORITY LEVEL 0
550	000040	PR1=	40	:::	PRIORITY LEVEL 1
551	000100	PR2=	100	:::	PRIORITY LEVEL 2
552	000140	PR3=	140	:::	PRIORITY LEVEL 3
553	000200	PR4=	200	:::	PRIORITY LEVEL 4
554	000240	PR5=	240	:::	PRIORITY LEVEL 5
555	000300	PR6=	300	:::	PRIORITY LEVEL 6
556	000340	PR7=	340	:::	PRIORITY LEVEL 7

557		:*"SWITCH REGISTER" SWITCH DEFINITIONS			
558	100000	SW15=	100000		
559	040000	SW14=	40000		
560	020000	SW13=	20000		
561	010000	SW12=	10000		
562	004000	SW11=	4000		
563	002000	SW10=	2000		
564	001000	SW09=	1000		
565	000400	SW08=	400		
566	000200	SW07=	200		
567	000100	SW06=	100		
568	000040	SW05=	40		
569	000020	SW04=	20		
570	000010	SW03=	10		
571	000004	SW02=	4		
572	000002	SW01=	2		
573	000001	SW00=	1		
574		.EQUIV	SW09, SW9		
575		.EQUIV	SW08, SW8		
576		.EQUIV	SW07, SW7		
577		.EQUIV	SW06, SW6		
578		.EQUIV	SW05, SW5		
579		.EQUIV	SW04, SW4		
580		.EQUIV	SW03, SW3		
581		.EQUIV	SW02, SW2		
582		.EQUIV	SW01, SW1		
583		.EQUIV	SW00, SW0		

584		:*DATA BIT DEFINITIONS (BIT00 TO BIT15)			
585	100000	BIT15=	100000		

590 040000
591 020000
592 010000
593 004000
594 002000
595 001000
596 000400
597 000200
598 000100
599 000040
600 000020
601 000010
602 000004
603 000002
604 000001

616 000004
617 000010
618 000014
619 000014
620 000014
621 000014
622 000020
623 000024
624 000030
625 000034
626 000060
627 000064
628 000024

637 000100
638 000200
639 000400
640 001000
641 002000
642 020000
643 040000

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
ERRVEC= 4 ;: TIME OUT AND OTHER ERRORS
RESVEC= 10 ;: RESERVED AND ILLEGAL INSTRUCTIONS
TBITVEC= 14 ;: "T" BIT
TRIVEC= 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

;;*****

.SBTTL R411 REGISTERS

;;*****

;CONTROL AND STATUS REGISTER 1 (RPCS1)

IE= 100 ;: INTERRUPT ENABLE (BIT #6)
RDY= 200 ;: READY (BIT #7)
A16= 400 ;: HIGH ORDER BUS ADDRESS BIT (BIT #8)
A17= 1000 ;: HIGH ORDER BUS ADDRESS BIT (BIT #9)
PSEL= 2000 ;: PORT SELECT (BIT #10)
MCPPE= 20000 ;: MASSBUSS PARITY ERROR (BIT #13)
TPE= 40000 ;: TRANSFER ERROR (BIT #14)
;SC= 100000 ;: SPECIAL CONDITION (BIT #15)

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

000400
001000
002000
004000
010000
020000
040000
100000

DPR= 400
PGM= 1000
LST= 2000
WRL= 4000
MOL= 10000
PIP= 20000
EPR= 40000
ATA= 100000

: DRIVE PRESENT (BIT #8)
: PROGRAMABLE (BIT #9)
: LAST SECTOR TRANSFERRED (BIT #10)
: WRITE LOCK (BIT #11)
: MEDIUM ON-LINE (BIT #12)
: POSITIONING OPERATION IN PROGRESS (BIT #13)
: COMPOSITE ERROR (BIT #14)
: ATTENTION ACTIVE (BIT #15)

; ERROR REGISTER #01 (RPER!) (#02)

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

ILF= 1
ILR= 2
RMR= 4
PAR= 10
FER= 20
WCF= 40
ECH= 100
HCE= 200
HCRC= 400
ROE= 1000
IAE= 2000
WLE= 4000
OTE= 10000
OPI= 20000
UNS= 40000
DCK= 100000

: ILLEGAL FUNCTION (BIT #0)
: ILLEGAL REGISTER (BIT #1)
: REGISTER MODIFICATION REFUSED (BIT #2)
: PARITY ERROR (BIT #3)
: FORMAT ERROR (BIT #4)
: WRITE CLOCK FAIL (BIT #5)
: ECC HARD ERROR (BIT #6)
: HEADER COMPARE ERROR (BIT #7)
: HEADER CRC ERROR (BIT #8)
: ADDRESS OVERFLOW ERROR (BIT #9)
: INVALID ADDRESS ERROR (BIT #10)
: WRITE LOCK ERROR (BIT #11)
: DRIVE TIMING ERROR (BIT #12)
: OPERATION INCOMPLETE (BIT #13)
: DRIVE UNSAFE (BIT #14)
: DATA CHECK ERROR (BIT #15)

; MAINTAINABILITY REGISTER (RPMR) (#03)

000001
000002
000004
000010
000020
000040
000200

DMD= 1
MCLK= 2
MINX= 4
MSTCK= 10
MRD= 20
MWR= 40
DTSY= 200

: DIAGNOSTIC MODE (BIT #0)
: MAINTAINABILITY CLOCK (BIT #1)
: MAINTAINABILITY INDEX (BIT #2)
: MAINTAINABILITY SECTOR CLOCK (BIT #3)
: MAINTAINABILITY READ (BIT #4)
: MAINTAINABILITY WRITE (BIT #5)
: MAINTAINABILITY SYNC DETECTED (BIT #7)

; ATTENTION SUMMARY PSEUDO-REGISTER (RPAS) (#04)

000001
000002
000004
000010
000020
000040
000100
000200

AT0= 1
AT1= 2
AT2= 4
AT3= 10
AT4= 20
AT5= 40
AT6= 100
AT7= 200

: DEVICE 0 (BIT #0)
: DEVICE 1 (BIT #1)
: DEVICE 2 (BIT #2)
: DEVICE 3 (BIT #3)
: DEVICE 4 (BIT #4)
: DEVICE 5 (BIT #5)
: DEVICE 6 (BIT #6)
: DEVICE 7 (BIT #7)

; DESIRED SECTOR/TRACK ADDRESS REGISTER (RPDA) (#05)
; (EACH BIT IS CALLED BY BIT NUMBER)

; DRIVE TYPE REGISTER (RPDT) (#06)

000001
000002

DT00= 1
DT01= 2

: DRIVE TYPE NUMBER BIT 1
: DRIVE TYPE NUMBER BIT 2

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

000004
000010
000020
000040
000100
000200
000400
004000
020000
040000
100000

DT02= 4
DT03= 10
DT04= 20
DT05= 40
DT06= 100
DT07= 200
DT08= 400
DRQ= 4000
MOH= 20000
TAP= 40000
NBA= 100000

: DRIVE TYPE NUMBER BIT 3
: DRIVE TYPE NUMBER BIT 4
: DRIVE TYPE NUMBER BIT 5
: DRIVE TYPE NUMBER BIT 6
: DRIVE TYPE NUMBER BIT 7
: DRIVE TYPE NUMBER BIT 8
: DRIVE TYPE NUMBER BIT 9
: DRIVE REQUEST REQUIRED (BIT #11)
: MOVING HEAD (BIT #13)
: TAPE DRIVE (BIT #14)
: NOT BLOCK ADDRESSED (BIT #15)

:LOOK-AHEAD REGISTER (RPLA) (#07)

000001
000002
000004
000010
000020
000040
000100
000200

001000
002000
004000
010000
020000
040000
100000

EXT1= 1
EXT2= 2
EXT4= 4
EXT10= 10
EXT20= 20
EXT40= 40
SC1= 100
SC2= 200
SC4= 400
SC10= 1000
SC20= 2000
TRK1= 4000
TRK2= 10000
TRK4= 20000
TRK10= 40000
TRK20= 100000

: EXTENSION 1 (BIT #0)
: EXTENSION 2 (BIT #1)
: EXTENSION 3 (BIT #2)
: EXTENSION 4 (BIT #3)
: EXTENSION 5 (BIT #4)
: EXTENSION 6 (BIT #5)
: SECTOR COUNT FIELD 0 (BIT #6)
: SECTOR COUNT FIELD 1 (BIT #7)
: SECTOR COUNT FIELD 2 (BIT #8)
: SECTOR COUNT FIELD 3 (BIT #9)
: SECTOR COUNT FIELD 4 (BIT #10)
: TRACK FIELD 1 (BIT #11)
: TRACK FIELD 2 (BIT #12)
: TRACK FIELD 3 (BIT #13)
: TRACK FIELD 4 (BIT #14)
: TRACK FIELD 5 (BIT #15)

:RPO4 ERROR REGISTER #2 (RPER2) (#10)

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

WCU= 1
CSF= 2
WSU= 4
CSU= 10
MSE= 20
TDF= 40
TUF= 100
FEN= 200
WRU= 400
MHS= 1000
NHS= 2000
IXE= 4000
VU30= 10000
PLU= 20000
ACU= 100000

: WRITE CURRENT UNSAFE (BIT #0)
: CURRENT SINK FAILURE (BIT #1)
: WRITE SELECT UNSAFE (BIT #2)
: CURRENT SWITCH UNSAFE (BIT #3)
: MOTOR SEQUENCE ERROR (BIT #4)
: TRANSITIONS DETECTOR FAILURE (BIT #5)
: TRANSITIONS UNSAFE (BIT #6)
: FAILSAFE ENABLED (BIT #7)
: WRITE READY UNSAFE (BIT #8)
: MULTIPLE HEAD SELECT (BIT #9)
: NO HEAD SELECTION (BIT #10)
: INDEX ERROR (BIT #11)
: 30VOLT UNSAFE (BIT #12)
: PLO UNSAFE (BIT #13)
: AC UNSAFE (BIT #15)

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

000001
000002
000004
000010
000020

WCU= 1
CSF= 2
WSU= 4
CSU= 10
RAW= 20

: WRITE CURRENT UNSAFE (BIT #0)
: CURRENT SINK FAILURE (BIT #1)
: WRITE SELECT UNSAFE (BIT #2)
: CURRENT SWITCH UNSAFE (BIT #3)
: READ AND WRITE (BIT #4)

014
015
016
017
018
019
020
021
022
023
024
025
026
027
028
029
030
031
032
033
034
035
036
037
038
039
040
041
042
043
044
045
046
047
048
049
050
051
052
053
054
055
056
057
058
059
060
061
062
063
064
065
066
067
068
069
070
071
072
073
074
075
076
077
078
079
080
081
082
083
084
085
086
087
088
089
090
091
092
093
094
095
096
097
098
099
100
101
102
103
104
105
106
107
108
109
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

000040
000100
000200
000400
001000
002000
004000
020000

TDF= 40
TUF= 100
ABS= 200
WRU= 400
MHS= 1000
NHS= 2000
IXE= 4000
PLU= 20000

; TRANSITIONS DETECTOR FAILURE (BIT #5)
; TRANSITIONS UNSAFE (BIT #6)
; ABNORMAL STOP (BIT #7)
; WRITE READY UNSAFE (BIT #8)
; MULTIPLE HEAD SELECT (BIT #9)
; NO HEAD SELECTION (BIT #10)
; INDEX ERROR (BIT #11)
; PLO UNSAFE (BIT #12)

; OFFSET REGISTER (RPOF) (#11)

000001
000002
000004
000010
000020
000040
000200
002000
004000
010000

OF25= 1
OF50= 2
OF100= 4
OF200= 10
OF400= 20
OF800= 40
OFREV= 200
HCI= 2000
ECI= 4000
FMT22= 10000

; OFFSET 25 MICRO INCHES (BIT #0)
; OFFSET 50 MICRO INCHES (BIT #1)
; OFFSET 100 MICRO INCHES (BIT #2)
; OFFSET 200 MICRO INCHES (BIT #3)
; OFFSET 400 MICRO INCHES (BIT #4)
; OFFSET 800 MICRO INCHES (BIT #5)
; OFFSET NEGATIVE (REVERSE) (BIT #5)
; HEADER COMPARE INHIBIT (BIT #10)
; ERROR CORRECTION CODE INHIBIT (BIT #11)
; 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)

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

000001
000002
000010
000040
000100
040000
100000

PSU= 1
VUF= 2
UWR= 10
ACL= 40
DCL= 100
SKI= 40000
OCYL= 100000

; PACK SPEED UNSAFE (BIT #0)
; VELOCITY UNSAFE (BIT #1)
; ANY UNSAFE EXCEPT READ/WRITE (BIT #3)
; AC LOW (BIT #5)
; DC LOW (BIT #6)
; SEEK INCOMPLETE (BIT #14)
; OFF CYLINDER (BIT #15)

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

000001
000002
000040
000100
020000
040000
100000

DCU= 1
WAO= 2
ACL= 40
DCL= 100
OPE= 20000
SKI= 40000
OCYL= 100000

; DC UNSAFE (BIT #0)
; WRITE AND OFFSET (BIT #1)
; AC LOW (BIT #5)
; DC LOW (BIT #6)
; OPERATOR PLUG ERROR (BIT #13)
; SEEK INCOMPLETE (BIT #14)
; 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)

000000
000001
000002
000003
000004
000005
000006
000007
000008
000009
000010
000011
000012
000013
000014
000015
000016
000017
000018
000019
000020
000021
000022
000023
000024
000025
000026
000027
000028
000029
000030
000031
000032
000033
000034
000035
000036
000037
000038
000039
000040
000041
000042
000043
000044
000045
000046
000047
000048
000049
000050
000051
000052
000053
000054
000055
000056
000057
000058
000059
000060
000061
000062
000063
000064
000065
000066
000067
000068
000069
000070
000071
000072
000073
000074
000075
000076
000077
000078
000079
000080
000081
000082
000083
000084
000085
000086
000087
000088
000089
000090
000091
000092
000093
000094
000095
000096
000097
000098
000099
000100

.SBTTL RPO4/5/6 DRIVER COMMANDS

000101	RNOP	=	101	:NO OPERATION
000103	UNLOAD	=	103	:UNLOAD
000105	SEEK	=	105	:SEEK
000107	RECF	=	107	:RECALIBRATE
000111	DRVCLR	=	111	:DRIVE CLEAR
000113	RELSE	=	113	:RELEASE
000115	OFFSET	=	115	:OFFSET
000117	RTC	=	117	:RETURN TO CENTER LINE
000121	READIN	=	121	:READ IN PRESET
000123	ACK	=	123	:PACK ACKNOWLEDGE
000131	SEARCH	=	131	:SEARCH
000141	GETREG	=	141	:GET REGISTERS
000143	SETFMT	=	143	:SET FORMAT (& ECI OR HCI)
000145	SELDRV	=	145	:SELECT DRIVE
000151	WCKD	=	151	:WRITE CHECK DATA
000153	WCKHD	=	153	:WRITE CHECK HEADER & DATA
000161	WRDAT	=	161	:WRITE DATA
000163	WRTHD	=	163	:WRITE HEADER & DATA
000171	RDDAT	=	171	:READ DATA
000173	RDHD	=	173	:READ HEADER & DATA

```

899
900
901
902
903
904
905      001100
906      001100      000000
907      001100      000000
908      001102      000
909      001103      000
910      001104      000000
911      001106      000000
912      001110      000000
913      001112      000000
914      001114      000
915      001115      001
916      001116      000000
917      001120      000000
918      001122      000000
919      001124      000000
920      001126      000000
921      001130      000000
922      001132      000000
923      001134      000
924      001135      000
925      001136      000000
926      001140      177570
927      001142      177570
928      001144      177560
929      001146      177562
930      001150      177564
931      001152      177566
932      001154      000
933      001155      002
934      001156      012
935      001157      000
936      001160      000000
937      001162      177607      000377
938      001166      077
939      001167      015
940      001170      000012
941
942      000015
943      000012
944      001172      176700
945      001174      000254
946      001176      172540
947      001200      172542
948      001202      000104      000106
949      001206      177546
950      001210      000100      000102
951      001214      000000
952      001216      000000
953      001220      000000
954      001222      000632

```

.SBTTL COMMON TAGS

```

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

```

```

      . = 1100
SCMTAG:      . = 1100
SPASS:      .WORD      0
STSTNM:     .BYTE
SERFLG:     .BYTE
$ICNT:      .WORD
$LPADR:     .WORD
$LPERR:     .WORD
$ERTTL:     .WORD
$ITEMB:     .BYTE
$ERMAX:     .BYTE
$ERRPC:     .WORD
$GDADR:     .WORD
$BDADR:     .WORD
$GDDAT:     .WORD
$BDDAT:     .WORD
SAUTOB:     .BYTE
$INTAG:     .BYTE
SWR:        .WORD      DSWR
DISPLAY:    .WORD      DDISP
$TKS:       177560
$TKB:       177562
$TPS:       177564
$TPB:       177566
$NULL:      .BYTE      0
$FILLS:     .BYTE      2
$FILLC:     .BYTE      12
$TPFLG:     .BYTE      0
SE=APE:    0
$BELL:      .ASCIZ    <207><377><377>
$QUES:      .ASCII    '?'
$CRLF:      .ASCII    <15>
$LF:        .ASCIZ    <12>
*****
CR          =      15
LF          =      12
$RPADR:     .WORD      176700
$RPVEC:     .WORD      254
$LKCSR:     .WORD      172540
$LKCSB:     .WORD      172542
$LPVEC:     .WORD      104,106
$LKS:       .WORD      177546
$LLVEC:     .WORD      100,102
DRIVE:      .WORD      0
SOF5W:      .WORD      0
MODE:       .WORD      0
ENDCYL:     .WORD      410.

```

```

: START OF COMMON TAGS
: CONTAINS PASS COUNT
: CONTAINS THE TEST NUMBER
: CONTAINS ERROR FLAG
: CONTAINS SUBTEST ITERATION COUNT
: CONTAINS SCOPE LOOP ADDRESS
: CONTAINS SCOPE RETURN FOR ERRORS
: CONTAINS TOTAL ERRORS DETECTED
: CONTAINS ITEM CONTROL BYTE
: CONTAINS MAX. ERRORS PER TEST
: CONTAINS PC OF LAST ERROR INSTRUCTION
: CONTAINS ADDRESS OF 'GOOD' DATA
: CONTAINS ADDRESS OF 'BAD' DATA
: CONTAINS 'GOOD' DATA
: CONTAINS 'BAD' DATA
: RESERVED--NOT TO BE USED

: AUTOMATIC MODE INDICATOR
: INTERRUPT MODE INDICATOR

: ADDRESS OF SWITCH REGISTER
: ADDRESS OF DISPLAY REGISTER
: TTY KBD STATUS
: TTY KBD BUFFER
: TTY PRINTER STATUS REG. ADDRESS
: TTY PRINTER BUFFER REG. ADDRESS
: CONTAINS NULL CHARACTER FOR FILLS
: CONTAINS # OF FILLER CHARACTERS REQUIRED
: INSERT FILL CHARS. AFTER A "LINE FEED"
: "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
: ESCAPE ON ERROR ADDRESS
: CODE FOR BELL
: QUESTION MARK
: CARRIAGE RETURN
: LINE FEED
*****
: RH11/RPO4/5/6 UNIBUS ADDRESS
: RH11 INTERRUPT VECTOR
: ADDRESS OF KW11-P CSR
: ADDRESS OF KW11-P COUNTER BUFFER
: ADDRESS OF KW11-P VECTOR
: ADDRESS OF KW11-L CONTROL REGISTER
: ADDRESS OF KW11-L VECTOR
: CONTAINS DRIVE NUMBER SELECTED
: CONTENTS ARE FOR SOFTWARE DECISIONS
: 'FORMAT & VERIFY' OR 'CHECK' MODE INDICATOR
: ENDING CYLINDER

```

955	001224	000000	BEGCYL: .WORD	0	; STARTING CYLINDER
956	001226	000022	ENDTRK: .WORD	18.	; ENDING TRACK
957	001230	000000	BEGTRK: .WORD	0	; STARTING TRACK
958	001232	000000	TTRKS: .WORD	0	; TOTAL # OF TRACKS TO BE FORMATTED
959	001234	000000	TTRKSC: .WORD	0	; TOTAL # OF TRACKS COUNTER
960	001236	000000	TRKCNT: .WORD	0	; COUNTS TRKS FROM 0-18 PER CYL
961	001240	000000	PATSEL: .WORD	0	; CONTAINS PATTERN SELECTED
962	001242	000000	PATA: .WORD	0	; 1ST WORD OF PATTERN
963	001244	000000	PATB: .WORD	0	; 2ND WORD OF PATTERN
964	001246	000000	CYLCK: .WORD	0	; STORE CYLINDER ADDRESS FOR FORMAT VERIFICATION
965	001250	000000	RETRY: .WORD	0	; MAINTAINS # OF WRITE RETRIES MADE
966	001252	000000	SAVSEC: .WORD	0	; CONTAINS LAST BAD SECTOR ON FORMAT
967	001254	000000	SAVWC: .WORD	0	; CONTAINS WC FOR REMAINING SECTORS ON ERROR
968	001256	000000	WC: .WORD	0	; 20 OR 22 SECTOR TRACK SIZE (IN WORDS)
969	001258	000000	MWC: .WORD	0	; 2'S COMPLEMENT OF 'WC'
970	001252	000000	HEDERR: .WORD	0	; POSITIONING ERROR DURING FORMAT INDICATOR
971	001254	000000	SEC20: .WORD	0	; 20 OR 22 SECTOR MODE INDICATOR
972					; 0 = 20 SECTOR MODE
973					; 1'S = 22 SECTOR MODE
974	001266	000000	MAXSEC: .WORD	0	; MAXIMUM SECTOR ADDRESS (FOR EITHER 20 OR 22 SECTOR
975					; FORMAT)
976	001270	000000	DS.CYL: .WORD	0	; ADDRESS OF CURRENT CYLINDER
977	001272	000000	DS.TRK: .WORD	0	; ADDRESS OF CURRENT TRACK
978	001274	000000	DDRIVE: .WORD	0	; DRIVE NUMBER OF 'DRIVER' ERROR MESSAGES
979	001276	000000	ATTN: .WORD	0	; ATTENTION REGISTER IMAGE FOR 'DRIVER'
980					; ERROR MESSAGES
981	001300	000000	CNTLC: .WORD	0	; ADDRESS OF 'IC' RETURN
982	001302	000000	CHGADR: .WORD	0	; 'CHANGE RH11 ADDRESS' FLAG
983					
984					; RH11/RPO4/5/6 REGISTERS STORED HERE AFTER AN OPERATION
985					
986	001304	000000	RP.REG: .WORD	0	; RPCS1
987	001306	000000	.WORD	0	; RPWC
988	001310	000000	.WORD	0	; RPBA
989	001312	000000	.WORD	0	; RPOA
990	001314	000000	.WORD	0	; RPCS2
991	001316	000000	.WORD	0	; RPOSI
992	001320	000000	.WORD	0	; RPER1
993	001322	000000	.WORD	0	; RPAS
994	001324	000000	.WORD	0	; RPLA
995	001326	000000	.WORD	0	; RPDB
996	001330	000000	.WORD	0	; RPRM
997	001332	000000	.WORD	0	; RPOT
998	001334	000000	.WORD	0	; RPSN
999	001336	000000	.WORD	0	; RPOF
1000	001340	000000	.WORD	0	; RPCA
1001	001342	000000	.WORD	0	; RPCC
1002	001344	000000	.WORD	0	; RPER2
1003	001346	000000	.WORD	0	; RPER3
1004	001350	000000	.WORD	0	; RPEC1
1005	001352	000000	.WORD	0	; RPEC2
1006					
1007					
1008					
1009					
1010	001354	CCC	FMTDPB: .BYTE	0	; DRIVE NUMBER

; DATA/PARAMETER BLOCK - USED FOR ALL DRIVE OPERATION

1011	001355	000	.BYTE	0	: OFFSET VALUE OR FMT22, ECI, AND HCI
1012	001356	000	.BYTE	00	: COMMAND
1013	001357	000	.BYTE	00	: PSEL AND A17 AND A16
1014	001360	000000	.WORD	00	: WORD COUNT (NEG)
1015	001362	000000	.WORD	00	: BUFFER ADDRESS
1016	001364	000	.BYTE	00	: SECTOR ADDRESS
1017	001365	000	.BYTE	00	: TRACK ADDRESS
1018	001366	000000	.WORD	0	: CYLINDER ADDRESS
1019	001370	001304	.WORD	RP.REG	: ERROR TABLE POINTER
1020	001372	000000	.WORD	0	: STATUS-ERROR INDICATOR
1021					: BIT 15 = 1: ERROR OCCURRED
1022					: BIT 07 = 1: DONE
1023					: BIT 14-10 AND BIT 06-03
1024					: INDICATE TYPE OF ERROR

:PARAMETER POINTER TABLE

1029	001374	001426	000000	001224	TABLE: PAR1,0,BEGCYL
1030	001402	001441	000022	001230	PAR2,18.,BEGTRK
1031	001410	001454	000000	001222	PAR3,0,ENDCYL
1032	001416	001465	000022	001226	PAR4,18.,ENDTRK,0
1033	001424	000000			

:ASCII MESSAGES FOR ADDRESS PARAMETERS

1037	001426	052123	051101	020124	PAR1: .ASCIZ @START CYL @
1038	001434	054503	020114	000	
1039	001441	123	040524	052122	PAR2: .ASCIZ @START TRK @
1040	001446	052040	045522	000040	
1041	001454	047105	020104	054503	PAR3: .ASCIZ @END CYL @
1042	001462	020114	000		
1043	001465	105	042116	052040	PAR4: .ASCIZ @END TRK @
1044	001472	045522	000040		

:SECTOR BUFFER ADDRESS TABLE

1048	001476	025130	ADRTBL: .WORD	BUFP	: ADDRESS OF SECTOR 0
1049	001500	026140	.WORD	BUFP+(520.*1.)	: ADDRESS OF SECTOR 1
1050	001502	027150	.WORD	BUFP+(520.*2.)	: ADDRESS OF SECTOR 2
1051	001504	030160	.WORD	BUFP+(520.*3.)	: ADDRESS OF SECTOR 3
1052	001506	031170	.WORD	BUFP+(520.*4.)	: ADDRESS OF SECTOR 4
1053	001510	032200	.WORD	BUFP+(520.*5.)	: ADDRESS OF SECTOR 5
1054	001512	033210	.WORD	BUFP+(520.*6.)	: ADDRESS OF SECTOR 6
1055	001514	034220	.WORD	BUFP+(520.*7.)	: ADDRESS OF SECTOR 7
1056	001516	035230	.WORD	BUFP+(520.*8.)	: ADDRESS OF SECTOR 8
1057	001520	036240	.WORD	BUFP+(520.*9.)	: ADDRESS OF SECTOR 9
1058	001522	037250	.WORD	BUFP+(520.*10.)	: ADDRESS OF SECTOR 10
1059	001524	040260	.WORD	BUFP+(520.*11.)	: ADDRESS OF SECTOR 11
1060	001526	041270	.WORD	BUFP+(520.*12.)	: ADDRESS OF SECTOR 12
1061	001530	042300	.WORD	BUFP+(520.*13.)	: ADDRESS OF SECTOR 13
1062	001532	043310	.WORD	BUFP+(520.*14.)	: ADDRESS OF SECTOR 14
1063	001534	044320	.WORD	BUFP+(520.*15.)	: ADDRESS OF SECTOR 15
1064	001536	045330	.WORD	BUFP+(520.*16.)	: ADDRESS OF SECTOR 16
1065	001540	046340	.WORD	BUFP+(520.*17.)	: ADDRESS OF SECTOR 17
1066	001542	047350	.WORD	BUFP+(520.*18.)	: ADDRESS OF SECTOR 18

1067	001544	050360	.WORD	BUF+(<520.*19.>	; ADDRESS OF SECTOR 19
1068	001546	051370	.WORD	BUF+(<520.*20.>	; ADDRESS OF SECTOR 20
1069	001550	052400	.WORD	BUF+(<520.*21.>	; ADDRESS OF SECTOR 21

; REMAINING WORD COUNT TABLE

1070					
1071					
1072					
1073	001552	000404	WCTBL: .WORD	260.	; REMAINING WORD COUNT AFTER SECTOR 0
1074	001554	001010	.WORD	260.+(<260.*1.>	; REMAINING WORD COUNT AFTER SECTOR 1
1075	001556	001414	.WORD	260.+(<260.*2.>	; REMAINING WORD COUNT AFTER SECTOR 2
1076	001560	002020	.WORD	260.+(<260.*3.>	; REMAINING WORD COUNT AFTER SECTOR 3
1077	001562	002424	.WORD	260.+(<260.*4.>	; REMAINING WORD COUNT AFTER SECTOR 4
1078	001564	003030	.WORD	260.+(<260.*5.>	; REMAINING WORD COUNT AFTER SECTOR 5
1079	001566	003434	.WORD	260.+(<260.*6.>	; REMAINING WORD COUNT AFTER SECTOR 6
1080	001570	004040	.WORD	260.+(<260.*7.>	; REMAINING WORD COUNT AFTER SECTOR 7
1081	001572	004444	.WORD	260.+(<260.*8.>	; REMAINING WORD COUNT AFTER SECTOR 8
1082	001574	005050	.WORD	260.+(<260.*9.>	; REMAINING WORD COUNT AFTER SECTOR 9
1083	001576	005454	.WORD	260.+(<260.*10.>	; REMAINING WORD COUNT AFTER SECTOR 10
1084	001600	006060	.WORD	260.+(<260.*11.>	; REMAINING WORD COUNT AFTER SECTOR 11
1085	001602	006464	.WORD	260.+(<260.*12.>	; REMAINING WORD COUNT AFTER SECTOR 12
1086	001604	007070	.WORD	260.+(<260.*13.>	; REMAINING WORD COUNT AFTER SECTOR 13
1087	001606	007474	.WORD	260.+(<260.*14.>	; REMAINING WORD COUNT AFTER SECTOR 14
1088	001610	010000	.WORD	260.+(<260.*15.>	; REMAINING WORD COUNT AFTER SECTOR 15
1089	001612	010504	.WORD	260.+(<260.*16.>	; REMAINING WORD COUNT AFTER SECTOR 16
1090	001614	011110	.WORD	260.+(<260.*17.>	; REMAINING WORD COUNT AFTER SECTOR 17
1091	001616	011514	.WORD	260.+(<260.*18.>	; REMAINING WORD COUNT AFTER SECTOR 18
1092	001620	012120	.WORD	260.+(<260.*19.>	; REMAINING WORD COUNT AFTER SECTOR 19
1093	001622	012524	.WORD	260.+(<260.*20.>	; REMAINING WORD COUNT AFTER SECTOR 20
1094	001624	013130	.WORD	260.+(<260.*21.>	; REMAINING WORD COUNT AFTER SECTOR 21

.EVEN

1095
1096
1097

1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153

001626

001626 022112
001630 023231
001632 024502
001634 025014

001636 022153
001640 023236
001642 024504
001644 025020

001646 022211
001650 023313
001652 024520
001654 025024

001656 022247
001660 023341
001662 024526
001664 025030

001666 022304
001670 023236
001672 024504
001674 025020

001676 022340
001700 023400
001702 024536
001704 025014

.SBTTL ERROR POINTER TABLE

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

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

\$ERRTB:

;ERROR 1

EM1
DH1
DT1
DF1

;RH11 INTERRUPT OCCURRED (RPAS=0)

;ERROR 2

EM2
DH2
DT2
DF2

;UNEXPECTED ATTENTION OCCURRED

;ERROR 3

EM3
DH3
DT3
DF3

;MASSBUS PARITY ERROR (MCPE=1)

;ERROR 4

EM4
DH4
DT4
DF4

;MASSBUS PARITY ERROR (PAR=1)

;ERROR 5

EM5
DH2
DT2
DF2

;ADDRESS PLUG CHANGE BIT SET

;ERROR 6

EM6
DH6
DT6
DF1

;RH11 DIDN'T RESPOND TO ADDRESSING

1154				
1155			; ERROR 7	
1156				
1157	001706	000000	0	; UNUSED
1158	001710	000000	0	
1159	001712	000000	0	
1160	001714	000000	0	
1161				
1162			; ERROR 10	
1163				
1164	001716	022402	EM10	; DRIVE OFFLINE
1165	001720	023413	DH10	
1166	001722	024540	DT10	
1167	001724	025034	DF10	
1168				
1169			; ERROR 11	
1170				
1171	001726	022420	EM11	; PERSISTENT DRIVE UNSAFE ERROR
1172	001730	023413	DH10	
1173	001732	024540	DT10	
1174	001734	025034	DF10	
1175				
1176			; ERROR 12	
1177				
1178	001736	022456	EM12	; UNCORRECTABLE MASSBUS PARITY ERROR
1179	001740	023413	DH10	
1180	001742	024540	DT10	
1181	001744	025034	DF10	
1182				
1183			; ERROR 13	
1184				
1185	001746	022521	EM13	; SOFTWARE TIMEOUT
1186	001750	023413	DH10	
1187	001752	024540	DT10	
1188	001754	025034	DF10	
1189				
1190			; ERROR 14	
1191				
1192	001756	022542	EM14	; DRIVE UNSAFE ERROR
1193	001760	023413	DH10	
1194	001762	024540	DT10	
1195	001764	025034	DF10	
1196				
1197			; ERROR 15	
1198				
1199	001766	022565	EM15	; CONTROLLER/DRIVE ERROR DURING WRITE
1200	001770	023413	DH10	
1201	001772	024540	DT10	
1202	001774	025034	DF10	
1203				
1204			; ERROR 16	
1205				
1206	001776	022631	EM16	; CONTROLLER/DRIVE ERROR DURING WRITE CHECK
1207	002000	023413	DH10	
1208	002002	024540	DT10	
1209	002004	025034	DF10	

```

1210
1211 ;ERROR 17
1212
1213 002006 022703 EM17 ;RETRIES NOT SUCESSFUL - SECTOR NOT ACCEPTABLE
1214 002010 023663 DH17
1215 002012 024614 DT17
1216 002014 025050 DF17
1217
1218 ;ERROR 20
1219
1220 002016 022761 EM20 ;DATA ERROR DURING WRITE CHECK
1221 002020 023732 DH20
1222 002022 024626 DT20
1223 002024 025054 DF20
1224
1225 ;ERROR 21
1226
1227 002026 023017 EM21 ;CONTROLLER/DRIVE ERROR VERIFYING HEADERS
1228 002030 023413 DH10
1229 002032 024540 DT10
1230 002034 025034 DF10
1231
1232 ;ERROR 22
1233
1234 002036 023070 EM22 ;'HCE' ERROR VERIFYING HEADERS
1235 002040 023413 DH10
1236 002042 024540 DT10
1237 002044 025034 DF10
1238
1239 ;ERROR 23
1240
1241 002046 023137 EM23 ;CYLINDER FIELD IN HEADER IS NOT CORRECT
1242 002050 024231 DH23
1243 002052 024710 DT23
1244 002054 025074 DF23
1245
1246 ;ERROR 24
1247
1248 002056 023207 EM24 ;WRITE CHECK ERROR
1249 002060 024327 DH24
1250 002062 024726 DT24
1251 002064 025100 DF24
1252
1253 ;;*****
1254
1255 .SBTTL MAIN PROGRAM
1256
1257 ;;*****
1258
1259 002066 012737 177777 001302 BEGIN: MOV #-1,CHGADR ;SET CHANGE 'RH11 BUS ADDRESS' INDICATOR
1260 002074 000402 BR BEGIN2 ;START THE PROGRAM
1261 002076 005037 001302 BEGIN1: CLR CHGADR ;CLEAR THE 'CHANGE RH11 ADDRESS' INDICATOR
1262 002102 000005 BEGIN2: RESET ;CLEAR THE BUS
1263 .SBTTL INITIALIZE THE COMMON TAGS
1264 ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
1265 002104 012706 001100 MOV #$CMTAG,R6 ;;FIRST LOCATION TO BE CLEARED

```

M02

CZRJ880, RPO4/5/6 FMTR MACY11 30(1046) 07-NOV-77 10:07 PAGE 26
 CZRJ88.P11 04-NOV-77 12:54 INITIALIZE THE COMMON TAGS

SEQ 0025

1266	002110	005026			CLR	(R6)+	;; CLEAR MEMORY LOCATION
1267	002112	022706	001140		CMP	#SWR,R6	;; DONE?
1268	002116	001374			BNE	-6	;; LOOP BACK IF NO
1269	002120	012706	001100		MOV	#STACK,SP	;; SETUP THE STACK POINTER
1270					;; INITIALIZE A FEW VECTORS		
1271	002124	012737	006636	000030	MOV	#ERROR,@#EMTVEC	;; EMT VECTOR FOR ERROR ROUTINE
1272	002132	012737	000340	000032	MOV	#340,@#EMTVEC+2	;; LEVEL 7
1273	002140	012737	012014	000034	MOV	#TRAP,@#TRAPVEC	;; TRAP VECTOR FOR TRAP CALLS
1274	002146	012737	000340	000036	MOV	#340,@#TRAPVEC+2	;; LEVEL 7
1275	002154	005037	001160		CLR	#ESCAPE	;; CLEAR THE ESCAPE ON ERROR ADDRESS
1276	002160	112737	000001	001115	MOVB	#1,#SERMAX	;; ALLOW ONE ERROR PER TEST
1277					;; SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS		
1278					;; EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.		
1279	002166	013746	000004		MOV	@#ERRVEC,-(SP)	;; SAVE ERROR VECTOR
1280	002172	012737	002226	000004	MOV	#64\$,@#ERRVEC	;; SET UP ERROR VECTOR
1281	002200	012737	177570	001140	MOV	#DSWR,SWR	;; SETUP FOR A HARDWARE SWICH REGISTER
1282	002206	012737	177570	001142	MOV	#DDISP,DISPLAY	;; AND A HARDWARE DISPLAY REGISTER
1283	002214	022777	177777	176716	CMP	#-1,@SWR	;; TRY TO REFERENCE HARDWARE SWR
1284	002222	001012			BNE	66\$;; BRANCH IF NO TIMEOUT TRAP OCCURRED
1285							;; AND THE HARDWARE SWR IS NOT = -1
1286	002224	000403			BR	65\$;; BRANCH IF NO TIMEOUT
1287	002226	012716	002234		64\$: MOV	#65\$,(SP)	;; SET UP FOR TRAP RETURN
1288	002232	000002			RTI		
1289	002234	012737	000176	001140	65\$: MOV	#SWREG,SWR	;; POINT TO SOFTWARE SWR
1290	002242	012737	000174	001142	MOV	#DISPREG,DISPLAY	
1291	002250	012637	000004		66\$: MOV	(SP)+,@#ERRVEC	;; RESTORE ERROR VECTOR
1292							
1293	002254	000005			RESET		;; CLEAR WORLD
1294	002256	012737	000240	000036	MOV	#PRS,@#TRAPVEC+2	;; CHANGE TRAP PRIORITY BACK TO 5
1295	002264	012737	000240	000032	MOV	#PR5,@#EMTVEC+2	;; CHANGE EMT PRIORITY BACK TO 5
1296	002272	012737	002076	001300	MOV	#BEGIN1,CNTLC	;; 'CONTROL C' ADDRESS

```

1297 002300 005227 177777          INC      #-1          ; FIRST START ?
1298 002304 001010          BNE      1$          ; BR IF NOT
1299 002306 104401 025130          TYPE    TITLE        ; ADPS OF 'TITLE' MESSAGE
1300 002312 122737 000011 000041      CMPB    #11,41       ; LOADED FROM AN RPO4/5/6 ?
1301 002320 001002          BNE      1$          ; BR IF NOT
1302 002322 104401 025205          TYPE    ,LOADR:      ; INSTRUCT OPERATOR TO REMOVE 'XXDP'
1303                                     ; PACK FROM DRIVE 0
1304 002326 004737 010204          1$: JSR     PC,STKINT  ; TURN ON THE KEYBOARD INTERRUPT
      .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
1305                                     ; ARE WE RUNNING UNDER XXDP/ACT?
1306 002332 075737 000042          TST     @#42        ; BRANCH IF YES
1307 002336 001006          BNE      67$        ; SOFTWARE SWITCH REG SELECTED?
1308 002340 023727 001140 000176      CMP     SWR,#SWREG  ; BRANCH IF NO
1309 002346 001005          BNE      68$        ; GET SOFT-SWR SETTINGS
1310 002350 104406          GTSWR
1311 002352 000403          BR      68$
1312 002354 112737 000001 001134 67$: MOVB    #1,$AUTOB  ; ; SET AUTO-MODE INDICATOR
      68$:
1313 002362 005227 177777          INC      #-1          ; CHECK FOR FIRST START
1314 002366 001010          BNE      SETVEC      ; BR IF NOT
1315 002370 004737 025432          JSR     PC,BUSADR   ; CHECK THE RH11 ADDRESS
1316 002374 013737 001172 012246      MCV     $RPADR,$RPADR ; RH11 ADDRESS
1317 002402 013737 001174 012250      MOV     $RPVEC,$RPVEC ; RH11 VECTOR ADDRESS
1318
1319                                     ; DISPLAY DRIVE STATUS AND SET UP THE OTHER UNITS THE
1320                                     ; PROGRAM WILL USE
1321
1322
1323 002410 004737 005742          SETVEC: JSR     PC,ST.CLK ; START THE CLOCK
1324 002414 004737 012264          JSR     PC,RPINIT   ; INITIALIZE THE RPO4/5/6 DRIVER
1325 002420 012737 177777 012206      MOV     #-1,$SAVEFG ; SET THE SAVE REGISTERS FLAG
1326 002426 005227 177777          INC     #-1          ; SEE IF FIRST START
1327 002432 001404          BEQ     11$          ; BR IF YES
1328 002434 032777 000004 176476      BIT     #SWD2,@SWR  ; TYPEOUT THE DRIVE STATUS TABLE ?
1329 002442 001076          BNE     10$          ; BR IF NOT
1330 002444 012737 000340 177776 11$: MOV     #PR7,$PS   ; SET PRIORITY TO 7
1331 002452 005004          CLR     R4          ; DRIVE TABLE POINTER
1332 002454 104401 001167          TYPE   '$CRLF'      ; CR-LF
1333 002460 104401 020640          TYPE   ',SYSTAT'   ; TYPE STATUS HEADING
1334 002464
1335 002464 010446          1$: MOV     R4,-(SP)   ; ; SAVE R4 FOR TYPEOUT
1336                                     ; ; TYPE DRIVE NUMBER
1337 002466 104403          TYPOS   2          ; ; GO TYPE--OCTAL ASCII
1338 002470 002          .BYTE  0          ; ; TYPE 2 DIGIT(S)
1339 002471 000          .BYTE  0          ; ; SUPPRESS LEADING ZEROS
1340 002472 104401 020703          TYPE   'LIN4SP'    ; ; SPACES
1341 002476 105764 012120          TSTB   DRVSTA(R4)  ; ; CHECK DRIVE'S STATUS
1342 002502 100416          BMI     4$          ; ; BR IF UNSAFE
1343 002504 001020          BNE     5$          ; ; BR IF ONLINE
1344 002506 105764 012130          TSTB   DRVSTYP(R4) ; ; SEE IF OFFLINE OR NONEXISTENT
1345 002512 001404          BEQ     2$          ; ; BR IF NONEXISTENT
1346 002514 100006          BPL     3$          ; ; BR IF OFFLINE
1347 002516 104401 020553          TYPE   NOTRP       ; ; DRIVE NOT AN RPO4/5/6
1348 002522 000440          BR      4$          ; ; CHECK NEXT DRIVE
1349 002524 104401 020574          2$: TYPE   NOTPRS   ; ; DRIVE NOT PRESENT
1350 002530 000435          BR      4$          ; ; CHECK NEXT DRIVE
1351 002532 104401 020532          3$: TYPE   UNTOFF   ; ; DRIVE OFFLINE
1352 002536 000405          BR      6$          ; ; PRINT DRIVE TYPE
    
```

```

1353 002540 104401 020611 4S: TYPE NOTSAF ;DRIVE UNSAFE
1354 002544 000402 BR 6S ;PRINT DRIVE TYPE
1355 002546 104401 020543 5S: TYPE ,UNTON ;DRIVE ONLINE
1356 002552 104401 020705 6S: TYPE LINSF ;SPACES
1357 002556 012737 020621 002622 MOV #RPO4B,8S ;ADDRESS OF RPO4 MESSAGE
1358 002564 132764 000001 012130 BITB #BIT00,DRV TYP(R4) ;RPO4 ?
1359 002572 001012 BNE 7S ;BR IF YES
1360 002574 012737 000626 002622 MOV #RPOS,8S ;ADDRESS OF RPOS MESSAGE
1361 002602 132764 000002 012130 BITB #BIT01,DRV TYP(R4) ;RPOS ?
1362 002610 001003 BNE 7S ;BR IF YES
1363 002612 012737 020633 002622 MOV #RPO6,8S ;ADDRESS OF RPO6 MESSAGE
1364 002620 104401 7S: TYPE ;TYPE THE DRIVE TYPE MESSAGE
1365 002622 000000 8S: .WORD 0 ;MESSAGE ADDRESS HERE
1366 002624 104401 001167 9S: TYPE ,SCRLF ;CR-LF
1367 002630 005204 INC R4 ;INCREMENT DRIVE NUMBER/TABLE POINTER
1368 002632 020427 000010 CMP R4,#8. ;FINISHED ?
1369 002636 001312 BNE 1S ;BR IF NOT
1370 002640 104401 001167 10S: TYPE ,SCRLF ;CR-LF
1371
1372
1373 ;SEE WHICH MODE THE PROGRAM IS TO BE RUN IN.
1374 ;MODES ARE: 'FORMAT & VERIFY' OR 'CHECK FORMAT'
1375
1376 002644 005037 001220 M1: CLR MODE ;SET MODE TO 'CHECK FORMAT'
1377 002650 104401 021121 TYPE ,MMODE ;TYPE 'PROGRAM MODE'
1378 002654 104411 RDLIN ;READ THE KEYBOARD
1379 002656 012601 MOV (SP)+,R1 ;GET ADDRESS OF INPUT
1380 002660 105711 TSTA (R1) ;'CR' ENTERED (DEFAULT)
1381 002662 001414 BEQ 2S ;BR IF YES
1382 002664 122711 000103 CMPB #'C',(R1) ;'CHECK' ?
1383 002670 001406 BEQ 1S ;BR IF YES
1384 002672 122711 000106 CMPB #'F',(R1) ;'FORMAT' ?
1385 002676 001411 BEQ 3S ;BR IF YES
1386 002700 104401 001166 TYPE ,SQUES ;NO CORRECT ENTRY
1387 002704 000757 BR M1 ;TRY AGAIN
1388 002706 104401 021174 1S: TYPE ,MHECK ;TYPE REST OF 'CHECK'
1389 002712 000410 BR M1A ;GET STARTING ADDRESS
1390 002714 104401 021153 2S: TYPE ,MFORMT ;TYPE DEFAULT MESSAGE
1391 002720 000402 BR 4S ;SET UP MODE
1392 002722 104401 021207 3S: TYPE ,MORMAT ;TYPE REST OF 'FORMAT'
1393 002726 012737 177777 001220 4S: MOV #-1,MODE ;SET MODE TO 'FORMAT & VERIFY'
1394
1395 ;FIND OUT IF FORMAT IS TO BE IN 20 OR 22 SECTOR MODE
1396
1397 002734 104401 021227 M1A: TYPE ,MSIZE ;TYPE FORMAT MODE REQUEST
1398 002740 104411 RDLIN ;READ THE KEYBOARD
1399 002742 012601 MOV (SP)+,R1 ;ADDRESS OF ENTRY
1400 002744 122711 000131 CMPB #'Y',(R1) ;IS ENTRY A 'Y' ?
1401 002750 001410 BEQ 1S ;BR IF IT IS
1402 002752 122711 000116 CMPB #'N',(R1) ;IS ENTRY A 'N' ?
1403 002756 001424 BEQ 2S ;BR IF IT IS
1404 002760 105711 TSTB (R1) ;DEFAULT MODE ?
1405 002762 001403 BEQ 1S ;BR IF IT IS
1406 002764 104401 001166 TYPE ,SQUES ;TYPE '?'
1407 002770 000761 BR M1A ;TRY AGAIN
1408 002772 104401 021300 1S: TYPE ,MSEC22 ;TYPEOUT MODE SELECTED
    
```

```

1409 002776 012737 013130 001256      MOV      #<260.*22.>,WC ; TRACK SIZE IN 22 SECTOR MODE
1410 003004 012737 164650 001260      MOV      #-<260.*22.>,MWC ; 2'S COMPLEMENT WORD COUNT
1411 003012 012737 177777 001264      MOV      #-1,SEC20 ; 22 SECTOR INDICATOR
1412 003020 012737 000025 001266      MOV      #21,MAXSEC ; MAX SECTOR ADDRESS IN 22 SECTOR MODE
1413 003026 000415 ; ; CONTINUE
1414 003030 104401 021357 ; 2$: TYPE ; TYPE OUT 20 SECTOR MODE SELECTED
1415 003034 012737 012120 001256      MOV      #<260.*20.>,WC ; TRACK SIZE IN 20 SECTOR MODE
1416 003042 012737 165660 001260      MOV      #-<260.*20.>,MWC ; 2'S COMPLEMENT WORD COUNT
1417 003050 005037 001264 ; ; 20 SECTOR INDICATOR
1418 003054 012737 000023 001266      MOV      #19,MAXSEC ; MAX SECTOR ADDRESS IN 20 SECTOR MODE
1419 003062 005037 001230 ; M18: CLR ; CLEAR STARTING TRACK ADDRESS
1420 003066 005037 001224 ; CLR ; CLEAR BEGINNING CYLINDER ADDRESS
1421 003072 012737 000022 001226      MOV      #18,ENDTRK ; SETUP END TRACK ADDRESS
1422 003100 012737 000002 001240      MOV      #2,PATSEL ; SETUP FOR WORST CASE PATTERN
1423 003106 112737 177777 001354      MOV      #-1,FMTDPB ; SETUP INVALID DRIVE NUMBER
1424 003114 000400 ; BR ; BRANCH TO START
1425
1426 ; GO FIND OUT WHAT DRIVE
1427
1428 003116 012737 006116 001300 ; M0: MOV ; CONTROL C ABORT ENTRANCE
1429 003124 005037 001112 ; CLR ; CLEAR THE ERROR ACCUMULATOR
1430 003130 004737 010204 ; JSR ; INITIALIZE THE TTY KEYBOARD
1431 003134 104401 020657 ; TYPE ; ASK FOR DRIVE NUMBER
1432 003140 104411 ; RDLIN ; READ THE KEYBOARD
1433 003142 012601 ; MOV ; ADDRESS OF ENTRY
1434 003144 004537 006426 ; JSR ; CHECK ONE CHARACTER
1435 003150 003176 ; JS ; ILLEGAL CHARACTER
1436 003152 003164 ; JS ; CARRIAGE RETURN
1437 003154 003176 ; JS ;
1438 003156 003164 ; JS ;
1439 003160 003204 ; JS ;
1440 003162 003176 ; JS ;
1441 003164 005037 001214 ; 2$: CLR ; DRIVE
1442 003170 104401 020701 ; TYPE ; MDRVD
1443 003174 000413 ; BR ; $
1444 003176 104401 001166 ; 3$: TYPE ; $QUES
1445 003202 000745 ; BR ; M0
1446 003204 010237 001214 ; 4$: MOV ; R2,DRIVE
1447 003210 005702 ; TST ; R2
1448 003212 001004 ; BNE ; $
1449 003214 122737 000011 000041 ; CMPB ; #11,41
1450 003222 001431 ; BEQ ; 7$
1451 003224 004737 005742 ; 5$: JSR ; PC,ST.CLK
1452 003230 004737 012264 ; JSR ; PC,RPINIT
1453 003234 012737 177777 012206 ; MOV ; #-1,SAVEFG
1454 003242 012737 177777 012210 ; MOV ; #-1,SEEKFG
1455 003250 005037 177776 ; CLR ; PS
1456 003254 105762 012120 ; TSTB ; DRVSTA(R2)
1457 003260 003015 ; BGT ; 8$
1458 003262 001403 ; BEQ ; 6$
1459 003264 104401 021067 ; TYPE ; MUSD
1460 003270 000712 ; BR ; M0
1461 003272 105762 012130 ; 6$: TSTB ; DRVSTYP(R2)
1462 003276 001003 ; BNE ; 7$
1463 003300 104401 020762 ; TYPE ; MDRNP
1464 003304 000704 ; BR ; M0
    
```

```

1465 003306 104401 021007 7$: TYPE MER11 ;'DRIVE NOT AVAILABLE'
1466 003312 000701 BR M0 ;GO GET DRIVE NUMBER AGAIN
1467 003314 123737 001214 001354 8$: CMPB DRIVE,FMTDPB ;SAME DRIVE AS LAST TIME ?
1468 003322 001430 BEQ M2 ;BR IF IT IS
1469 003324 113737 001214 001354 MOVB DRIVE,FMTDPB ;SETUP DRIVE ADDRESS
1470 003332 012737 000632 001222 MOV #410,ENDCYL ;SETUP FOR RP04/5
1471 003340 132762 000003 012130 BITB #BIT00!BIT01,DRV1YP(R2) ;SEE IF DRIVE RP04/5
1472 003346 001003 BNE 9$ ;BR IF EITHER
1473 003350 012737 001456 001222 MOV #814,ENDCYL ;SETUP ENDING CYLINDER FOR RP06
1474 003356 005037 001230 9$: CLR BEGTRK ;CLEAR STARTING TRACK ADDRESS
1475 003362 005037 001224 CLR BEGCYL ;CLEAR STARTING CYLINDER ADDRESS
1476 003366 013737 001222 001376 MOV ENDCYL,TABLE+2 ;ENTRY LIMIT FOR BEGINNING CYLINDER
1477 003374 013737 001222 001412 MOV ENDCYL,TABLE+16 ;ENTRY LIMIT FOR END CYLINDER
1478 003402 000400 BR M2 ;GET ADDRESS LIMITS FROM THE OPERATOR
1479
1480 ;GET ADDRESS LIMITS
1481
1482 003404 104401 020710 M2: TYPE ENTADR ;'ENTER ADDRESS LIMITS'
1483 003410 004737 006236 JSR PC,PARENT ;GET THE ADDRESS LIMITS
1484 003414 023737 001222 001224 CMP ENDCYL,BEGCYL ;SEE IF ENDING CYLINDER EQ TO GT THAN BEGINNING
1485 003422 101010 BHI M4 ;BR IF HIGHER
1486 003424 103404 BLO 3$ ;BR IF LESS
1487 003426 023737 001226 001230 2$: CMP ENDRK,BEGTRK ;SEE IF ENDING TRACK EQ OR GT THAN BEGINNING
1488 003434 103003 BHIS M4 ;BR IF YES
1489 003436 104401 021456 3$: TYPE MADRER ;INVALID ADDRESS ENTERED
1490 003442 000760 BR M2 ;TRY AGAIN
1491
1492 ;GO GET DATA PATTERN FOR FORMAT
1493
1494 003444 104401 021560 M4: TYPE ,MSELP ;GO TYPE 'SELECT PATTERN'
1495 003450 104411 RDLIN ;READ THE KEYBOARD
1496 003452 012601 MOV (SP)+,R1 ;ENTRY ADDRESS
1497 003454 004537 006426 JSR R5,CK.CHR ;CHECK ONE CHARACTER
1498 003460 003514 3$ ;ILLEGAL CHARACTER
1499 003462 003474 1$ ;CARRIAGE RETURN
1500 003464 003514 3$ ;" "
1501 003466 003474 1$ ;" "
1502 003470 003506 2$ ;DIGIT 0-7
1503 003472 003514 3$ ;DIGIT 8-9
1504 003474 104401 021660 1$: TYPE MPATD ;TYPE DEFAULT PATTERN
1505 003500 012702 000002 MOV #2,R2 ;WORST CASE PATTERN
1506 003504 000406 BR 4$ ;GO SAVE PATTERN
1507 003506 020227 000002 2$: CMP R2,#2 ;IS # LARGER THAN 2
1508 003512 101403 BLOS 4$ ;BRANCH IF NOT
1509 003514 104401 001166 3$: TYPE $QUES ;TYPE '?'
1510 003520 000751 BR M4 ;RETYPE LINE
1511 003522 010237 001240 4$: MOV R2,PATSEL ;SAVE PATTERN SELECTED
1512
1513 ;GO TYPE 'STARTING FORMAT ON DRIVE N'
1514
1515 003526 012737 006136 001300 M5: MOV #TYPADR,CNTLC ;CHANGE IC ENTRANCE
1516 003534 005737 001220 TST MODE ;'FORMAT' OR 'CHECK' MODE ?
1517 003540 001403 BEQ 1$ ;BR IF 'CHECK' MODE
1518 003542 104401 021673 TYPE MSFOU ;TYPE 'STARTING FORMAT ON DRIVE N'
1519 003546 000402 BR 2$
1520 003550 104401 021730 1$: TYPE ,MSCHK ;TYPE 'STARTING CHECK ON DRIVE N'
    
```

E03

CZRJ880 RP04.5/6 FMTR MACY11 30(1046)
 CZRJ88.P11 04-NOV-77 12:54

07-NOV-77 10:07 PAGE 31
 GET VALUE FOR SOFTWARE SWITCH REGISTER

SEQ 0030

```

1521 003554
1522 003554 013746 001214
1523
1524 003560 104403
1525 003562 001
1526 003563 000
1527 003564 104401 001167
1528
1529
1530
1531 003570 023737 001222 001224 CKADRS: CMP ENDCYL,BEGCYL ; STARTING AND ENDING CYLINDERS THE SAME ?
1532 003576 001011 BNE 1$ ; BRANCH IF BEGCYL NOT EQUAL TO ENDCYL
1533 003600 013737 001226 001232 MOV ENDRK,TTRKS ; END TRACK ADDRESS
1534 003606 163737 001230 001232 SUB BEGTRK,TTRKS ; SUBTRACT THE STARTING TRACK ADDRESS
1535 003614 005237 001232 INC TTRKS ; MAKE THE NUMBER OF TRACKS INCLUSIVE
1536 003620 000427 BR SETPAT ; SETUP THE DATA PATTERN
1537 003622 013732 001222 1$: MOV ENDCYL,R2 ; ENDING CYLINDER
1538 003626 163732 001224 SUB BEGCYL,R2 ; SUBTRACT THE STARTING CYLINDER
1539 003632 013737 001226 001232 MOV ENDRK,TTRKS ; CALCULATE THE RESIDUAL TRACKS
1540 003640 163737 001230 001232 SUB BEGTRK,TTRKS ; SUBTRACT THE STARTING TRACK
1541 003646 100004 BPL 2$ ; BR IF ENDING TRACK GREATER
1542 003650 062737 000023 001232 ADD #19.,TTRKS ; CORRECT THE VALUE
1543 003656 005302 DEC R2 ; COUNT THE PARTIAL TRACK
1544 003660 005237 001232 2$: INC TTRKS ; MAKE THE NUMBER INCLUSIVE
1545 003664 005302 3$: DEC R2 ; DECREMENT THE CYLINDER COUNT
1546 003666 100404 BMI SETPAT ; BR IF DONE
1547 003670 062737 000023 001232 ADD #19.,TTRKS ; ADD CYLINDER WORTH OF TRACKS
1548 003676 000772 BR 3$ ; CONTINUE
1549
1550 ; THIS CODE SETS UP THE SELECTED DATA PATTERN TO BE WRITTEN ON EACH SECTOR IMAGE
1551
1552 003700 005037 001242 SETPAT: CLR PATA ; CLEAR DATA PATTERN A
1553 003704 005037 001244 CLR PATB ; CLEAR DATA PATTERN B
1554 003710 005737 001240 TST PATSEL ; SEE IF PATTERN OF ONES
1555 003714 001416 BEQ 1$ ; BR IF SO
1556 003716 005137 001242 COM PATA ; SET PATA TO ONES
1557 003722 005137 001244 COM PATB ; SET PATB TO ONES
1558 003726 022737 000001 001240 CMP #1,PATSEL ; SEE IF PATTERN OF ZEROS
1559 003734 001406 BEQ 1$ ; BRANCH IF SO
1560 003736 012737 165555 001242 MOV #165555,PATA ; SET UP WORST CASE
1561 003744 012737 133333 001244 MOV #133333,PATB ; SET UP WORST CASE
1562 003752 013703 001256 1$: MOV WC,R3 ; SET UP COUNTER
1563 003756 012700 025130 MOV #BUF,RO ; SET UP MEMORY POINTER
1564 003762 013701 001242 MOV PATA,R1 ; SET UP PATTERN IN R1
1565 003766 013702 001244 MOV PATB,R2 ; SET UP PATTERN IN R2
1566 003772 010120 2$: MOV R1,(R1)+ ; MOV 1ST PAT INTO MEM
1567 003774 010220 MOV R2,(R2)+ ; MOV 2ND PAT INTO MEM
1568 003776 005303 DEC R3 ; KEEP COUNT
1569 004000 005303 DEC R3 ; KEEP COUNT
1570 004002 001373 BNE 2$ ; DO IT AGAIN IF R3 NOT 0
1571
1572 ; THIS CODE SETS UP FOR THE ACTUAL FORMAT
1573
1574 004004 004737 005402 WTRK: JSR PC,SETTBL ; GO SET UP DRIVER TABLE
1575 004010 004737 005602 JSR PC,SETHDR ; GO INITIALIZE HEADERS IN THE SECTOR BUFFER IN CORE
1576 004014 112737 000020 001355 MOVB #20,FMTDPB+1 ; LOAD FMT22 BIT

```

F03

CZRJB80 RPO4 5/6 FMTR MACY11 30(1046) 07-NOV-77 10:07 PAGE 32
 CZRJB8.P11 04-NOV-77 12:54 GET VALUE FOR SOFTWARE SWITCH REGISTER

SEQ 0031

1577	004022	005737	001264			TST	SEC20	:	18 BIT MODE ?
1578	004026	001002				BNE	1\$:	BR IF NOT
1579	004030	105037	001355			CLRB	FMTDPB+1	:	CLEAR THE FMT22 BIT
1580	004034	112737	000143	001356	1\$:	MOVB	#SETFMT,FMTDPB+2	:	'LOAD FORMAT' COMMAND
1581	004042	004037	013034		2\$:	JSR	RD,RPO4	:	START THE COMMAND
1582	004046	001354				FMTDPB		:	DPB ADDRESS
1583	004050	000774				BR	2\$:	QUEUE FULL RETURN
1584	004052	005737	001372		3\$:	TST	FMTDPB+16	:	LOOK FOR DONE
1585	004056	001775				BEQ	3\$:	LOOP UNTIL FINISHED
1586	004060	005037	001216		WRTRK1:	CLR	SOF5W	:	CLEAR ERROR COUNTER
1587	004064	005037	001250			CLR	RETRY	:	ZERO THE RETRY COUNTER
1588	004070	105037	001364			CLRB	FMTDPB+10	:	RESTORE SECTOR
1589	004074	013737	001260	001360		MOV	MWC,FMTDPB+4	:	RESTORE WC
1590	004102	012737	025130	001362		MOV	#BUFP,FMTDPB+6	:	RESTORE BA
1591	004110	005737	001220			TST	MODE	:	'FORMAT' OR 'CHECK' MODE ?
1592	004114	001450				BEQ	CKTRK	:	BR IF 'CHECK' MODE
1593	004116	112737	000163	001356	WRTRK2:	MOVB	#WRTHD,FMTDPB+2	:	SET WRITE HEADER & DATA COMMAND IN TBL
1594	004124	012737	004124	001110		MOV	#,\$LPERR	:	SETUP LOOP ON ERROR ADDRESS
1595	004132	012706	001100			MOV	#STACK,SP	:	LOAD STACK POINTER
1596	004136	004037	013034		1\$:	JSR	RD,RPO4	:	GO FORMAT A TRACK
1597	004142	001354				FMTDPB		:	ADRS OF PARAMETERS - TBL
1598	004144	000774				BR	1\$:	WAIT FOR QUEUE IF FULL
1599	004146	005737	001372		2\$:	TST	FMTDPB+16	:	WAIT FOR COMMAND TO COMPLETE
1600	004152	001775				BEQ	2\$:	BRANCH IF NOT DONE
1601	004154	100024				BPL	4\$:	BRANCH IF NO ERROR
1602	004156	012737	004204	001160		MOV	#3\$,\$ESCAPE	:	ESCAPE TO 3\$ ON ERROR
1603	004164	004737	005234			JSR	PC,ERINDX	:	SEE WHICH ERROR
1604	004170	104010				ERROR	10	:	DRIVE OFFLINE
1605	004172	104011				ERROR	11	:	PERSISTENT DRIVE UNSAFE ERROR
1606	004174	104012				ERROR	12	:	UNCORRECTABLE MASSBUS PARITY ERROR
1607	004176	104013				ERROR	13	:	SOFTWARE TIMEOUT
1608	004200	104014				ERROR	14	:	DRIVE UNSAFE ERROR
1609	004202	104015				ERROR	15	:	DRIVE/CONTROLLER ERROR DURING WRITE
1610	004204	004737	005334		3\$:	JSR	PC,LOP.CK	:	LOOP ON THE ERROR ?
1611	004210	022737	000003	001250		CMP	#3,RETRY	:	ERROR RETRY LIMIT ?
1612	004216	001403				BEQ	4\$:	BR IF REACHED
1613	004220	005237	001250			INC	RETRY	:	COUNT THE ERROR
1614	004224	000744				BR	1\$:	TRY AGAIN
1615	004226	005037	001160		4\$:	CLR	\$ESCAPE	:	CLEAR ERROR ESCAPE ADDRESS
1616	004232	005037	001250			CLR	RETRY	:	CLEAR THE RETRY COUNTER
1617									
1618									
1619									;CHECK THE TRACK JUST WRITTEN
1620	004236	112737	000153	001356	CKTRK:	MOVB	#WCKHD,FMTDPB+2	:	SET WRITE CHECK HEADER & DATA COMMAND IN TBL
1621	004244	012737	004244	001110		MOV	#,\$LPERR	:	SETUP LOOP ON ERROR ADDRESS
1622	004252	012706	001100			MOV	#STACK,SP	:	LOAD STACK POINTER
1623	004256	004037	013034		1\$:	JSR	RD,RPO4	:	GO CHECK THE TRACK JUST FORMATTED
1624	004262	001354				FMTDPB		:	ADRS OF PARAMETERS - TBL
1625	004264	000774				BR	1\$:	WAIT FOR QUEUE IF FULL
1626	004266	005737	001372		2\$:	TST	FMTDPB+16	:	WAIT FOR COMMAND TO COMPLETE
1627	004272	001775				BEQ	2\$:	BRANCH IF NOT DONE
1628	004274	100112				BPL	8\$:	BRANCH IF NO ERROR
1629	004276	113737	001312	001252		MOVB	RP.REG+RPDA,SAVSEC	:	GET THE SECTOR ADDRESS
1630	004304	001005				BNE	3\$:	BRANCH IF NOT SECTOR 0
1631	004306	013737	001266	001252		MOV	MAXSEC,SAVSEC	:	RESTORE TO LAST SECTOR +1
1632	004314	005237	001252			INC	SAVSEC	:	INCREMENT TO LAST SECTOR +1

1633	004320	005337	001252		3%:	DEC	SAVSEC	:ADJUST SECTOR TO THE ONE THAT FAILED
1634	004324	012737	004570	001160		MOV	#10\$, \$ESCAPE	:ESCAPE TO 10\$ ON ERROR
1635	004332	004737	00520			JSR	PC, ERINDX	:SEE WHICH ERROR
1636	004336	104010				ERROR	10	:DRIVE OFFLINE
1637	004340	104011				ERROR	11	:PERSISTENT DRIVE UNSAFE ERROR
1638	004342	104012				ERROR	12	:UNCORRECTABLE MASSBUS PARITY ERROR
1639	004344	104013				ERROR	13	:SOFTWARE TIMEOUT
1640	004346	104014				ERROR	14	:DRIVE UNSAFE ERROR
1641	004350	032737	040000	001314		BIT	#WCE, RP.REG+RPCS2	:WRITE CHECK ERROR ?
1642	004356	001005				BNE	4\$:BR IF SET
1643	004360	033727	001320			BIT	RP.REG+RPER1, (PC)+	:CHECK FOR DATA ERRORS
1644	004364	130620				.WORD	DCK!OPI!DTE!4CRC!	:HCE!FER ;DATA ERROR BITS
1645	004366	001001				BNE	4\$:BR IF SET
1646	004370	104016				ERROR	16	:CONTROLLER/DRIVE ERROR DURING WRITE CHECK
1647	004372	005737	001220		4%:	TST	MODE	:FORMAT OR CHECK MODE ?
1648	004376	001407				BEQ	5\$:BR IF CHECK
1649	004400	005737	001216			TST	SOF SW	:RETRYING THE SECTOR ?
1650	004404	001404				BEQ	5\$:BR IF NOT
1651	004406	012737	004612	001160		MOV	#11\$, \$ESCAPE	:ESCAPE TO 11\$ ON ERROR
1652	004414	104017				ERROR	17	:SECTOR NOT ACCEPTABLE
1653	004416	005037	001160		5%:	CLR	\$ESCAPE	:CLEAR THE ERROR ESCAPE ADDRESS
1654	004422	032737	040000	001316		BIT	#ERR, RP.REG+RPDS1	:DATA ERROR ?
1655	004430	001002				BNE	6\$:BR IF IT IS
1656	004432	104024				ERROR	24	:WRITE CHECK ERROR
1657	004434	000401				BR	7\$:CONTINUE
1658	004436	104020			6%:	ERROR	20	:DATA ERROR DURING WRITE CHECK
1659	004440	013701	001252		7%:	MOV	SPVSEC, R1	:FAILING SECTOR ADDRESS
1660	004444	113737	001252	001364		MOVB	SAVSEC, FMTDPB+10	:SECTOR ADDRESS
1661	004452	006301				ASL	R1	:SETUP INDEX TO ADDRESS WORDS
1662	004454	016137	001476	001362		MOV	ADRTBL(R1), FMTDPB+6	:BUFFER ADDRESS FOR FAILING SECTOR
1663	004462	013746	001260			MOV	MWC, -(SP)	:GET MAXIMUM WORD COUNT VALUE (2'S COMP)
1664	004466	066116	001552			ADD	WCTBL(R1), (SP)	:ADD WORD COUNT THROUGH FAILING SECTOR
1665	004472	012637	001254			MOV	(SP)+, SAVWC	:STORE WORD COUNT FOR REMAINDER OF TRACK
1666	004476	005737	001220			TST	MODE	:FORMAT OR CHECK MODE ?
1667	004502	001421				BEQ	9\$:BR IF CHECK
1668	004504	012737	177374	001360		MOV	#-260., FMTDPB+4	:WORD COUNT FOR 1 SECTOR
1669	004512	005237	001216			INC	SOF SW	:INDICATE THAT A RETRY IS IN PROGRESS
1670	004516	000137	004116			JMP	WRTRK2	:REFORMAT ERROR SECTOR
1671	004522	005737	001216		8%:	TST	SOF SW	:RETRY IN PROGRESS ?
1672	004526	001431				BEQ	11\$:BR IF NOT
1673	004530	022737	000002	001216		CMP	#2, SOF SW	:SEE IF LAST RETRY
1674	004536	001403				BEQ	9\$:BR IF IT IS
1675	004540	005237	001216			INC	SOF SW	:INCREMENT RETRY COUNT
1676	004544	000644				BR	1\$:READ AGAIN
1677	004546	123737	001266	001364	9%:	CMPB	MAXSEC, FMTDPB+10	:SEE IF LAST SECTOR ON THE TRACK
1678	004554	001416				BEQ	11\$:BR IF IT IS
1679	004556	004737	005554			JSR	PC, SCAWC	:SETUP TO CHECK REMAINING SECTORS ON THE TRACK
1680	004562	005037	001216			CLR	SOF SW	:CLEAR RETRY COUNTER
1681	004566	000633				BR	1\$:FINISH CHECKING THE TRACK
1682	004570	004737	005334		10%:	JSR	PC, LOP.CK	:CHECK FOR LOOP ON ERROR
1683	004574	022737	000003	001250		CMP	#3, RETRY	:ERROR RETRY REACHED ?
1684	004602	001403				BEQ	11\$:BR IF IT IS
1685	004604	005237	001250			INC	RETRY	:COUNT THE RETRY
1686	004610	000622				BR	1\$:DO THE WRITE CHECK AGAIN
1687	004612	005037	001160		11%:	CLR	\$ESCAPE	:CLEAR THE ERROR RETURN ESCAPE ADDRESS
1688	004616	005037	001250			CLR	RETRY	:CLEAR THE RETRY COUNTER

H03

CZRJB80, RPO4.5/6 FMTR MACY11 30(1046)
 CZRJB8.P11 04-NOV-77 12:54

07-NOV-77 10:07 PAGE 34
 GET VALUE FOR SOFTWARE SWITCH REGISTER

SEQ 0033

```

1689 004622 032777 000002 174310      BIT      #BIT1,DSWR      ;LOOP ON CURRENT TRACK ?
1690 004630 001402                BEQ      12$          ;BR IF NOT
1691 004632 000137 004060                JMP      WTRK1       ;START AGAIN ON THE SAME TRACK
1692 004636 004737 005472      12$:     JSR      PC,TRKTST ;GET UPDATED TRACK AND CYLINDER ADDRESSES
1693 004642 000402                BR       HDREAD      ;RETURN HERE IF DONE - DO QUICK CHECK OF DISK
1694 004644 000137 004060                JMP      WTRK1       ;CONTINUE WITH THE FORMAT
1695
1696
1697
1698
1699
1700 004650 005037 001262      HDREAD: CLR      HEDERR ;CLEAR HEADER CHECK ERROR INDICATOR
1701 004654 004737 005402                JSR      PC,SETTBL   ;GO SET UP DRIVER TABLE
1702 004660 004737 005602                JSR      PC,SETHDR   ;GO SET UP HEADERS IN CORE
1703 004664 013737 001224 001246      MOV      BEGCVL,CYLCK ;USE 'BEGCVL' FOR FORMAT VERIFICATION
1704 004672 012737 177774 001360      MOV      #-4,FMTDPB+4 ;SET UP WORD COUNT
1705 004700 012737 025120 001362      MOV      #RBUF,FMTDPB+6 ;SET UP BUFFER ADRS
1706 004706 005037 001364                CLR      FMTDPB+10   ;CLEAR THE SECTOR & TRACK ADDRESS FIELD
1707 004712 013737 001224 001366      MOV      BEGCVL,FMTDPB+12 ;SETUP THE CYLINDER FIELD
1708 004720 112737 000173 001356      MOV      #RDHD,FMTDPB+2 ;SET UP READ HEADER & DATA COMMAND
1709 004726 012737 004726 001110      MOV      #,SLPERR    ;SETUP LOOP ON ERROR ADDRESS
1710 004734 012706 001100                MOV      #STACK,SP   ;LOAD STACK POINTER
1711 004740 004037 013034      1$:     JSR      RO,RPO4  ;GO READ HEADER
  
```

```

1712 004744 001354          FMTDPB          ; ADRS OF PARAMETER TBL
1713 004746 000774          BR              1$          ; WAIT IF QUEUE IS FULL
1714 004750 005737 001372 2$: TST          FMTDPB+16      ; WAIT FOR COMMAND TO COMPLETE
1715 004754 001775          BEQ              2$          ; BRANCH IF NOT DONE
1716 004756 100024          BPL              4$          ; BR IF NOT ERROR
1717 004760 012737 005042 001160 MOV          #5$, $ESCAPE      ; ESCAPE TO 5$ ON ERROR
1718 004766 004737 005234 JSR          PC, $RINDX      ; SEE WHICH ERROR
1719 004772 104010          ERROR          10          ; DRIVE OFFLINE
1720 004774 104011          ERROR          11          ; PERSISTENT DRIVE UNSAFE ERROR
1721 004776 104012          ERROR          12          ; UNCORRECTABLE MASSBUS PARITY ERROR
1722 005000 104013          ERROR          13          ; SOFTWARE TIMEOUT
1723 005002 104014          ERROR          14          ; DRIVE UNSAFE ERROR
1724 005004 032737 177577 001372 BIT          #1<HCE>, FMTDPB+16 ; IS ONLY ERROR A HEADER COMPARE ERROR ?
1725 005012 001401          BEQ              3$          ; BR IF YES
1726 005014 104021          ERROR          21          ; CONTROLLER/DRIVE ERROR VERIFYING HEADERS
1727 005016 005037 001160 3$: CLR          $ESCAPE      ; CLEAR THE ERROR ESCAPE ADDRESS
1728 005022 104022          ERROR          22          ; HEADER COMPARE ERROR VERIFYING HEADERS
1729 005024 004737 005334 JSR          PC, LOP.CK      ; CHECK FOR LOOP ON ERROR
1730 005030 023737 025120 025130 4$: CMP          RBUF, BUFP      ; SEE IF CYL READ EQUALS CYL EXPECTED
1731 005036 001403          BEQ              6$          ; BR IF CYL CORRECT
1732 005040 104023          ERROR          23          ; CYLINDER NOT CORRECT
1733 005042 004737 005334 JSR          PC, LOP.CK      ; CHECK FOR LOOP ON ERROR
1734 005046 023737 001222 001246 6$: CMP          ENDCYL, CYLCK  ; SEE IF LAST CYLINDER
1735 005054 001002          BNE              7$          ; BR IF NOT FINISHED
1736 005056 000137 005100 JMP          $EOP            ; END OF FORMAT
1737 005062 005237 001246 7$: INC          CYLCK          ; INCREMENT CYLINDER ADDRESS BEING CHECKED
1738 005066 004737 005662 JSR          PC, UPDCY      ; SET UP FOR NEXT CYL
1739 005072 005237 001366 INC          FMTDPB+12      ; ADVANCE TO NEXT CYL #
1740 005076 000720          BR              1$          ; GO READ NEXT HEADER

```

.SBTTL END OF PASS ROUTINE

```

;*****
; INCREMENT THE PASS NUMBER ($PASS)
; IF THERES A MONITOR GO TO IT
; IF THERE ISN'T JUMP TO DONE

```

```

1749 005100          $EOP:
1750 005100 005737 001220          TST          MODE          ; 'FORMAT' OR 'CHECK' MODE ?
1751 005104 001403          BEQ              3$          ; BR IF 'CHECK'
1752 005106 104401 021764          TYPE          MFCMPT      ; 'FORMAT COMPLETE'
1753 005112 000402          BR              4$          ; FINISH THE END MESSAGE
1754 005114 104401 022011 3$: TYPE          , MCCMPT      ; 'CHECK COMPLETE'
1755 005120 104401 022035 4$: TYPE          , NUMERR      ; 'TOTAL ERRORS DETECTED: '
1756 005124 013746 001'12          MOV          $ERTTL, -(SP)  ; SAVE $ERTTL FOR TYPEOUT
1757 005130 104405          TYPDS          ; GO TYPE--DECIMAL ASCII WITH SIGN
1758 005132 104401 001107          TYPE          $CRLF      ; CR-LF
1759 005136 005237 001100          INC          $PASS        ; INCREMENT THE PASS NUMBER
1760 005142 042737 100000 001100 BIC          #100000, $PASS  ; DON'T ALLOW A NEG. NUMBER
1761 005150 005327          DEC          (PC)+        ; LOOP?
1762 005152 000001          $EOPCT: .WORD          1
1763 005154 003013          BGT          $DOAGN        ; YES
1764 005156 012737          MOV          (PC)+, 2(PC)+ ; RESTORE COUNTER
1765 005160 000001          $ENDCT: .WORD          1
1766 005162 005152          $EOPCT
1767 005164 013700 000042          $GET42: MOV          2#42, R0 ; GET MONITOR ADDRESS

```

```

1768 005170 001405          BEQ      $DOAGN          ;; BRANCH IF NO MONITOR
1769 005172 000005          RESET          ;; CLEAR THE WORLD
1770 005174 004710 SENDAD: JSR      PC,(R0)  ;; GO TO MONITOR
1771 005176 000240          NOP           ;; SAVE ROOM
1772 005200 000240          NOP           ;; FOR
1773 005202 000240          NOP           ;; ACT11
1774 005204          SDOAGN:          ;;
1775 005204 000137          JMP      @PC+          ;; RETURN
1776 005206 005210 SRTNAD: .WORD  DONE
1777 005210 032777 000001 173722 DONE:  BIT      #SW0,@SWR  ;; SEE IF SWR 0 SET
1778 005216 001002          BNE     1$          ;; BR IF SET
1779 005220 000137 003116          JMP      MO          ;; ASK FOR NEXT DRIVE
1780 005224 012706 001100          MOV     #STACK,SP  ;; RESET STACK POINTER
1781 005230 000137 003526          JMP      M$          ;; DO THE FORMAT OR CHECK AGAIN
1782
1783 ;*****
1784
1785 .SBTTL  SUPPORT SUBROUTINES
1786
1787 ;*****
1788
1789 ;THIS ROUTINE DETERMINES THE ERROR TYPE
1790
1791 005234 010146 ERINDY: MOV     R1,-(SP)  ;; STORE R1
1792 005236 005001          CLR     R1          ;; CLEAR R1
1793 005240 033727 001372          BIT     FMTDPB+16,(PC)+ ;; CHECK ERROR TYPE
1794 005244 060006          .WORD  BIT14!BIT13!BIT2!BIT1 ;; DRIVE OFFLINE
1795 005246 001025          BNE     5$          ;; BR IF OFFLINE
1796 005250 033727 001372          BIT     FMTDPB+16,(PC)+ ;; CHECK ERROR TYPE
1797 005254 010000          .WORD  BIT12          ;; DRIVE PERSISTENTLY UNSAFE
1798 005256 001020          BNE     4$          ;; BR IF UNSAFE
1799 005260 033727 001372          BIT     FMTDPB+16,(PC)+ ;; CHECK ERROR TYPE
1800 005264 006000          .WORD  BIT11!BIT10    ;; UNCORRECTABLE MASSBUS PARITY ERROR ?
1801 005266 001013          BNE     3$          ;; BR IF PARITY ERROR
1802 005270 033727 001372          BIT     FMTDPB+16,(PC)+ ;; CHECK ERROR TYPE
1803 005274 001400          .WORD  BIT09!BIT08    ;; SOFTWARE TIMEOUT ?
1804 005276 001006          BNE     2$          ;; BR IF YES
1805 005300 033727 001372          BIT     FMTDPB+16,(PC)+ ;; CHECK ERROR TYPE
1806 005304 000020          .WORD  BIT04          ;; DRIVE UNSAFE ERROR ?
1807 005306 001001          BNE     1$          ;; BR IF YES
1808 005310 005201          INC     R1          ;; INCREMENT THE RETURN INDEX
1809 005312 005201          1$: INC     R1          ;; INCREMENT THE RETURN INDEX
1810 005314 005201          2$: INC     R1          ;; INCREMENT THE RETURN INDEX
1811 005316 005201          3$: INC     R1          ;; INCREMENT THE RETURN INDEX
1812 005320 005201          4$: INC     R1          ;; INCREMENT THE RETURN INDEX
1813 005322 006301          5$: ASL     R1          ;; DOUBLE THE INCREMENT
1814 005324 060166 000002          ADD     R1,2(SP)    ;; DEVELOP THE RETURN ADDRESS
1815 005330 012601          MOV     (SP)+,R1    ;; RESTORE R1
1816 005332 000207          RTS     PC          ;; RETURN
1817
1818 ;ROUTINE TO CHECK FOR LOOP ON ERROR
1819
1820 005334 032777 001000 173576 LOP.CK: BIT     #SW09,@SWR  ;; LOOP ON ERROR ?
1821 005342 001402          BEQ     1$          ;; BR IF NOT
1822 005344 000177 173540          JMP     @SLPERF     ;; GO TO THE LOOP ON ERROR ADDRESS
1823 005350 005037 001160          1$: CLR     $ESCAPE  ;; CLEAR THE ERROR ESCAPE ADDRESS

```

K03

CZRJB80, RPO4/5-6 FMTR MACY11 30(1046) 07-NOV-77 10:07 PAGE 37
 CZRJB8.P11 04-NOV-77 12:54 SUPPORT SUBROUTINES

SEQ 0036

```

1824 005354 033727 001372      BIT      FMTDPB+16,(PC)+ ;CHECK FOR 'FATAL' ERROR
1825 005360 072002                .WORD    BIT14:BIT13:BIT12:BIT10:BIT01 ;'FATAL' ERROR BITS
1826 005362 001004                BNE     2$ ;BR IF NOT
1827 005364 032737 004000 001320  BIT     #WLE,RP.REG+RPER1 ;WRITE LOCK ERROR ?
1828 005372 001402                BEQ     3$ ;BR IF NOT
1829 005374 000137 005100      2$:    JMP     SEOP ;TERMINATE THE FORMAT
1830 005400 000207      3$:    RTS     PC ;RETURN
1831
1832 ;THIS ROUTINE SETS UP THE INPUT TABLE FOR THE DRIVER ROUTINE
1833
1834 005402 113737 001214 001354  SETTBL: MOVB   DRIVE,FMTDPB ;SET UP DRIVE #
1835 005410 105037 001357                CLRB   FMTDPB+3 ;CLEAR HIGH ORDER ADRS BITS
1836 005414 013737 001260 001360      MOV    MWC,FMTDPB+4 ;LOAD UP WORD COUNT
1837 005422 012737 025130 001362      MOV    #BUFF,FMTDPB+6 ;LOAD UP CURRENT ADRS
1838 005430 105037 001364                CLRB   FMTDPB+10 ;SET SECTOR TO ZERO
1839 005434 113737 001230 001365      MOVB   BEGTRK,FMTDPB+11 ;SET UP STARTING TRK ADRS
1840 005442 013737 001224 001366      MOV    BEGCYL,FMTDPB+12 ;SET UP STARTING CYL
1841 005450 005037 001372                CLR    FMTDPB+16 ;CLEAR RPO4 STATUS
1842 005454 013737 001230 001236      MOV    BEGTRK,TRKCNT ;SET UP PARTIAL CYL TRACK COUNT
1843 005462 013737 001232 001234      MOV    TTRKS,↑TRKSC ;SET UP TOTAL TRACKS COUNTER
1844 005470 000207                RTS     PC ;RETURN FROM SETUP
1845
1846 ;THIS ROUTINE CONTROLS THE DISK ADDRESSING AND TOTAL TRK COUNT
1847 ;IT IS ENTERED AFTER EVERY TRK OPERATION
1848
1849 005472 005337 001234      TRKTST: DEC    TTRKSC ;SEE IF LAST TRACK
1850 005476 001425                BEQ     3$ ;BRANCH IF LAST TRACK
1851 005500 022737 000022 001236      CMP    #18.,TRKCNT ;IS THIS THE LAST TRACK IN THE CYLINDER ?
1852 005506 001407                BEQ     1$ ;BRANCH IF SO
1853 005510 005237 001236      INC    TRKCNT ;COUNT UP TRACK WITHIN CYLINDER
1854 005514 105237 001365      INCB   FMTDPB+11 ;INCREMENT TRACK NUMBER
1855 005520 004737 005712      JSR    PC,UPDATK ;UPDATE TRACK ADDRESS IN BUFFER
1856 005524 000410                BR     2$ ;EXIT
1857 005526 005037 001236      1$:    CLR    TRKCNT ;CLEAR TRACK COUNT FOR NEXT CYLINDER
1858 005532 105037 001365      CLRB   FMTDPB+11 ;RESET TRACK ADDRESS TO 0
1859 005536 005237 001366      INC    FMTDPB+12 ;UPDATE CYLINDER NUMBER
1860 005542 004737 005662      JSR    PC,UPDACY ;UPDATE CYLINDER NUMBER IN BUFFER
1861 005546 062716 000002      2$:    ADD    #2,(SP) ;ADD TWO TO RETURN ADRS
1862 005552 000207      3$:    RTS     PC ;RETURN
1863
1864 ;THIS ROUTINE SETS UP WC & BA FOR REMAINING SECTORS
1865 ;AFTER A WRITE CHECK ERROR
1866
1867 005554 013737 001254 001360  SCAWC: MOV    SAVWC,FMTDPB+4 ;SET UP WC FOR REMAINING SECTORS
1868 005562 062737 001010 001362      ADD    #520,FMTDPB+6 ;SET UP BA FOR REMAINING SECTORS
1869 005570 005237 001364      INC    FMTDPB+10 ;ADVANCE TO NEXT SECTOR IN TBL
1870 005574 005037 001216      CLR    SOFSW ;RESET RETRY COUNTER
1871 005600 000207                RTS     PC ;RETURN TO COMPLETE TRK FORMAT
1872
1873 ;THIS ROUTINE SETS UP THE CYLINDER ADRS, FORMAT BIT, TRACK AND
1874 ;SECTOR ADRS IN MEMORY WITH THE STARTING CYL - TRK INFORMATION
1875
1876
1877 005602 013701 001266      SETHDR: MOV    MAXSEC,R1 ;SET UP SECTOR COUNT
1878 005606 012700 025130      MOV    #BUFF,R0 ;SET UP HEADER POINTER IN R0
1879 005612 013702 001224      MOV    BEGCYL,R2 ;PUT STARTING CYL # IN R2

```

```

1880 005616 013703 001230      MOV      BEGTRK,R3      ;PUT STARTING TRK # IN R3
1881 005622 000303      SWAB     R3            ;JUSTIFY TRACK ADRS
1882 005624 005737 001264      TST      SEC20        ;SEE IF 20 OR 22 SECTOR MODE
1883 005630 001402      BEQ      1$           ;BR IF 20 SECTOR MODE
1884 005632 052702 010000      BIS      #10000,R2    ;SET THE 22 SECTOR FORMAT BIT
1885 005636 010220      1$: MOV     R2,(R0)+    ;WRITE IN HEADER AREA OF CORE THE CYL ADRS
1886 005640 010320      MOV     R3,(R0)+    ;WRITE IN HEADER AREA OF CORE THE TRK ADRS
1887 005642 005020      CLR     (R0)+       ;CLR 1ST KEYWORD
1888 005644 005010      CLR     (R0)        ;CLR 2ND KEYWORD
1889 005646 062700 001002      ADD     #514.,R0     ;SET UP FOR NEXT HEADER
1890 005652 005203      INC     R3          ;UPDATE SECTOR ADRS FOR NEXT HEADER
1891 005654 005301      DEC     R1          ;MAINTAIN COUNT OF SECTORS
1892 005656 002367      BGE     1$          ;BRANCH IF NOT LAST SECTOR
1893 005660 000207      RTS      PC         ;EXIT - HEADERS ARE LOADED INTO CORE
1894
1895 ;THIS ROUTINE UPDATES THE CYLINDER ADRS OF THE HEADER WORDS IN CORE
1896
1897 005662 013701 001266      UPDACY: MOV     MAXSEC,R1 ;SET UP SECTOR COUNT
1898 005666 012700 025130      MOV     #BUFP,R0     ;SET UP HEADER POINTER IN R0
1899 005672 005220      1$: INC     (R0)+     ;INCREMENT FOR NEXT CYLINDER
1900 005674 042710 177400      BIC     #177400,(R0) ;RESET TRK ADRS TO 0
1901 005700 062700 001006      ADD     #518.,R0     ;SET UP FOR NEXT HEADER
1902 005704 005301      DEC     R1          ;COUNT SECTORS
1903 005706 002371      BGE     1$          ;BRANCH IF NOT LAST SECTOR
1904 005710 000207      RTS      PC         ;EXIT
1905
1906 ;THIS ROUTINE UPDATES THE TRACK ADRS OF THE HEADER WORDS IN CORE
1907
1908 005712 013701 001266      UPDATK: MOV     MAXSEC,R1 ;SET UP SECTOR COUNT
1909 005716 012700 025130      MOV     #BUFP,R0     ;SET UP HEADER POINTER IN R0
1910 005722 005720      TST     (R0)+       ;POINT HEADER POINTER TO TRK - SEC ADRS
1911 005724 062710 000400      1$: ADD     #400,(R0)  ;INDEX TRK ADRS
1912 005730 062700 001010      ADD     #520.,R0     ;SET UP FOR NEXT HEADER
1913 005734 005301      DEC     R1          ;COUNT SECTORS
1914 005736 002372      BGE     1$          ;BRANCH IF NOT LAST SECTOR
1915 005740 000207      RTS      PC         ;EXIT
1916
1917 ;ROUTINE TO CHECK FOR KW11-L OR KW11-P CLOCKS
1918
1919 005742 012737 006020 000004      ST.CLK: MOV     #STCLK1,@#ERRVEC ;SET UP VECTOR FOR P CLK
1920 005750 005037 000006      CLR     @#ERRVEC+2   ;NEW PSW
1921 005754 005777 173216      TST     @SLKCSR      ;CHECK FOR KW11-P
1922 005760 013746 001202      MOV     $LPVEC,-(SP) ;VECTOR ADDRESS
1923 005764 012776 006104 000000      MOV     #CLOCK,@(SP) ;SET UP KW11-P VECTOR
1924 005772 062716 000002      ADD     #2,(SP)      ;POINT TO PSW
1925 005776 012736 000300      MOV     #PR6,@(SP)+  ;PSW - PRI 6
1926 006002 012777 177777 173170      MOV     #-1,@SLKCSB ;LOAD COUNTER BUFFER
1927 006010 012777 000135 173160      MOV     #135,@SLKCSR ;SET CLK - CNT UP
1928 006016 000426      BR      STCLK3
1929 006020 062706 000004      STCLK1: ADD     #4,SP  ;RESTORE THE STACK POINTER
1930 006024 012737 006070 000004      MOV     #STCLK2,@#ERRVEC ;CHANGE ERROR VECTOR
1931 006032 005777 173150      TST     @SLKS        ;LOOK FOR KW11-L
1932 006036 013746 001210      MOV     $LLVEC,-(SP) ;KW11-L VECTOR ADDRESS
1933 006042 012776 006104 000000      MOV     #CLOCK,@(SP) ;SET UP KW11-L VECTOR
1934 006050 062716 000002      ADD     #2,(SP)      ;INCREMENT VECTOR ADDRESS
1935 006054 012736 000300      MOV     #PR6,@(SP)+  ;PSW - PRI 6

```

```

1936 006060 012777 000100 173120      MOV      #100,2$CLKS      ;SET KW11-L INTERRUPT ENABLE
1937 006066 000402                BR        STCLK3
1938 006070 062706 000004                STCLK2: ADD     #4,SP      ;RESTORE THE STACK POINTER
1939 006074 012737 000006 000004  STCLK3: MOV     #6,2$ERRVEC ;RESTORE THE ERROR VECTOR
1940 006102 000207                RTS        PC
1941
1942                ;THIS CODE SERVICES A CLOCK INTERRUPT EVERY 16MS
1943
1944 006104 012746 000020  CLOCK: MOV     #16,-(SP)   ;PUT MILLISECONDS ON THE STACK
1945 006110 004737 016466                JSR      PC,RPTMR      ;GO REPORT TIME
1946 006114 000002                RTI                ;RETURN AND CONTINUE
1947
1948                ;ROUTINE TO INTERCEPT 'CONTROL C' TYPED DURING PARAMETER ENTRY TIME
1949
1950 006116 012706 001100  CENTER: MOV     #STACK,SP  ;INITIALIZE THE STACK
1951 006122 005737 012160  1$:  TST      TRANSW      ;ALL ACTIVITY STOPPED ?
1952 006126 001375                BNE     1$           ;BR IF NOT
1953 006130 000005                RESET    ;CLEAR THE BUS
1954 006132 000137 003062                JMP      M1B        ;START AGAIN WITH DRIVE SELECTION
1955
1956                ;ROUTINE TO TYPE THE PRESENT DISK ADDRESS. ENTERED BY TYPING 'CONTROL C'
1957
1958 006136 005037 177776  TYPADR: CLR     PSW        ;SET PROCESSOR TO PRIORITY 0
1959 006142 012737 006116 001300  MOV     #CENTER,CNTLC    ;CHANGE 'CONTROL C' RETURN ADDRESS
1960 006150 104401 022065                TYPE    ,ADDRIS        ;'PRESENT ADDRESS IS: '
1961 006154 104401 020675                TYPE    'C'           ;'C'
1962 006160 013746 001366  MOV     FMTDPB+12,-(SP)  ;PUT THE CYLINDER ADDRESS ON THE STACK
1963 006164 004737 011470                JSR     PC,$S82D       ;CONVERT IT TO DECIMAL
1964 006170 004737 011430                JSR     PC,$SUPRS      ;TYPE IT
1965 006174 104401 020705                TYPE    ,LINSF        ;SPACES
1966 006200 104401 020677                TYPE    'T'           ;'T'
1967 006204 005046                CLR     -(SP)         ;CLEAR THE STACK
1968 006206 113716 001365  MOV     FMTDPB+11,(SP)  ;PUT THE TRACK ADDRESS ON THE STACK
1969 006212 004737 011470                JSR     PC,$S82D       ;CONVERT IT TO DECIMAL
1970 006216 004737 011430                JSR     PC,$SUPRS      ;TYPE IT
1971 006222 104401 001167                TYPE    $CRLF         ;CR-LF
1972 006226 012737 006136 001300  MOV     #TYPADR,CNTLC   ;RESTORE ENTRANC. TO THIS ROUTINE
1973 006234 000002                RTI                ;RETURN
1974
1975                ;PARAMETER ENTRY ROUTINE
1976                ;CALL
1977                ;
1978                ;      MOV     #ADR,R3      ;PARAMETER TABLE ADDRESS
1979                ;      JSR     PC,PARENT   ;GET THE PARAMETERS
1980
1980 006236 010346  PARENT: MOV     R3,-(SP)   ;SAVE R3
1981 006240 012703 001374  MOV     #TABLE,R3      ;PARAMETER TABLE ADDRESS
1982 006244 012337 006254  1$:  MOV     (R3)+,3$     ;ADDRESS OF PARAMETER NAME
1983 006250 001436                BEQ     9$           ;BR IF AT END OF TABLE
1984 006252 104401                TYPE    ;TYPE THE PARAMETER NAME
1985 006254 000000  3$:  .WORD  0           ;ADDRESS OF PARAMETER NAME TEXT
1986 006256 012302                MOV     (R3)+,R2     ;MAXIMUM PARAMETER VALUE
1987 006260 012305                MOV     (R3)+,R5     ;ADDRESS OF PARAMETER
1988 006262 011546                MOV     (R5),-(SP)   ;CURRENT VALUE OF PARAMETER
1989 006264 104405                TYPDS   ;TYPE THE CURRENT VALUE OF THE PARAMETER
1990 006266 104401 020671                TYPE    ,SLASH       ;'/'
1991 006272 104411                PDLIN   ;READ THE KEYBOARD

```

```

1992 006274 012601      MOV      (SP)+,R1      ;INPUT ASCII STRING ADDRESS
1993 006276 004537 006500 JSR      R5,CK.DIG    ;CHECK THE DIGIT(S)
1994 006302 006244      1$      ;CARRIAGE RETURN ONLY ENTERED
1995 006304 006346      9$      ;PERIOD ONLY ENTERED
1996 006306 006322      6$      ;ILLEGAL INPUT
1997 006310 006316      5$      ;TERMINATED WITH A CARRIAGE RETURN
1998 006312 006322      6$      ;TERMINATED WITH A " "
1999 006314 006334      7$      ;TERMINATED WITH A "' '
2000 006316 010215      5$: MOV      R2,(R5)    ;MOVE NEW VALUE TO PARAMETER LOCATION
2001 006320 000751      BR       1$          ;GET MORE PARAMETERS
2002 006322 104401 021436      6$: TYPE     BADENT   ;'BAD ENTRY'
2003 006326 162703 000006      SUB      #6,R3      ;DECREMENT THE TABLE POINTER
2004 006332 000744      BR       1$          ;TRY AGAIN
2005 006334 010215      7$: MOV      R2,(R5)    ;NEW VALUE
2006 006336 000403      BR       9$          ;EXIT
2007 006340 012703 001374      8$: MOV      #TABLE,R3 ;RELOAD THE PARAMETER TABLE ADDRESS
2008 006344 000737      BR       1$          ;TRY AGAIN
2009 006346 012503      9$: MOV      (SP)+,R3   ;RESTORE R3
2010 006350 000207      RTS      PC          ;RETURN
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022 006352 121127 000060      CK.OCT: CMPB     (R1),#'0 ;LESS THAN ZERO?
2023 006356 103407      BLO     1$          ;YES -- BRANCH
2024 006360 121127 000067      CMPB     (R1),#'7     ;GREATER THAN SEVEN?
2025 006364 101004      BHI     1$          ;YES -- BRANCH
2026 006366 111102      MOVB     (R1),R2      ;GET THE CHARACTER
2027 006370 042702 177770      BIC      #107,R2     ;STRIP AWAY THE ASCII
2028 006374 005725      TST      (R5)+      ;ADJUST FOR RETURN
2029 006376 000205      1$: RTS      R5      ;RETURN
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040 006400 121127 000060      CK.DEC: CMPB     (R1),#'0 ;LESS THAN ZERO?
2041 006404 103407      BLO     1$          ;YES -- BRANCH
2042 006406 121127 000071      CMPB     (R1),#'9     ;GREATER THAN NINE?
2043 006412 101004      BHI     1$          ;YES -- BRANCH
2044 006414 111102      MOVB     (R1),R2      ;GET THE CHARACTER
2045 006416 042702 000060      BIC      #10,R2     ;STRIP AWAY THE ASCII
2046 006422 005725      TST      (R5)+      ;ADJUST FOR RETURN
2047 006424 000205      1$: RTS      R5      ;RETURN

```

2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062 006436 105711
2063 006430 001417
2064 006432 121127 000054
2065 006436 001413
2066 006440 121127 000056
2067 006444 001407
2068 006446 004537 006400
2069 006452 000410
2070 006454 004537 006352
2071 006460 005725
2072 006462 005725
2073 006464 005725
2074 006466 005725
2075 006470 005725
2076 006472 005201
2077 006474 011505
2078 006476 000205
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093 006500 010446
2094 006502 010346
2095 006504 010246
2096 006506 005002
2097 006510 005003
2098 006512 005004
2099 006514 004537 006426
2100 006520 006614
2101 006522 006622
2102 006524 006614
2103 006526 006616

```

: THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO
: DETERMINE WHAT IT IS.
CALL
      MOV      #ADR,R1      ; ADDRESS OF ASCII CHARACTER
      JSR     R5,CK.CHR    ; CHECK CHARACTER
      RETURN  ADR1         ; UNKNOWN CHARACTER
      RETURN  ADR2         ; CARRIAGE RETURN * (R1)=ADR+1
      RETURN  ADR3         ; COMMA * (R1)=ADR+1
      RETURN  ADR4         ; PERIOD * (R1)=ADR+1
      RETURN  ADR5         ; DIGIT BETWEEN 0 AND 7.
      RETURN  ADR6         ; DIGIT BETWEEN 8 AND 9.
                          ; R2 = DIGIT * (R1)=ADR+1
CK.CHR: TSTB   (R1)        ; "CARRIAGE RETURN"?
        BEQ   3$          ; YES -- BRANCH
        CMPB  (R1),#' ,  ; "COMMA"?
        BEQ   2$          ; YES -- BRANCH
        CMPB  (R1),#'.  ; "PERIOD"?
        BEQ   1$          ; YES -- BRANCH
        JSP   R5,CK.DEC  ; "DIGIT"?
        BR    4$          ; NO -- BRANCH
        JSR   R5,CK.OCT  ; OCTAL ?
        TST   (R5)+       ; DIGIT BETWEEN 8-9
        TST   (R5)+       ; DIGIT BETWEEN 0-7
1$:     TST   (R5)+       ; PERIOD
2$:     TST   (R5)+       ; COMMA
3$:     TST   (R5)+       ; CARRIAGE RETURN
4$:     INC   R1          ; MOVE POINTER TO NEXT CHARACTER
        MOV   (R5),R5     ; UNKNOWN CHARACTER
        RTS   R5         ; RETURN

```

```

: THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL
: CHARACTERS AND FORMS A DECIMAL VALUE BINARY NUMBER IN R2.
CALL
      MOV      #ADR,R1      ; ADDRESS OF ASCII STRING
      MOV      #NUM,R2     ; MAX. MAGNITUDE OF INPUT NUMBER
      JSR     R5,CK.DIG    ; CHECK DIGITS
      RETURN  ADR1         ; "CR" ONLY ENTERED -- R2=0
      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
CK.DIG: MOV     R4,-(SP)    ; SAVE R4
        MOV     R3,-(SP)    ; SAVE R3
        MOV     R2,-(SP)    ; SAVE THE MAX. SIZE ON THE STACK
        CLR     R2         ; START WITH 0
        CLR     R3
        CLR     R4
        JSR    R5,CK.CHR  ; CHECK ONE CHARACTER
        BEQ    6$         ; ILLEGAL CHARACTER
        BEQ    9$         ; CARRIAGE RETURN
        BEQ    6$         ; " , "
        BEQ    6$         ; " ."
        BEQ    7$         ; " "

```

```

2104 006530 006534 1$ ;DIGIT 0-7
2105 006532 006534 1$ ;DIGIT 8-9
2106 006534 062705 000004 1$: ADD #4,R5 ;STEP RETURN POINTER PAST "CR" & "PERIOD" RETURNS
2107 006540 006303 2$: ASL R3 ;INPUT NUMBER #2
2108 006542 010346 ASL R3, -(SP) ;SAVE #2
2109 006544 006303 ASL R3 ;#4
2110 006546 006303 ASL R3 ;#8
2111 006550 062603 ADD (SP)+,R3 ;(#2)+(#8) = #10
2112 006552 060203 ADD R2,R3 ;UPDATE THE INPUT NUMBER
2113 006554 004537 006426 JSR R5,CK.CHR ;CHECK ONE CHARACTER
2114 006560 006620 B$ ;ILLEGAL CHARACTER
2115 006562 006604 S$ ;CARRIAGE RETURN
2116 006564 006602 4$ ;" "
2117 006566 006574 3$ ;" "
2118 006570 006540 2$ ;DIGIT 0-7
2119 006572 006540 2$ ;DIGIT 8-9
2120 006574 105711 3$: TSTB (R1) ;DOES A "CR" FOLLOW THE "PERIOD"
2121 006576 001010 BNE B$ ;BR IF NOT
2122 006600 005724 TST (R4)+ ;INCREMENT THE RETURN
2123 006602 005724 4$: TST (R4)+ ;INCREMENT THE RETURN
2124 006604 005724 5$: TST (R4)+ ;INCREMENT THE RETURN
2125 006606 020316 CMP R3,(SP) ;CHECK THE MAGNITUDE OF THE NUMBER
2126 006610 101004 BHI 9$ ;BR IF ENTERED NUMBER TOO LARGE
2127 006612 000402 BR 6$ ;BYPASS INCREMENT
2128 006614 005725 6$: TST (R5)+ ;INCREMENT RETURN PAST INVALID RETURN
2129 006616 005725 7$: TST (R5)+ ;INCREMENT RETURN
2130 006620 060405 8$: ADD R4,R5 ;SETUP RETURN POINTER
2131 006622 010302 9$: MOV R3,R2 ;ENTERED VALUE
2132 006624 005726 TST (SP)+ ;CLEAN MAX. SIZE OFF OF STACK
2133 006626 012603 MOV (SP)+,R3 ;RESTORE R3
2134 006630 012604 MOV (SP)+,R4 ;RESTORE R4
2135 006632 011505 MOV (R5),R5 ;GET RETURN ADDRESS
2136 006634 000205 RTS R5 ;RETURN

```

.SBTTL MACRO ROUTINES

.SBTTL ERROR HANDLER ROUTINE

```

*****
*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT.
*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
*AND GO TO TYPERR ON ERROR
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW15=1 HALT ON ERROR
*SW13=1 INHIBIT ERROR TYPEOUTS
*SW10=1 BELL ON ERROR
*SW09=1 LOOP ON ERROR
*CALL
* ERROR N ;:ERROR=EMT AND N=ERROR ITEM NUMBER

```

```

$ERROR:
        CKSWR ;:TEST FOR CHANGE IN SOFT-SWR
        MOV R1,DDRIVE ;:DRIVE ADDRESS IF DRIVE ERROR CALL
        MOV R3,ATTN ;:ATTENTION REGISTER CONTENTS
        MOV FMTDPB+12,DS.CYL ;:CURRENT CYLINDER ADDRESS

```

```

2155 006636 104407
2156 006636 010137 001274
2157 006640 010337 001276
2158 006644 013737 001366 001270
2159 006650 013737 001366 001270

```

```

2160 006656 113737 001365 001272      MOVB   FMDPB+11,DS,TRK      ;REQUESTED TRACK ADDRESS
2161 006664 005737 001310              TST   RP,REG+RPBA          ;NON-ZERO BUFFER ADDRESS ?
2162 006670 001406              BEQ   7$                   ;BR IF NO BUFFER ADDRESS
2163 006672 013746 001310      MOV   RP,REG+RPBA,-(SP)    ;BUFFER ADDRESS
2164 006676 162716 000002      SUB   #2,(SP)              ;DECREMENT THE ADDRESS
2165 006702 013637 001124      MOV   2(SP)+,$GDDAT       ;GET THE BUFFER WORD WHICH DIDN'T COMPARE
2166 006706 105237 001103      7$:  INCB  $ERFLG            ;SET THE ERROR FLAG
2167 006712 001775              BEQ   7$                   ;DON'T LET THE FLAG GO TO ZERO
2168 006714 013777 001102 172220      MOV   $STNM,$DISPLAY      ;DISPLAY TEST NUMBER AND ERROR FLAG
2169 006722 032777 002000 172210      BIT   #BIT10,$SWR         ;BELL ON ERROR?
2170 006730 001402              BEQ   1$                   ;NO - SKIP
2171 006732 104401 001162              TYPE  $BELL                ;RING BELL
2172 006736 005237 001112      1$:  INC   $ERTTL           ;COUNT THE NUMBER OF ERRORS
2173 006742 011637 001116      MOV   (SP),$ERRPC         ;GET ADDRESS OF ERROR INSTRUCTION
2174 006746 162737 000002 001116      SUB   #2,$ERRPC
2175 006754 117737 172136 001114      MOVB  $ERRPC,$ITEMB       ;STRIP AND SAVE THE ERROR ITEM CODE
2176 006762 032777 020000 172150      BIT   #BIT13,$SWR         ;SKIP TYPEOUT IF SET
2177 006770 001004              BNE   20$                 ;SKIP TYPEOUTS
2178 006772 004737 007044      JSR   PC,TYPERR           ;GO TO USER ERROR ROUTINE
2179 006776 104401 001167              TYPE  , $CRLF
2180 007002      20$:
2181 007002 005777 172132      2$:  TST   $SWR              ;HALT ON ERROR
2182 007006 100002              BPL   3$                   ;SKIP IF CONTINUE
2183 007010 000000              HALT                                ;HALT ON ERROR!
2184 007012 104407              CKSWR                             ;TEST FOR CHANGE IN SOFT-SWR
2185 007014 032777 001000 172116      3$:  BIT   #BIT09,$SWR       ;LOOP ON ERROR SWITCH SET?
2186 007022 001402              BEQ   4$                   ;BR IF NO
2187 007024 013716 001110      MOV   $LPERR,(SP)         ;FUDGE RETURN FOR LOOPING
2188 007030 005737 001160      4$:  TST   $ESCAPE           ;CHECK FOR AN ESCAPE ADDRESS
2189 007034 001402              BEQ   5$                   ;BR IF NONE
2190 007036 013716 001160      MOV   $ESCAPE,(SP)       ;FUDGE RETURN ADDRESS FOR ESCAPE
2191 007042      5$:
2192 007042 000002              RTI                               ;RETURN
2193
2194
2195      ;THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE
2196      ;WHICH ERROR IS TO BE REPORTED, IT THEN OBTAINS FROM THE "ERROR
2197      ;TABLE" ($ERRTB) AND REPORTS THE APPROPRIATE INFORMATION
2198      ;CONCERNING THE ERROR.
2199
2200      TYPERR: SAVREG          ;SAVE R0-R5
2201      CLR   R0              ;CLEAR R0 FOR ERROR NUMBER
2202      MOVB  $ITEMB,R0       ;ERROR NUMBER
2203      DEC   R0              ;FORM INDEX FOR ERROR TABLE
2204      ASL   R0
2205      ASL   R0
2206      ASL   R0
2207      1$:  ADD   # $ERRTB,R0    ;FORM ADDRESS
2208      MOV   (R0)+,2$       ;GET ERROR MESSAGE (EM) POINTER
2209      BEQ   3$             ;BRANCH IF THERE ISN'T ONE
2210      TYPE  , $CRLF        ;"CARRIAGE RETURN - LINE FEED
2211      TYPE
2212      2$:  .WORD 0           ;"EM" POINTER GOES HERE
2213      3$:  MOV   (R0)+,4$    ;PICK UP DATA HEADER (DH) POINTER
2214      BEQ   5$             ;BRANCH IF NONE
2215      TYPE  , $CRLF        ;CARRIAGE RETURN-LINE FEED

```

```

2216 007120 104401
2217 007122 000000
2218 007124 012001
2219 007126 001460
2220 007130 005005
2221 007132 012000
2222 007134 012002
2223 007136 001451
2224 007140 005105
2225 007142 104401 001167
2226 007146 112003
2227 007150 112004
2228 007152 006004
2229 007154 103403
2230 007156 013146
2231 007160 104402
2232 007162 000402
2233 007164
2234 007164 013146
2235 007166 104405
2236 007170 005303
2237 007172 001403
2238 007174 104401 020705
2239 007200 000764
2240 007202 005302
2241 007204 003431
2242 007206 104401 001167
2243 007212 005760 000002
2244 007216 001404
2245 007220 005105
2246 007222 001002
2247 007224 104401 020705
2248 007230 012037 007236
2249 007234 104401
2250 007236 000000
2251 007240 005710
2252 007242 001003
2253 007244 062700 000004
2254 007250 000754
2255 007252 104401 001167
2256 007256 005705
2257 007260 001332
2258 007262 104401 020705
2259 007266 000727
2260 007270 104413
2261 007272 000207

```

```

4$: TYPE
5$: .WORD 0
MOV (R0)+,R1
BEQ 20$
CLR R5
MOV (R0)+,R0
MOV (R0)+,R2
BEQ 17$
COM R5
TYPE $CRLF
MOV (R0)+,R3
MOV (R0)+,R4
ROR R4
BCS 12$
MOV @ (R1)+, -(SP)
TYPOC
BR 13$
10$: MOV @ (R1)+, -(SP)
TYPOC
13$: DEC R3
BEQ 14$
TYPE L INSP
BR 11$
14$: DEC R2
BLE 20$
TYPE $CRLF
TST @ (R0)
BEQ 15$
COM R5
BNE 15$
TYPE L INSP
15$: MOV (R0)+, 16$
TYPE
16$: .WORD 0
TST (R0)
BNE 21$
ADD #4,R0
BR 14$
21$: TYPE $CRLF
TST R5
BNE 10$
TYPE L INSP
BR 10$
20$: RESREG
RTS PC

```

```

;"DH" POINTER GOES HERE
; PICKUP DATA TABLE (DT) POINTER
; BRANCH IF NONE
; SET INDENT SWITCH
; DATA FORMAT (DF) POINTER
; NUMBER OF DH'S TO TYPE
; BRANCH IF DH NUMBER IS 0
; NO INDENT
; CARRIAGE RETURN-LINE FEED
; NUMBER OF DATA WORDS TO TYPE
; AND HOW TO TYPE THEM
; OCTAL OR DECIMAL?
; DECIMAL--BRANCH
; SAVE @ (R1)+ FOR TYPEOUT
; GO TYPE--OCTAL ASCII(ALL DIGITS)

; ; SAVE @ (R1)+ FOR TYPEOUT
; ; GO TYPE--DECIMAL ASCII WITH SIGN
; MORE NUMBERS TO TYPE?
; NO--BRANCH
; YES--TYPE SEPERATORS
; LOOP
; MORE DH'S?
; NO--BRANCH
; YES--START A NEW LINE
; ONLY A 'DH' IN THIS REQUEST ?
; BR IF YES - BYPASS THE INDENT
; INDENT?
; NO--BRANCH
; YES--TYPE SPACES
; GET NEXT DH
; AND TYPE IT
; DH POINTER GOES HERE
; TYPE A 'DT' ?
; BR IF A 'DT'
; INCREMENT THE 'DF' POINTER
; SEE IF END OF 'DF' BLOCK
; CARRIAGE RETURN-LINE FEED
; INDENT?
; NO--BRANCH
; YES--TYPE SPACES
; LOOP
; RESTORE R0-R5
; RETURN

```

.SBTTL TYPE ROUTINE

```

;*****
;ROUTINE TO TYPE ASCII MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
;THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
;NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
;NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
;NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
;

```

```

2272
2273
2274
2275
2276
2277
2278
2279
2280 007274 105737 001157 $TYPE: TSTB $TFPLG ;; IS THERE A TERMINAL?
2281 007300 100002 BPL 1$ ;; BR IF YES
2282 007302 000000 HALT ;; HALT HERE IF NO TERMINAL
2283 007304 000407 BR 3$ ;; LEAVE
2284 007306 010046 1$: MOV RO, -(SP) ;; SAVE RO
2285 007310 017600 000002 MOV 2(SP), RO ;; GET ADDRESS OF ASCIZ STRING
2286 007314 112046 2$: MOVB (RO)+, -(SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
2287 007316 001005 BNE 4$ ;; BR IF IT ISN'T THE TERMINATOR
2288 007320 005726 TST (SP)+ ;; IF TERMINATOR POP IT OFF THE STACK
2289 007322 012600 6$: MOV (SP)+, RO ;; RESTORE RO
2290 007324 062716 3$: ADD #2, (SP) ;; ADJUST RETURN PC
2291 007330 000002 RTI ;; RETURN
2292 007332 122716 4$: CMPB #HT, (SP) ;; BRANCH IF <HT>
2293 007336 001430 BEQ 8$
2294 007340 122716 000200 CMPB #CRLF, (SP) ;; BRANCH IF NOT <CRLF>
2295 007344 001006 BNE 5$
2296 007346 005726 TST (SP)+ ;; POP <CR><LF> EQUIV
2297 007350 104401 TYPE ;; TYPE A CR AND LF
2298 007352 001167 $CRLF
2299 007354 105037 007510 CLRB $CHARCNT ;; CLEAR CHARACTER COUNT
2300 007360 000755 BR 2$ ;; GET NEXT CHARACTER
2301 007362 004737 007444 5$: JSR PC, $TYPEC ;; GO TYPE THIS CHARACTER
2302 007366 123726 001156 6$: CMPB $FILLC, (SP)+ ;; IS IT TIME FOR FILLER CHARS.?
2303 007372 001350 BNE 2$ ;; IF NO GO GET NEXT CHAR.
2304 007374 013746 001154 MOV $NULL, -(SP) ;; GET # OF FILLER CHARS. NEEDED
2305 AND THE NULL CHAR.
2306 007400 105366 000001 7$: DECB 1(SP) ;; DOES A NULL NEED TO BE TYPED?
2307 007404 002770 BLT 6$ ;; BR IF NO--GO POP THE NULL OFF OF STACK
2308 007406 004737 007444 JSR PC, $TYPEC ;; GO TYPE A NULL
2309 007412 105337 007510 DECB $CHARCNT ;; DO NOT COUNT AS A COUNT
2310 007416 000770 BR 7$ ;; LOOP
2311
2312 ;HORIZONTAL TAB PROCESSOR
2313
2314 007420 112716 000040 8$: MOVB #' , (SP) ;; REPLACE TAB WITH SPACE
2315 007424 004737 007444 9$: JSR PC, $TYPEC ;; TYPE A SPACE
2316 007430 132737 000007 007510 BITB #7, $CHARCNT ;; BRANCH IF NOT AT
2317 007436 001372 BNE 9$ ;; TAB STOP
2318 007440 005726 TST (SP)+ ;; POP SPACE OFF STACK
2319 007442 000724 BR 2$ ;; GET NEXT CHARACTER
2320 007444 105777 171500 $TYPEC: TSTB 2$TPS ;; WAIT UNTIL PRINTER IS READY
2321 007450 100375 BPL $TYPEC
2322 007452 116677 000002 171472 MOVB 2(SP), 2$TPB ;; LOAD CHAR TO BE TYPED INTO DATA REG.
2323 007460 122756 000015 000002 CMPB #CR, 2(SP) ;; IS CHARACTER A CARRIAGE RETURN?
2324 007466 001003 BNE 1$ ;; BRANCH IF NO
2325 007470 105037 007510 CLRB $CHARCNT ;; YES--CLEAR CHARACTER COUNT
2326 007474 000406 BR $TYPEC ;; EXIT
2327 007476 122766 000012 000002 1$: CMPB #LF, 2(SP) ;; IS CHARACTER A LINE FEED?

```

2328 007504 001402
2329 007506 105227
2330 007510 000000
2331 007512 000207
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359 007514 017646 000000
2360 007520 116637 000001 007737
2361 007526 112637 007741
2362 007532 062716 000002
2363 007536 000406
2364 007540 112737 000001 007737
2365 007546 112737 000006 007741
2366 007554 112737 000005 007736
2367 007562 010346
2368 007564 010446
2369 007566 010546
2370 007570 113704 007741
2371 007574 005404
2372 007576 062704 000006
2373 007602 110437 007740
2374 007606 113704 007737
2375 007612 016605 000012
2376 007616 005003
2377 007620 006105 1\$:
2378 007622 000404 BR
2379 007624 006105 2\$:
2380 007626 006105 ROL
2381 007630 006105 ROL
2382 007632 010503 MOV R5,R3
2383 007634 006103 3\$:
ROL R3

```
BEQ $TYPEX ;: BRANCH IF YES
INCB (PC)+ ;: COUNT THE CHARACTER
$CHARCNT: WORD 0 ;: CHARACTER COUNT STORAGE
$TYPEX: RTS PC
```

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```
*****
; THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
; OCTAL (ASCII) NUMBER AND TYPE IT.
; $TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
; CALL:
; MOV NUM, -(SP) ;: NUMBER TO BE TYPED
; TYPOS ;: CALL FOR TYPEOUT
; .BYTE N ;: N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
; .BYTE M ;: M=1 OR 0
; ;: I=TYPE LEADING ZEROS
; ;: O=SUPPRESS LEADING ZEROS
; $TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
; $TYPOS OR $TYPOC
; CALL:
; MOV NUM, -(SP) ;: NUMBER TO BE TYPED
; TYPON ;: CALL FOR TYPEOUT
; $TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
; CALL:
; MOV NUM, -(SP) ;: NUMBER TO BE TYPED
; TYPOC ;: CALL FOR TYPEOUT
; $TYPOS: MOV @ (SP), -(SP) ;: PICKUP THE MODE
; MOVB 1 (SP), $OFILL ;: LOAD ZERO FILL SWITCH
; MOVB (SP)+, $OMODE+1 ;: NUMBER OF DIGITS TO TYPE
; ADD #2, (SP) ;: ADJUST RETURN ADDRESS
; BR $TYPON
; $TYPOC: MOVB #1, $OFILL ;: SET THE ZERO FILL SWITCH
; MOVB #6, $OMODE+1 ;: SET FOR SIX(6) DIGITS
; $TYPON: MOVB #5, $OCNT ;: SET THE ITERATION COUNT
; MOV R3, -(SP) ;: SAVE R3
; MOV R4, -(SP) ;: SAVE R4
; MOV R5, -(SP) ;: SAVE R5
; MOVB $OMODE+1, R4 ;: GET THE NUMBER OF DIGITS TO TYPE
; NEG R4
; ADD #6, R4 ;: SUBTRACT IT FOR MAX. ALLOWED
; MOVB R4, $OMODE ;: SAVE IT FOR USE
; MOVB $OFILL, R4 ;: GET THE ZERO FILL SWITCH
; MOV 12 (SP), R5 ;: PICKUP THE INPUT NUMBER
; CLR R3 ;: CLEAR THE OUTPUT WORD
; ROL R5 ;: ROTATE MSB INTO "C"
; BR 3$ ;: GO DO MSB
; ROL R5 ;: FORM THIS DIGIT
; MOV R5, R3
; ROL R3 ;: GET LSB OF THIS DIGIT
```

```

2384 007636 105337 007740
2385 007642 100016
2386 007644 042703 177770
2387 007650 001002
2388 007652 005704
2389 007654 001403
2390 007656 005204
2391 007660 052703 000060
2392 007664 052703 000040
2393 007670 110337 007734
2394 007674 104401 007734
2395 007700 105337 007736
2396 007704 003347
2397 007706 002402
2398 007710 005204
2399 007712 000744
2400 007714 012605
2401 007716 012604
2402 007720 012603
2403 007722 016666 000002 000004
2404 007730 012616
2405 007732 000002
2406 007734 000
2407 007735 000
2408 007736 000
2409 007737 000
2410 007740 000000
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424 007742
2425 007742 010046
2426 007744 010146
2427 007746 010246
2428 007750 010346
2429 007752 010546
2430 007754 012746 020200
2431 007760 016605 000020
2432 007764 100004
2433 007766 005405
2434 007770 112766 000055 000001
2435 007776 005000
2436 010000 012703 010136
2437 010004 112723 000040
2438 010010 005002
2439 010012 016001 010146

```

```

DEC8 $OMODE
BPL 7$
BIC #177770,R3
BNE 4$
TST R4
BEQ 5$
INC R4
BIS #'0,R3
BIS #' ,R3
MOVB R3,8$
TYPE 8$
DEC8 $OCNT
BGT 2$
BLT 6$
INC R4
BR 2$
MOV (SP)+,R5
MOV (SP)+,R4
MOV (SP)+,R3
MOV 2(SP),4(SP)
MOV (SP)+,(SP)
RTI
8$: .BYTE 0
.BYTE 0
$OCNT: .BYTE 0
$OFILL: .BYTE 0
$OMODE: .WORD 0

```

```

:: TYPE THIS DIGIT?
:: BR IF NO
:: GET RID OF JUNK
:: TEST FOR 0
:: SUPPRESS THIS 0?
:: BR IF YES
:: DON'T SUPPRESS ANYMORE 0'S
:: MAKE THIS DIGIT ASCII
:: MAKE ASCII IF NOT ALREADY
:: SAVE FOR TYPING
:: GO TYPE THIS DIGIT
:: COUNT BY 1
:: BR IF MORE TO DO
:: BR IF DONE
:: INSURE LAST DIGIT ISN'T A BLANK
:: GO DO THE LAST DIGIT
:: RESTORE R5
:: RESTORE R4
:: RESTORE R3
:: SET THE STACK FOR RETURNING
:: RETURN
:: STORAGE FOR ASCII DIGIT
:: TERMINATOR FOR TYPE ROUTINE
:: OCTAL DIGIT COUNTER
:: ZERO FILL SWITCH
:: NUMBER OF DIGITS TO TYPE

```

.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.
MOVB #'-,1(SP) ;: MAKE THE ASCII NUMBER NEG.
1$: CLR R0 ;: ZERO THE CONSTANTS INDEX
MOV #0BLK,R3 ;: SETUP THE OUTPUT POINTER
MOVB #' ,(R3)+ ;: SET THE FIRST CHARACTER TO A BLANK
2$: CLR R2 ;: CLEAR THE BCD NUMBER
MOV $DTBL(R0),R1 ;: GET THE CONSTANT

```

```

2440 010016 160105 3$: SUB R1,R5 ;:FORM THIS BCD DIGIT
2441 010020 002402 ;:BLT 4$ ;:BR IF DONE
2442 010022 005202 ;:INC R2 ;:INCREASE THE BCD DIGIT BY 1
2443 010024 000774 ;:BR 3$
2444 010026 060105 4$: ADD R1,R5 ;:ADD BACK THE CONSTANT
2445 010030 005702 ;:TST R2 ;:CHECK IF BCD DIGIT=0
2446 010032 001002 ;:BNE 5$ ;:FALL THROUGH IF 0
2447 010034 105716 ;:TSTB (SP) ;:STILL DOING LEADING 0'S?
2448 010036 100407 ;:BMI 7$ ;:BR IF YES
2449 010040 106316 ;:ASLB (SP) ;:MSD?
2450 010042 103003 ;:BCC 6$ ;:BR IF NO
2451 010044 116663 000001 177777 ;:MOVB 1(SP),-1(R3) ;:YES--SET THE SIGN
2452 010052 052702 000060 ;:BIS #'0,R2 ;:MAKE THE BCD DIGIT ASCII
2453 010056 052702 000040 ;:BIS #' ,R2 ;:MAKE IT A SPACE IF NOT ALREADY A DIGIT
2454 010062 110223 ;:MOVB R2,(R3)+ ;:PUT THIS CHARACTER IN THE OUTPUT BUFFER
2455 010064 005720 ;:TST (R0)+ ;:JUST INCREMENTING
2456 010066 020027 000010 ;:CMP R0,#10 ;:CHECK THE TABLE INDEX
2457 010072 002746 ;:BLT 2$ ;:GO DO THE NEXT DIGIT
2458 010074 003002 ;:BGT 8$ ;:GO TO EXIT
2459 010076 010502 ;:MOV R5,R2 ;:GET THE LSD
2460 010100 000764 ;:BR 6$ ;:GO CHANGE TO ASCII
2461 010102 105726 ;:TSTB (SP)+ ;:WAS THE LSD THE FIRST NON-ZERO?
2462 010104 100003 ;:BPL 9$ ;:BR IF NO
2463 010106 116663 177777 177776 ;:MOVB -1(SP),-2(R3) ;:YES--SET THE SIGN FOR TYPING
2464 010114 105013 9$: CLRB (R3) ;:SET THE TERMINATOR
2465 010116 012605 ;:MOV (SP)+,R5 ;:POP STACK INTO R5
2466 010120 012603 ;:MOV (SP)+,R3 ;:POP STACK INTO R3
2467 010122 012602 ;:MOV (SP)+,R2 ;:POP STACK INTO R2
2468 010124 012601 ;:MOV (SP)+,R1 ;:POP STACK INTO R1
2469 010126 012600 ;:MOV (SP)+,R0 ;:POP STACK INTO R0
2470 010130 104401 010156 000004 ;:TYPE $DBLK ;:NOW TYPE THE NUMBER
2471 010134 016666 000002 ;:MOV 2(SP),4(SP) ;:ADJUST THE STACK
2472 010142 012616 ;:MOV (SP)+,(SP)
2473 010144 000002 ;:RTI ;:RETURN TO USER
2474 010146 023420 ;:SOTBL: 10000.
2475 010150 001750 ;: 1000.
2476 010152 000144 ;: 100.
2477 010154 000012 ;: 10.
2478 010156 000004 ;:SDBLK: .BLKW 4
2479 ;:
2480 ;:SBTTL TTY INPUT ROUTINE
2481 ;:*****
2482 ;:ENABL LSB
2483 $TKCNT: .WORD 0 ;:NUMBER OF ITEMS IN QUEUE
2484 $TKQIN: .WORD 0 ;:INPUT POINTER
2485 $TKQOUT: .WORD 0 ;:OUTPUT POINTER
2486 $TKQSRT: .BLKB 7 ;:TTY KEYBOARD QUEUE
2487 $TKQEND=.
2488 .EVEN
2489 ;:
2490 ;:*TK INITIALIZE ROUTINE
2491 ;:*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
2492 ;:*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
2493 ;:
2494 ;:*CALL:

```

```

2496          ;*      JSR      PC,$TKINT
2497          ;*      RETURN
2498
2499 010204 005037 010166          $TKINT: CLR      $TKCNT      ;; CLEAR COUNT OF ITEMS IN QUEUE
2500 010210 012737 010174 010170  MOV     #$TKQSR $TKQIN    ;; MOVE THE STARTING ADDRESS OF THE
2501 010216 013737 010170 010172  MOV     $TKQIN,$TKQOUT ;; QUEUE INTO THE INPUT & OUTPUT POINTERS.
2502 010224 012737 010254 000060  MOV     #$TKSRV,$TKVEC ;; INITIALIZE THE KEYBOARD VECTOR
2503 010232 012737 000200 000062  MOV     #200,$TKVEC+2 ;; "BR" LEVEL 4
2504 010240 005777 170702          TST     $TKB        ;; CLEAR DONE FLAG
2505 010244 012777 000100 170672  MOV     #100,$TKS    ;; ENABLE TTY KEYBOARD INTERRUPT
2506 010252 000207          RTS      PC        ;; RETURN TO CALLER
2507
2508          ;*TK SERVICE ROUTINE
2509          ;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
2510          ;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
2511          ;*IT IN THE QUEUE.
2512          ;*IF THE CHARACTER IS A "CONTROL-C" (↑C) $TKINT IS CALLED AND
2513          ;*UPON RETURN EXIT IS MADE TO THE "CONTROL-C" RESTART ADDRESS (↑CNTLC)
2514
2515 010254 117746 170666          $TKSRV: MOVB   $TKB,-(SP) ;; PICK UP THE CHARACTER
2516 010260 042716 177600          BIC     #↑C177,(SP) ;; STRIP THE JUNK
2517 010264 021627 000003          CMP     (SP),#3      ;; IS IT A CONTROL C?
2518 010270 001007          BNE     1$          ;; BRANCH IF NO
2519 010272 104401 011367          TYPE   $CNTLC      ;; TYPE A CONTROL-C (↑C)
2520 010276 004737 010204          JSR     PC,$TKINT  ;; INIT THE KEYBOARD
2521 010302 005726          *ST     (SP)+      ;; CLEAN UP STACK
2522 010304 000177 170770          JMP     ↑CNTLC     ;; CONTROL C RESTART
2523 010310 021627 000007          1$:  CMP     (SP),#7   ;; IS IT A CONTROL G?
2524 010314 001004          BNE     2$          ;; BRANCH IF NO
2525 010316 022737 000176 001140  CMF     $SWREG,$SWR ;; IS SOFT-SWR SELECTED?
2526 010324 001500          BFG     5$          ;; GO TO SWR CHANGE
2527
2528          2$:
2529 010326 022737 000007 010166          CMP     #7,$TKCNT  ;; IS THE QUEUE FULL?
2530 010334 001004          BNE     3$          ;; BRANCH IF NO
2531 010336 104401 001162          TYPE   $BELL      ;; RING THE TTY BELL
2532 010342 005726          TST     (SP)+      ;; CLEAN CHARACTER OFF OF STACK
2533 010344 000451          BR      5$          ;; EXIT
2534 010346 021627 000023          3$:  CMP     (SP),#23   ;; IS IT A CONTROL-S?
2535 010352 001021          BNE     32$         ;; BRANCH IF NO
2536 010354 005077 170564          CLR     $TKS      ;; DISABLE TTY KEYBOARD INTERRUPTS
2537 010360 005726          TST     (SP)+      ;; CLEAN CHAR OFF STACK
2538 010362 105777 170556          31$: TSTB   $TKS      ;; WAIT FOR A CHAR
2539 010366 100375          BPL     31$        ;; LOOP UNTIL ITS THERE
2540 010370 117746 170552          MOVB   $TKB,-(SP) ;; GET THE CHARACTER
2541 010374 042716 177600          BIC     #↑C177,(SP) ;; MAKE IT 7-BIT ASCII
2542 010400 022627 000021          CMP     (SP)+,#21  ;; IS IT A CONTROL-Q?
2543 010404 001366          BNE     31$        ;; BRANCH IF NO
2544 010406 012777 000100 170530  MOV     #100,$TKS  ;; REENABLE TTY KEYBOARD INTERRUPTS
2545 010414 000002          RTI                    ;; RETURN
2546 010416 005237 010166          32$: INC     $TKCNT     ;; COUNT THIS CHARACTER
2547 010422 021627 000140          CMP     (SP),#140 ;; IS IT UPPER CASE?
2548 010426 002405          BLT     4$          ;; BRANCH IF YES
2549 010430 021627 000175          CMP     (SP),#175 ;; IS IT A SPECIAL CHAR?
2550 010434 003002          BGT     4$          ;; BRANCH IF YES
2551 010436 042716 000040          BIC     #40,(SP)   ;; MAKE IT UPPER CASE

```

```

2552 010442 112677 177522 4$:   MOVB   (SP)+, @STKQIN   ;; AND PUT IT IN QUEUE
2553 010446 005237 010170       INC   $TKQIN             ;; UPDATE THE POINTER
2554 010452 023727 010170 010203   CMP   $TKQIN, @STKQEND   ;; GO OFF THE END?
2555 010460 001003       BNE   5$               ;; BRANCH IF NO
2556 010462 012737 010174 010170   MOV   @STKQSR, $TKQIN    ;; RESET THE POINTER
2557 010470 000002 5$:   RTI                    ;; RETURN
2558
2559 *****
2560 *SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
2561 *ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
2562 *SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
2563 *CALL WHEN OPERATING IN TTY INTERRUPT MODE.
2564 010472 022737 000176 001140 $CKSWR: CMP   @SWREG, SWR      ;; IS THE SOFT-SWR SELECTED
2565 010500 001124       BNE   15$              ;; EXIT IF NOT
2566 010502 105777 170436   TSTB  @STKS           ;; IS A CHAR WAITING?
2567 010506 100121       BPL   15$              ;; IF NOT, EXIT
2568 010510 117746 170432   MOVB  @STKB, -(SP)      ;; YES
2569 010514 042716 177600   BIC   #1C177, (SP)     ;; MAKE IT 7-BIT ASCII
2570 010520 021627 000007   CMP   (SP), #7         ;; IS IT A CONTROL-G?
2571 010524 001300       BNE   2$               ;; IF NOT, PUT IT IN THE TTY QUEUE
2572                                     ;; AND EXIT
2573
2574 *****
2575 *CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
2576 *ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
2577 *CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
2578 010526 123727 001134 000001 6$:   CMPB  $AUTOB, #1       ;; ARE WE RUNNING IN AUTO-MODE?
2579 010534 001674       BEQ   2$               ;; BRANCH IF YES
2580 010536 005726       TST  (SP)+            ;; CLEAR CONTROL-G OFF STACK
2581 010540 004737 010204   JSR   PC, $TKINT       ;; FLUSH THE TTY INPUT QUEUE
2582 010544 005077 170374   CLR   @STKS           ;; DISABLE TTY KEYBOARD INTERRUPTS
2583 010550 112737 000001 001135   MOVB  #1, $INTAG      ;; SET INTERRUPT MODE INDICATOR
2584
2585 010556 104401 011401   $GTSWR: TYPE   , $CNTLG   ;; ECHO THE CONTROL-G (1G)
2586 010562 104401 011406   TYPE   $MSWR          ;; TYPE CURRENT CONTENTS
2587 010566 013746 000176   MOV   $SWREG, -(SP)   ;; SAVE SWREG FOR TYPEOUT
2588 010572 104402       TYPOC                ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
2589 010574 104401 011417   TYPE   , $MNEW        ;; PROMPT FOR NEW SWR
2590 010600 005046 19$:   CLR   -(SP)           ;; CLEAR COUNTER
2591 010602 005046       CLR   -(SP)           ;; THE NEW SWR
2592 010604 105777 170334 7$:   TSTB  @STKS           ;; CHAR THERE?
2593 010610 100375       BPL   7$              ;; IF NOT TRY AGAIN
2594
2595 010612 117746 170330   MOVB  @STKB, -(SP)    ;; PICK UP CHAR
2596 010616 042716 177600   BIC   #1C177, (SP)   ;; MAKE IT 7-BIT ASCII
2597
2598 010622 021627 000003   CMP   (SP), #3        ;; IS IT A CONTROL-C?
2599 010626 001015       BNE   9$              ;; BRANCH IF NOT
2600 010630 104401 011367   TYPE   , $CNTLC       ;; YES, ECHO CONTROL-C (1C)
2601 010634 062706 000006   ADD   #6, SP          ;; CLEAN UP STACK
2602 010640 123727 001135 000001 8$:   CMPB  $INTAG, #1     ;; REENABLE TTY KEYBOARD INTERRUPTS?
2603 010646 001003       BNE   8$              ;; BRANCH IF NO
2604 010650 012777 000100 170266   MOV   #100, @STKS     ;; ALLOW TTY KEYBOARD INTERRUPTS
2605 010656 000177 170416 8$:   JMP   @CNTLC         ;; CONTROL-C RESTART
2606
2607

```

```

2608 010662 021627 000025 95: CMP (SP),#25 ;: IS IT A CONTROL-U?
2609 010666 001005 BNE 10$ ;: BRANCH IF NOT
2610 010670 104401 011374 TYPE $CNTLU ;: YES, ECHO CONTROL-U (↑U)
2611 010674 062706 000006 20$: ADD #6,SP ;: IGNORE PREVIOUS INPUT
2612 010700 000737 BR 19$ ;: LET'S TRY IT AGAIN
2613
2614
2615 010702 021627 000015 10$: CMP (SP),#15 ;: IS IT A <CR>?
2616 010706 001022 BNE 16$ ;: BRANCH IF NO
2617 010710 005766 000004 TST 4(SP) ;: YES, IS IT THE FIRST CHAR?
2618 010714 001403 BEQ 11$ ;: BRANCH IF YES
2619 010716 016677 000002 170214 MOV 2(SP),@SWR ;: SAVE NEW SWR
2620 010724 062706 000006 11$: ADD #6,SP ;: CLEAR UP STACK
2621 010730 104401 001167 14$: TYPE $CRLF ;: ECHO <CR> AND <LF>
2622 010734 123727 001135 000001 CMPB $INTAG,#1 ;: RE-ENABLE TTY KBD INTERRUPTS?
2623 010742 001003 BNE 15$ ;: BRANCH IF NOT
2624 010744 012777 000100 170172 MOV #100,@STKS ;: RE-ENABLE TTY KBD INTERRUPTS
2625 010752 000002 15$: RTI ;: RETURN
2626 010754 004737 007444 16$: JSR PC,$TYPEC ;: ECHO CHAR
2627 010760 021627 000060 CMP (SP),#60 ;: CHAR < 0?
2628 010764 002420 BLT 18$ ;: BRANCH IF YES
2629 010766 021627 000067 CMP (SP),#67 ;: CHAR > ?
2630 010772 003015 BGT 18$ ;: BRANCH IF YES
2631 010774 042726 000060 BIC #60,(SP)+ ;: STRIP-OFF ASCII
2632 011000 005766 000002 TST 2(SP) ;: IS THIS THE FIRST CHAR
2633 011004 001403 BEQ 17$ ;: BRANCH IF YES
2634 011006 006316 ASL (SP) ;: NO, SHIFT PRESENT
2635 011010 006316 ASL (SP) ;: CHAR OVER TO MAKE
2636 011012 006316 ASL (SP) ;: ROOM FOR NEW ONE.
2637 011014 005266 000002 17$: INC 2(SP) ;: KEEP COUNT OF CHAR
2638 011020 056616 177776 BIS -2(SP),(SP) ;: SET IN NEW CHAR
2639 011024 000667 BR 7$ ;: GET THE NEXT ONE
2640 011026 104401 001166 18$: TYPE $QUES ;: TYPE ?<CR><LF>
2641 011032 000720 BR 20$ ;: SIMULATE CONTROL-U
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653 011034 011646 $RDCHR: MOV (SP),-(SP) ;: PUSH DOWN THE PC AND
2654 011036 016666 000004 000C02 MOV 4(SP),2(SP) ;: THE PS
2655 011044 005066 000004 CLR 4(SP) ;: GET READY FOR A CHARACTER
2656 011050 005046 CLR -(SP) ;: PUT NEW PS ON STACK
2657 011052 012746 011060 MOV #64$,-(SP) ;: PUT NEW PC ON STACK
2658 011056 000002 RTI ;: POP NEW PC AND PS
2659 011060 64$:
2660 011060 005737 010166 1$: TST $TKCNT ;: WAIT ON A CHARACTER
2661 011064 001775 BEQ 1$
2662 011066 005337 010166 DEC $TKCNT ;: DECREMENT THE COUNTER
2663 011072 117766 177074 000004 MOVB @STKQOUT,4(SP) ;: GET ONE CHARACTER

```

```

;: *****
;: 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
;:

```

```

2664 011100 005237 010172          INC      $TKQOUT          ;; UPDATE THE POINTER
2665 011104 023727 010172 010203    CMP      $TKQOUT, $TKQEND ;; DID IT GO OFF OF THE END?
2666 011112 001003          BNE      2$              ;; BRANCH IF NO
2667 011114 012737 010174 010172    MOV      $STKQRT, $TKQOUT ;; RESET THE POINTER
2668 011122 000037          RTI                      ;; RETURN
2669                                     ;; *****
2670                                     ;; THIS ROUTINE WILL INPUT A STRING FROM THE TTY
2671                                     ;; CALL:
2672                                     ;; *
2673                                     ;; *   RDLIN          ;; INPUT A STRING FROM THE TTY
2674                                     ;; *   RETURN HERE  ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
2675                                     ;; *               ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
2676 011124 010346          $RDLIN: MOV      R3, -(SP)      ;; SAVE R3
2677 011126 005046          CLR      -(SP)          ;; CLEAR THE RUBOUT KEY
2678 011130 012703 011360          1$:   MOV      $STTYIN, R3 ;; GET ADDRESS
2679 011134 022703 011367          2$:   CMP      $STTYIN+7, R3 ;; BUFFER FULL?
2680 011140 101456          BLOS     4$              ;; BR IF YES
2681 011142 104410          RDCHR   (SP)+, (R3)     ;; GO READ ONE CHARACTER FROM THE TTY
2682 011144 112613          MOVVB   (SP)+, (R3)     ;; GET CHARACTER
2683 011146 122713 000177          10$:  CMPB     #177, (R3)     ;; IS IT A RUBOUT
2684 011152 001022          BNE      5$              ;; BR IF NO
2685 011154 005716          TST     (SP)            ;; IS THIS THE FIRST RUBOUT?
2686 011156 001007          BNE      6$              ;; BR IF NO
2687 011160 112737 000134 011356    MOVVB   #' \, 9$        ;; TYPE A BACK SLASH
2688 011166 104401 011356          TYPE    9$              ;;
2689 011172 012716 177777          MOV     #-1, (SP)      ;; SET THE RUBOUT KEY
2690 011176 005303          6$:   DEC     R3            ;; BACKUP BY ONE
2691 011200 020327 011360          CMP     R3, $STTYIN    ;; STACK EMPTY?
2692 011204 103434          BLOS     4$              ;; BR IF YES
2693 011206 111337 011356          MOVVB   (R3), 9$        ;; SETUP TO TYPEOUT THE DELETED CHAR.
2694 011212 104401 011356          TYPE    9$              ;; GO TYPE
2695 011216 000746          BR      2$              ;; GO READ ANOTHER CHAR.
2696 011220 005716          5$:   TST     (SP)            ;; RUBOUT KEY SET?
2697 011222 001406          BEQ     7$              ;; BR IF NO
2698 011224 112737 000134 011356    MOVVB   #' \, 9$        ;; TYPE A BACK SLASH
2699 011232 104401 011356          TYPE    9$              ;;
2700 011236 005016          CLR     (SP)            ;; CLEAR THE RUBOUT KEY
2701 011240 122713 000025          7$:   CMPB     #25, (R3)    ;; IS CHARACTER A CTRL U?
2702 011244 001003          BNE     8$              ;; BR IF NO
2703 011246 104401 011374          TYPE    $CNTLU          ;; TYPE A CONTROL "U"
2704 011252 000726          BR      1$              ;; GO START OVER
2705 011254 122713 000022          8$:   CMPB     #22, (R3)    ;; IS CHARACTER A "+R"?
2706 011260 001011          BNE     3$              ;; BRANCH IF NO
2707 011262 105013          CLRB   (R3)            ;; CLEAR THE CHARACTER
2708 011264 104401 001167          TYPE    $SCRLF          ;; TYPE A "CR" & "LF"
2709 011270 104401 011360          TYPE    $STTYIN         ;; TYPE THE INPUT STRING
2710 011274 000717          BR      2$              ;; GO PICKUP ANOTHER CHARACTER
2711 011276 104401 001166          4$:   TYPE    $QUES        ;; TYPE P '?'
2712 011302 000712          BR      1$              ;; CLEAR THE BUFFER AND LOOP
2713 011304 111337 011356          3$:   MOVVB   (R3), 9$        ;; ECHO THE CHARACTER
2714 011310 104401 011356          TYPE    9$              ;;
2715 011314 122723 000015          CMPB     #15, (R3)+     ;; CHECK FOR RETURN
2716 011320 001305          BNE     2$              ;; LOOP IF NOT RETURN
2717 011322 105063 177777          CLRB   -1(R3)          ;; CLEAR RETURN (THE 15)
2718 011326 104401 001170          TYPE    $LF            ;; TYPE A LINE FEED
2719 011332 005726          TST     (SP)+          ;; CLEAN RUBOUT KEY FROM THE STACK

```

```

2720 011334 012603          MOV      (SP)+,R3          ;;RESTORE R3
2721 011336 011646          MOV      (SP)-(SP)        ;;ADJUST THE STACK AND PUT ADDRESS OF THE
2722 011340 016666 000004 000002  MOV      4(SP),2(SP)      ;;FIRST ASCII CHARACTER ON IT
2723 011346 012766 011360 000004  MOV      #STTYIN,4(SP)
2724 011354 000002          RTI                      ;;RETURN
2725 011356 000          9$: .BYTE 0                ;;STORAGE FOR ASCII CHAR. TO TYPE
2726 011357 000          .BYTE 0                ;;TERMINATOR
2727 011360 000007  STTYIN: .BLKB 7          ;;RESERVE 7 BYTES FOR TTY INPUT
2728 011367 136 006503 000012  SCNTLC: .ASCIZ /↑C/⟨15⟩⟨12⟩ ;;CONTROL "C"
2729 011374 052536 005015 000          SCNTLU: .ASCIZ /↑U/⟨15⟩⟨12⟩ ;;CONTROL "U"
2730 011401 136 006507 000012  SCNTLG: .ASCIZ /↑G/⟨15⟩⟨12⟩ ;;CONTROL "G"
2731 011406 005015 053523 020122  SMSWR:  .ASCIZ  ⟨15⟩⟨12⟩/SWR = /
2732 011414 020075 000          SMNEW:  .ASCIZ  / NEW = /
2733 011417 040 047040 053505
2734 011424 036440 000040

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

.SBTTL 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, @#SSB2D   ;;CALL
*   RETURN   ;;ADDRESS OF THE 1ST ASCIZ CHAR. IS ON THE STACK
*
2771 011470 016637 000002 011520  SSB2D: MOV      2(SP), 1$      ;;SAVE BINARY NUMBER
2772 011476 012746 011520  MOV      #1$, -(SP)      ;;SET POINTER
2773 011502 004737 011524  JSR      PC, @#SSB2D     ;;CALL DOUBLE LENGTH CONVERT
2774 011506 062716 000005  ADD     #5, (SP)         ;;ONLY ALLOW FIVE CHARACTERS
2775 011512 012666 000002  MOV     (SP)+, 2(SP)     ;;PICKUP POINTER

```

```

2776 011516 000207          ; RTS          PC          ;; RETURN
2777 011520 000000 000000 1$: .WORD C,0
2778
2779 .SBTTL DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
2780
2781 ;*****
2782 ;*THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
2783 ;*DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
2784 ;*POSITIVE.
2785 ;*CALL
2786 ;*      MOV      #PNTR, -(SP)      ;; POINTER TO LOW WORD OF BINARY NUMBER
2787 ;*      JSR      PC, @#$DB2D      ;; THE FIRST ADDRESS OF ASCIZ
2788 ;*      RETURN                      ;; IS ON THE STACK
2789
2790
2791
2792 011524 104412          $DB2D: SAVREG      ;; SAVE REGISTERS
2793 011526 016602 000002      MOV      2(SP), R2      ;; PICKUP THE DATA POINTER
2794 011532 012700 011704      MOV      #$DECVL, R0    ;; GET ADDRESS OF "$DECVL" STRING
2795 011536 010066 000002      MOV      R0, 2(SP)     ;; PUT ADDRESS OF ASCIZ STRING ON STACK
2796 011542 012201          MOV      (R2)+, R1      ;; PICKUP THE BINARY NUMBER
2797 011544 012202          MOV      (R2)+, R2
2798 011546 012737 000012 011622 MOV      #10, R4        ;; SET UP TO DO 10 CONVERSIONS
2799 011554 012704 011634      MOV      $STNPWR, R4    ;; ADDRESS OF TEN POWER
2800 011560 012705 011636      MOV      $STNPWR+2, R5
2801 011564 005003          1$: CLR      R3          ;; CLEAR PARTIAL
2802 011566 161401          2$: SUB      (R4), R1      ;; SUBTRACT TEN POWER
2803 011570 005602          SBC      R2
2804 011572 161502          SUB      (R5), R2
2805 011574 002402          BLT      3$           ;; BR IF TEN POWER TOO LARGE
2806 011576 005203          INC      R3          ;; ADD 1 TO PARTIAL
2807 011600 000772          BR      2$           ;; LOOP
2808 011602 062401          3$: ADD      (R4)+, R1      ;; RESTORE SUBTRACTED VALUE
2809 011604 005502          ADC      R2
2810 011606 062402          ADD      (R4)+, R2
2811 011610 022525          CMP      (R5)+, (R5)+  ;; MOVE TO NEXT TEN POWER
2812 011612 052703 000060      B'S      #0, R3        ;; CHANGE PARTIAL TO ASCII
2813 011616 110320          MOV      R3, (R0)+     ;; SAVE IT
2814 011620 005327          DEC      (PC)+        ;; DONE?
2815 011622 000000          4$: .WORD      0
2816 011624 001357          BNE      1$           ;; BR IF NO
2817 011626 105020          CLRB    (R0)+        ;; TERMINATOR
2818 011630 104413          RESREG  ;; RESTORE REGISTERS
2819 011632 000207          RTS      PC          ;; RETURN
2820 011634 145000          $STNPWR: 145000      ;; 1.0E09
2821 011636 035632          35632
2822 011640 160400          160400      ;; 1.0E08
2823 011642 002765          2765
2824 011644 113200          113200      ;; 1.0E07
2825 011646 000230          230
2826 011650 041100          041100      ;; 1.0E06
2827 011652 000017          17
2828 011654 103240          103240      ;; 1.0E05
2829 011656 000001          1
2830 011660 023420          23420      ;; 1.0E04
2831 011662 000000          0

```

```

2832 011664 001750
2833 011666 000000
2834 011670 000144
2835 011672 000000
2836 011674 000012
2837 011676 000000
2838 011700 000001
2839 011702 000000
2840 011704 000014
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859 011720
2860 011720 010046
2861 011722 010146
2862 011724 010246
2863 011726 010346
2864 011730 010446
2865 011732 010546
2866 011734 016646 000022
2867 011740 01E646 000022
2868 011744 016646 000022
2869 011750 016646 000022
2870 011754 000002
2871
2872
2873
2874
2875 011756
2876 011756 012666 000022
2877 011762 012666 000022
2878 011766 012666 000022
2879 011772 012666 000022
2880 011776 012605
2881 012000 012604
2882 012002 012603
2883 012004 012602
2884 012006 012601
2885 012010 012600
2886 012012 000002
2887

```

```

1750 ;:1.0E03
0
144 ;:1.0E02
0
12 ;:1.0E01
0
1 ;:1.0E00
0
$DECVL: .BLKB 12. ;:RESERVE STORAGE FOR ASCIZ 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
$SAVREG:
MOV RO, -(SP) ;:PUSH RO ON STACK
MOV R1, -(SP) ;:PUSH R1 ON STACK
MOV R2, -(SP) ;:PUSH R2 ON STACK
MOV R3, -(SP) ;:PUSH R3 ON STACK
MOV R4, -(SP) ;:PUSH R4 ON STACK
MOV R5, -(SP) ;:PUSH R5 ON STACK
MOV 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
MOV (SP)+, R5 ;:POP STACK INTO R5
MOV (SP)+, R4 ;:POP STACK INTO R4
MOV (SP)+, R3 ;:POP STACK INTO R3
MOV (SP)+, R2 ;:POP STACK INTO R2
MOV (SP)+, R1 ;:POP STACK INTO R1
MOV (SP)+, R0 ;:POP STACK INTO R0
RTI

```

2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943

.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.

012014 010046
012016 016600 000002
012022 005740
012024 111000
012026 006300
012030 01600C 012050
012034 000200

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

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

012036 011646
012040 016666 000004 000002
012046 000002

\$TRAP2: MOV (SP), -(SP) ;; MOVE THE PC DOWN
MOV 4(SP), 2(SP) ;; MOVE THE PSW DOWN
RTI ;; RESTORE THE PSW

.SBTTL TRAP TABLE

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

	ROUTINE		

\$TRPAD:	.WORD \$TRAP2		
	\$TYPE ;; CALL=TYPE	TRAP+1(104401)	TTY TIMEOUT ROUTINE
	\$TYPOC ;; CALL=TYPOC	TRAP+2(104402)	TYPE OCTAL NUMBER (WITH LEADING ZEROS)
	\$TYPOS ;; CALL=TYPOS	TRAP+3(104403)	TYPE OCTAL NUMBER (NO LEADING ZEROS)
	\$TYPON ;; CALL=TYPON	TRAP+4(104404)	TYPE OCTAL NUMBER (AS PER LAST CALL)
	\$TYPDS ;; CALL=TYPDS	TRAP+5(104405)	TYPE DECIMAL NUMBER (WITH SIGN)
012050 012036	\$GTSWR ;; CALL=GTSWR	TRAP+6(104406)	GET SOFT-SWR SETTING
012052 007274	\$CKSWR ;; CALL=CKSWR	TRAP+7(104407)	TEST FOR CHANGE IN SOFT-SWR
012054 007540	\$RDCHR ;; CALL=RDCHR	TRAP+10(104410)	TTY TYPEIN CHARACTER ROUTINE
012056 007514	\$RDLIN ;; CALL=RDLIN	TRAP+11(104411)	TTY TYPEIN STRING ROUTINE
012060 007554	\$SAVREG ;; CALL=SAVREG	TRAP+12(104412)	SAVE RO-R5 ROUTINE
012062 007742	\$RESREG ;; CALL=RESREG	TRAP+13(104413)	RESTORE RO-R5 ROUTINE
012064 010562			
012066 010472			
012070 011034			
012072 011124			
012074 011720			
012076 011756			

.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

```

2944 ;STORAGE FOR RPDS1, RPER1, RPER2, AND RPER3 ON AN ERROR "2"
2945 ;RPERRS = RPDS1
2946 ;RPERRS+2 = RPER1
2947 ;RPERRS+4 = RPER2
2948 ;RPERRS+6 = RPER3

```

```

2949
2950 012100 000000 000000 000000 RPERRS: .WORD 0,0,0,0
2951 012106 000000

```

```

2952 ;TABLE OF DRIVE ACTIVE INDICATORS (DRVACT=8 BYTES)
2953 ;DRVACT=0 IF DRIVE IS IDLE
2954 ;DRVACT>0 IF DRIVE IS ACTIVE WITH A COMMAND
2955 ;DRVACT<0 IF DRIVE IS ACTIVE WITH AN ERROR RECOVERY OPERATION

```

```

2956
2957
2958 012110 000 DRVACT: .BYTE 0 ;DRIVE 0
2959 012111 000 .BYTE 00 ;DRIVE 1
2960 012112 000 .BYTE 00 ;DRIVE 2
2961 012113 000 .BYTE 00 ;DRIVE 3
2962 012114 000 .BYTE 00 ;DRIVE 4
2963 012115 000 .BYTE 00 ;DRIVE 5
2964 012116 000 .BYTE 00 ;DRIVE 6
2965 012117 000 .BYTE 0 ;DRIVE 7

```

```

2966 ;TABLE OF DRIVE STATUS INDICATORS (DRVSTA=8 BYTES)
2967 ;DRVSTA=0 IF DRIVE IS OFFLINE OR NONEXISTENT
2968 ;DRVSTA>0 IF DRIVE IS ONLINE
2969 ;DRVSTA<0 IF DRIVE IS UNSAFE

```

```

2970
2971
2972 012120 000 DRVSTA: .BYTE 0 ;DRIVE 0
2973 012121 000 .BYTE 00 ;DRIVE 1
2974 012122 000 .BYTE 00 ;DRIVE 2
2975 012123 000 .BYTE 00 ;DRIVE 3
2976 012124 000 .BYTE 00 ;DRIVE 4
2977 012125 000 .BYTE 00 ;DRIVE 5
2978 012126 000 .BYTE 00 ;DRIVE 6
2979 012127 000 .BYTE 0 ;DRIVE 7

```

```

2980 ;TABLE OF DRIVE TYPES (DRV TYP=8 BYTES)
2981 ;DRV TYP=0 IF DRIVE IS NONEXISTENT (DRVSTA=0, ALSO)
2982 ;DRV TYP=1 IF DRIVE IS RPO4
2983 ;DRV TYP=2 IF DRIVE IS RPO5
2984 ;DRV TYP=4 IF DRIVE IS RPO6
2985 ;DRV TYP=-1 IF NOT RPO4/5/6

```

```

2986
2987
2988 012130 000 DRV TYP: .BYTE 0 ;DRIVE 0
2989 012131 000 .BYTE 00 ;DRIVE 1
2990 012132 000 .BYTE 00 ;DRIVE 2
2991 012133 000 .BYTE 00 ;DRIVE 3
2992 012134 000 .BYTE 00 ;DRIVE 4
2993 012135 000 .BYTE 00 ;DRIVE 5
2994 012136 000 .BYTE 00 ;DRIVE 6
2995 012137 000 .BYTE 0 ;DRIVE 7

```

```

2996 ;TABLE OF DUAL PORT INITIALIZATION INDICATORS
2997 ;DPINT=0 IF INITIALIZATION IS NOT ACTIVE ON THE DRIVE
2998 ;DPINT<0 IF INITIALIZATION IS IN PROGRESS
2999

```

F05

CZRJB80 RPO4.5/6 FMTR MACY11 30(1046)
 CZRJB8.P11 04-NOV-77 12:54

07-NOV-77 10:07 PAGE 58
 SINGLE/DUAL PORT RH11/RPO4/5/6 DRIVER (REV 1.0)

SEQ 0057

3000					
3001	012140	000	DPINT:	.BYTE 0	;DRIVE 0
3002	012141	000		.BYTE 0	;DRIVE 1
3003	012142	000		.BYTE 0	;DRIVE 2
3004	012143	000		.BYTE 0	;DRIVE 3
3005	012144	000		.BYTE 0	;DRIVE 4
3006	012145	000		.BYTE 0	;DRIVE 5
3007	012146	000		.BYTE 0	;DRIVE 6
3008	012147	000		.BYTE 0	;DRIVE 7
3009					
3010			;TABLE OF PENDING DUAL PORT REQUESTS		
3011			;DPRQS=0 IF THAT A DUAL PORT REQUEST IS NOT PENDING FOR THAT DRIVE		
3012			;DPRQS<0 IF THAT A DUAL PORT REQUEST IS PENDING FOR THAT DRIVE		
3013					
3014	012150	000	DPRQS:	.BYTE 0	;DRIVE 0
3015	012151	000		.BYTE 0	;DRIVE 1
3016	012152	000		.BYTE 0	;DRIVE 2
3017	012153	000		.BYTE 0	;DRIVE 3
3018	012154	000		.BYTE 0	;DRIVE 4
3019	012155	000		.BYTE 0	;DRIVE 5
3020	012156	000		.BYTE 0	;DRIVE 6
3021	012157	000		.BYTE 0	;DRIVE 7
3022					
3023			;TRANSFER WAIT FLAG (TRNSWT=1 WORD)		
3024			;THIS IS A ONE WORD QUEUE. IT WILL CONTAIN THE ADDRESS OF		
3025			;"DPB" OF THE I/O OPERATION.		
3026					
3027	012160	000000	TRNSWT:	.WORD 0	
3028					
3029			;SEARCH WAIT KEYS (SRCHWT=1 WORD)		
3030			;THIS IS A ONE WORD QUEUE THAT WILL CONTAIN A KEY FOR EACH OF		
3031			;THE DRIVES THAT ARE PERFORMING A SEARCH COMMAND FOR THE I/O		
3032			;REQUEST THAT IS AT THE TOP OF THEIR REQUEST QUEUE.		
3033			;EACH DRIVE IS ASSIGNED ONE BIT, STARTING AT BIT00 FOR DRIVE 0.		
3034					
3035	012162	000000	SRCHWT:	.WORD 0	
3036					
3037			;RPO4/5/6 DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)		
3038			;ACTDRV=0 IF DRIVER IS INACTIVE		
3039			;ACTDRV>0 IF DRIVER IS ACTIVE		
3040					
3041	012164	000	ACTDRV:	.BYTE 0	
3042					
3043			;SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)		
3044			;ACTSTR=0 IF SOFTWARE TIMER ROUTINE IS INACTIVE		
3045			;ACTSTR>0 IF SOFTWARE TIMER ROUTINE IS ACTIVE		
3046					
3047	012165	000	ACTSTR:	.BYTE 0	
3048					
3049			;UNLOAD FLAG (ULDFLG=8 BYTES)		
3050			;ULDFLG=0 IF NO UNLOAD COMMAND		
3051			;ULDFLG>0 IF UNLOAD COMMAND IN PROGRESS		
3052			;ULDFLG<0 IF UNLOAD COMMAND IN WAIT QUEUE		
3053					
3054	012166	000	ULDFLG:	.BYTE 0	;DRIVE 0
3055	012167	000		.BYTE 0	;DRIVE 1

```

3056 012170 000 .BYTE 0 ;DRIVE 2
3057 012171 000 .BYTE 0 ;DRIVE 3
3058 012172 000 .BYTE 0 ;DRIVE 4
3059 012173 000 .BYTE 0 ;DRIVE 5
3060 012174 000 .BYTE 0 ;DRIVE 6
3061 012175 000 .BYTE 0 ;DRIVE 7
3062
3063 ;LOOK AHEAD COUNT (LACNT=8 BYTES)
3064 ;LACNT WILL INDICATE THE NUMBER OF LOOK AHEADS PERFORMED
3065
3066 012176 000 LACNT: .BYTE 0 ;DRIVE 0
3067 012177 000 .BYTE 0 ;DRIVE 1
3068 012200 000 .BYTE 0 ;DRIVE 2
3069 012201 000 .BYTE 0 ;DRIVE 3
3070 012202 000 .BYTE 0 ;DRIVE 4
3071 012203 000 .BYTE 0 ;DRIVE 5
3072 012204 000 .BYTE 0 ;DRIVE 6
3073 012205 000 .BYTE 0 ;DRIVE 7
3074
3075 ;SAVE REGISTERS FLAG (SAVEFG =1 WORD)
3076 ;SAVEFG <0 IF SAVE THE RH11/RPO4/5/6 REGISTERS WHEN THE
3077 ;OPERATION IS COMPLETED AS PER (DPB+14).
3078 ;SAVEFG=0 IF SAVE THE RH11/RPO4/5/6 REGISTERS, AS PER
3079 ;(DPB+14), AFTER AN ERROR.
3080
3081 012206 000000 SAVEFG: .WORD 0
3082
3083 ;SEEK FLAG (SEEKFG=1 WORD)
3084 ;SEEKFG=0 IF WHEN THE DISK ADDRESS ISN'T IN THE WINDOW
3085 ;FOR A DATA TRANSFER START A SEARCH COMMAND
3086 ;SEEKFG<0 IF DATA TRANSFER WILL DO IMPLIED SEEKS,
3087 ;DISREGARD THE WINDOW
3088
3089 012210 000000 SEEKFG: .WORD 0
3090
3091 ;TIMEOUT TABLE (TIMER=8 WORDS)
3092 ;THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION
3093
3094 012212 177777 TIMER: .WORD -1 ;DRIVE 0
3095 012214 177777 .WORD -1 ;DRIVE 1
3096 012216 177777 .WORD -1 ;DRIVE 2
3097 012220 177777 .WORD -1 ;DRIVE 3
3098 012222 177777 .WORD -1 ;DRIVE 4
3099 012224 177777 .WORD -1 ;DRIVE 5
3100 012226 177777 .WORD -1 ;DRIVE 6
3101 012230 177777 .WORD -1 ;DRIVE 7
3102
3103 ;DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)
3104 ;DTUW<0 IF NO DATA TRANSFER UNDERWAY
3105 ;DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE N
3106
3107 012232 177777 DTUW: .WORD -1
3108
3109 ;ATTENTION BITS TABLE (ATABIT=8 BYTES)
3110 ;THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES
3111 ;ATTENTION BIT

```

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

ATABIT: .BYTE 1 :DRIVE 0
 .BYTE 2 :DRIVE 1
 .BYTE 4 :DRIVE 2
 .BYTE 10 :DRIVE 3
 .BYTE 20 :DRIVE 4
 .BYTE 40 :DRIVE 5
 .BYTE 100 :DRIVE 6
 .BYTE 200 :DRIVE 7

;RPO4/5/6 TO RH11 "MASSBUS CONTROL BUS PARITY ERRORS" (MCPE) ALLOWED BEFORE
 ;CALLING IT FATAL (MCPEMX=1 WORD)

012244 000003

MCPEMX: .WORD 3

;STORAGE FOR RPADR (THE FIRST ADDRESS (776700) OF THE RH11/RPO4/5/6),
 ;RPVEC (THE VECTOR ADDRESS (254)), AND RPVEC+2 (THE BR LEVEL (5)).

012246 176700
012250 000254 000240

RPADR: .WORD 176700
 RPVEC: .WORD 254,5*32.

;MAXIMUM NUMBER OF LOOK AHEADS ALLOWED IS 4 (MXLACT=1 WORD)

012254 000004

MXLACT: .WORD 4
 ;MAXIMUM DELTA DELAY IS 8 SECTORS (MXDLTA=1 WORD)

012256 001000

MXDLTA: .WORD 8.*64
 ;MINIMUM DELTA DELAY IS 2 SECTORS (MNDLTA=1 WORD)

012260 000200

MNDLTA: .WORD 2*64.
 ;MAXIMUM SEARCH FOR I/O WINDOW IS 5 SECTORS (MXWNDW=1 WORD)

012262 000005

MXWNDW: .WORD 5

;DEFINITION OF THE RH11/RPO4/5/6 ADDRESS INDEXES

000000
 000002
 000004
 000006
 000010
 000012
 000014
 000016
 000020
 000022
 000024
 000026
 000030
 000032
 000034
 000036
 000040
 000042
 000044
 000046

RPCS1=0
 RPWC=2
 RPBA=4
 RPDA=6
 RPCS2=10
 RPOS1=12
 RPER1=14
 RPAS=16
 RPLA=20
 RPDB=22
 RPMR=24
 RPDT=26
 RPSN=30
 RPOF=32
 RPCA=34
 RPCC=36
 RPER2=40
 RPER3=42
 RPEC1=44
 RPEC2=46

;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00)
 ;WORD COUNT REGISTER (NOT A DRIVE REG)
 ;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)

3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223

012264 104412
012266 013746 177776
012277 012737 000240 177776
012300 004737 020250
012304 012701 012100
012310 012702 012210
012314 005021
012316 020102
012320 101775
012322 012702 012232
012326 012721 177777
012332 020102
012334 101774
012336 005037 012120
012342 005037 012122
012346 005037 012124
012352 005037 012126
012356 013703 012250
012362 012723 015130
012366 013713 012252
012372 013704 012246
012376 012764 000040 000010
012404 005001
012406 004037 012476
012412 000401
012414 000402
012416 105061 012120
012422 005201
012424 042701 177770
012430 001366
012432 012701 000007
012436 005037 177776
012442 105761 012140
012446 001405
012450 004737 017704
012454 105761 012140
012460 001375
012462 005301
012464 100366
012466 012637 177776
012472 104413
012474 000207

```
;RH11/RP04/5/6 DRIVER INITIALIZATION CODE
;THIS ROUTINE WILL DETERMINE WHICH RPO4/5/6 DRIVES ARE
;AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
;TO THE PROPER STATE FOR EACH DRIVE.
;NOTE: THIS ROUTINE CALLS DRVINT

:CALL
:
:JSR PC,RPINIT
:RETURN

:NOTE: THE 'P' OR 'L' CLOCK MUST BE STARTED

RPINIT: SAVREG                ;SAVE R0 - R5
MOV @#PS, -(SP)              ;SAVE THE PRESENT PROCESSOR STATUS
MOV @(<5*32.>, @#PS)          ;CHANGE THE PRIORITY TO 5
JSR PC, CLRQUE                ;CLEAR ALL REQUEST QUEUES
MOV @#RPERRS, R1              ;FIRST ADDRESS TO BE CLEARED
MOV @#SEEKFG, R2              ;LAST ADDRESS TO BE CLEARED
1$: CLR (R1)+                  ;CLEAR
CMP R1, R2                    ;ARE WE DONE?
BLOS 1$                        ;BRANCH IF NO
MOV @DTUW, R2                 ;LAST ADDRESS
2$: MOV @-1, (R1)+            ;INITIALIZE
CMP R1, R2                    ;DONE?
BLOS 2$                        ;LOOP IF NO
CLR DRVSTA                    ;SET ALL DRIVES TO OFFLINE
CLR DRVSTA+2
CLR DRVSTA+4
CLR DRVSTA+6
MOV RPVEC, R3                 ;SETUP THE RH11/RP04/5/6 VECTOR
MOV @ISR, (R3)+
MOV RPVEC+2, (R3)
MOV RPADR, R4                 ;FIRST ADDRESS OF RH11/RP04
MOV @BIT05, R4                ;: MASSBUS INIT
CLR R1                        ;: START WITH DRIVE 0
3$: JSR R0, DRVINT            ;: INIT THE DRIVE
BR 4$                          ;: 'DVA' NOT SET OR PARITY ERROR
BR 5$                          ;: NORMAL RETURN
CLR DRVSTA(R1)                ;: SET DRIVE STATUS TO OFFLINE
INC R1                        ;: GO TO NEXT DRIVE
BIC @C7, R1                   ;: MASK OUT UNUSED BITS
BNE 3$                         ;: BR IF MORE DRIVES TO GO
MOV @#7, R1                   ;: START WITH DRIVE 7
CLR @#PS                      ;: CLEAR THE PROCESSOR STATUS
6$: TSTB DPINT(R1)            ;: WAITING FOR DRIVE TO SWITCH PORTS ?
BEQ 8$                         ;: BR NOT WAITING
JSR PC, SET.IE                ;: SET INTERRUPT
7$: TSTB DPINT(R1)            ;: DRIVE SWITCHED PORTS ?
BNE 7$                         ;: BR IF NOT
DEC R1                        ;: GO TO THE NEXT DRIVE
BPL 6$                         ;: CHECK NEXT DRIVE
MOV (SP)+, @#PS              ;: RESTORE THE PROCESSOR STATUS
RESREG                        ;: RESTORE R0 - R5
RTS PC                         ;: BYE-BYE
```

```

3224
3225 ;DRIVE INITIALIZATION ROUTINE
3226 ;THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
3227 ;AN RPO4/5/6. IF IT IS, A "READ-IN PRESET" IS ISSUED AND FMT22
3228 ;IS SET TO A "1". THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
3229 ;INSURE THEY ARE ALL ON A "1". AND DEPENDING ON THEIR STATE,
3230 ;DRVSTA IS SET TO THE PROPER CONDITION.
3231
3232 ;CALL
3233     MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
3234     MOV     RPADR,R4       ;UNIBUS ADDRESS OF RH11/RPO4/5/6 (RPCS1)
3235     JSR     RO,DRVINT      ;CALLED BY A JSR
3236     RETURN1                ;ERROR OCCURRED (PARITY)
3237     RETURN2                ;NORMAL RETURN
3238
3239     DRVINT: MOV     R5,-(SP) ;SAVE R5
3240             CLR   DRVSTA(R1) ;START DRIVE STATUS AS OFFLINE
3241             CLR   DRVSTYP(R1) ;CLEAR THE DRIVE TYPE INDICATOR
3242             CLR   ULDFLG(R1) ;CLEAR THE UNLOAD FLAG
3243             MOV     R1,RPCS2(R4) ;SELECT A DRIVE
3244             MOV   #111,RPCS1(R4) ;DO A DRIVE CLEAR COMMAND (& SEIZE DRIVE)
3245             BIT   #BIT12,RPCS2(R4) ;NONEXISTENT DRIVE?
3246             BEQ   1$ ;NO---BRANCH
3247             JSR   PC,SET.IE ;GO SET "IE" WITHOUT A "TRE"
3248             BR   6$ ;LEAVE THIS ROUTINE
3249             1$: CLR   DRVSTA(R1) ;SET DRIVE STATUS TO OFFLINE
3250             BIT   #BIT11,RPCS1(R4) ;SEE IF DRIVE AVAILABLE
3251             BEQ   7$ ;BR IF DRIVE NOT AVAILABLE
3252             JSR   RO,RO.RP ;READ THE DRIVE TYPE REG.
3253             RPT   R5
3254             MOV   (SP)+,R5 ;ERROR RETURN ADDRESS
3255             MOV   #1,DRVSTYP(R1) ;PUT DRIVE TYPE IN R5
3256             CMP   #20020,R5 ;SET RPO4 INDICATOR
3257             BEQ   2$ ;IS IT A SINGLE PORT RPO4?
3258             BR   2$ ;BRANCH IF YES
3259             CMP   #24020,R5 ;IS IT A DUAL PORT RPO4?
3260             BEQ   2$ ;BR IF YES
3261             MOV   #2,DRVSTYP(R1) ;SET RPO5 INDICATOR
3262             CMP   #20021,R5 ;SINGLE PORT RPO5 ?
3263             BEQ   2$ ;BR IF YES
3264             CMP   #24021,R5 ;DUAL PORT RPO5 ?
3265             BEQ   2$ ;BR IF YES
3266             MOV   #4,DRVSTYP(R1) ;SET RPO6 INDICATOR
3267             CMP   #20022,R5 ;SINGLE PORT RPO6 ?
3268             BEQ   2$ ;BR IF YES
3269             CMP   #24022,R5 ;DUAL PORT RPO6 ?
3270             BEQ   2$ ;BR IF YES
3271             MOV   #-1,DRVSTYP(R1) ;SET INDICATOR TO 'OTHER'
3272             BR   6$ ;EXIT
3273             2$: MOV   #121,-(SP) ;DO A "READ-IN PRESET"
3274             JSR   RO,WRT.RP
3275             RPT   R5
3276             MOV   #BIT12,-(SP) ;SET FMT22=1
3277             JSR   RO,WRT.RP
3278             RPT   R5
3279

```

K05

```

3280 012716 013030          8$
3281 012720 004037 017214 JSR      RO,RO.RP      ;READ RPDS1
3282 012724 000012          RPDS1
3283 012726 013030          8$
3284 012730 012605          MOV      (SP)+,R5      ;AND SAVE IT IN R5
3285 012732 100015          BPL      4$            ;BRANCH IF ATA=0
3286 012734 116164 012234 000016 MOVB     ATABIT(R1),RPAS(R4) ;CLEAR ATTENTION BIT
3287 012742 004037 017214 JSR      RO,RO.RP      ;FIND OUT WHY ATA=1
3288 012746 000014          RPER1
3289 012750 013030          8$
3290 012752 006126          ROL      (SP)+        ;IS IT UNSAFE?
3291 012754 100004          BPL      4$            ;BR IF NOT
3292 012756 112761 177777 012120 MOVB     #-1,DRVSTA(R1) ;SET UNSAFE INDICATOR
3293 012764 000407          BR       6$            ;EXIT
3294 012766 005105          4$: COM      R5          ;CHECK MOL, DPR, DRY, AND VV
3295 012770 042705 167077 BIC      #1<BIT12:BIT08:BIT07:BIT06>,R5
3296 012774 001003          BNE      6$            ;BRANCH IF MOL, DPR, DRY, OR VV IS CLEAR
3297 012776 112761 000001 012120 MOVB     #1,DRVSTA(R1) ;SET DRIVE STATUS TO ONLINE
3298 013004 005720          6$: TST      (R0)+       ;STEP OVER THE ERROR RETURN
3299 013006 000410          BR       8$            ;EXIT
3300 013010 006301          7$: ASL      R1          ;CHANGE INDEX TO ADDRESS WORDS
3301 013012 012761 003720 012212 MOV      #2000.,TIMER(R1) ;START 2 SEC TIMER
3302 013020 006201          ASR      R1            ;RESTORE R1
3303 013022 105161 012140 COMB     DPINT(F')      ;SET PORT INITIALIZE INIDICATOR
3304 013026 005720          TST      (R0)+
3305 013030 012605          8$: MOV      (SP)+,R5      ;RESTORE R5
3306 013032 000200          RTS      RO            ;EXIT
3307
3308 ;REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
3309
3310 ;CALL
3311
3312 ;
3313 JSR      RO,#RPO4      ;CALL THE RPO4/5/6 DRIVER
3314 PNTADR   ;ADDRESS OF POINTER OF DRIVES PARAMETER BLOCK
3315 RETURN1  ;RETURN HERE IF QUEUE IS FULL
3316 RETURN2 ;RETURN HERE IF REQUEST IS IN QUEUE OR THERE
3317 ;IS AN ERROR CONDITION
3318
3319 RPO4: MOV      @#PS,-(SP) ;SAVE THE CALLING STATUS
3320 MOV      RPVEC+2,@#PS ;DON'T ALLOW ANY RPO4/5/6 INTERRUPTS
3321 MOVB     #1,ACTDRV    ;SET "ACTIVE DRIVER" FLAG
3322 SAVREG   ;SAVE R0 - R5
3323 MOV      (R0),R2      ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
3324 CLR      16(R2)       ;CLEAR THE STATUS/ERROR INDICATOR
3325 MOVB     (R2),R1      ;PICKUP THE DRIVE NUMBER
3326 MOV      RPADR,R4     ;UNIBUS ADDRESS OF RPCS1
3327 TSTB    DRVSTA(R1)   ;CHECK DRIVES STATUS
3328 BGT      1$           ;BRANCH IF ONLINE
3329 TSTB    ULDFLG(R1)   ;UNLOAD COMMAND IN QUEUE?
3330 BNE      3$           ;BRANCH IF YES
3331 TSTB    DPINT(R1)    ;TRYING TO INIT THE DRIVE
3332 BNE      5$           ;BR IF YES
3333 JSR      RO,DRVINT    ;GO INIT. THE DRIVE
3334 BR       4$           ;ERROR RETURN
3335 TSTB    DRVSTA(R1)   ;IS DRIVE STATUS ONLINE?
3336 BLE     6$           ;BR IF NOT

```

L05

CZRJBBO, RPO4/5/6 FMTR MACY11 30(1046)
 CZJBB.P11 04-NOV-77 12:54

07-NOV-77 10:07 PAGE 64
 SINGLE/DUAL PORT RH11/RPO4/5/6 DRIVER (REV 1.0)

SEQ 0063

```

3336 013130 105761 012150 1$: TSTB DPRQS(R1) ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
3337 013134 001031 BNE 5$ ;BR IF YES
3338 013136 010164 000010 MOV R1,RPCS2(R4) ;SELECT THE DRIVE
3339 013142 004037 020346 JSR RO,DRVQUE ;PUT THIS REQUEST IN QUEUE
3340 013146 000460 BR 9$ ;QUEUE IS FULL
3341 013150 122762 000103 000002 CMPB #103,2(R2) ;IS THIS REQ. FOR AN UNLOAD?
3342 013156 001003 BNE 2$ ;BR IF NO
3343 013160 112761 177777 012166 MOVB #-1,ULDFLG(R1) ;SET THE "UNLOAD IN QUEUE" FLAG
3344 013166 105761 012110 2$: TSTB DRVACT(R1) ;IS THIS DRIVE ACTIVE?
3345 013172 001043 BNE 8$ ;BR IF YES
3346 013174 004737 013326 JSR PC,OPT ;CALL THE OPTIMIZER
3347 013200 000440 BR 8$
3348 013202 012762 120000 000016 3$: MOV #BIT15!BIT13,16(R2) ;SET THE "UNLOAD IN QUEUE" ERROR FLAG
3349 013210 000434 BR 8$ ;EXIT
3350 013212 004737 014436 4$: JSR PC,CI7 ;GO HANDLE THE PARITY ERROR
3351 013216 000431 BR 8$
3352 013220 004037 020346 5$: JSR RO,DRVQUE ;PUT REQUEST IN QUEUE
3353 013224 000431 BR 9$ ;QUEUE IS FULL
3354 013226 032714 000100 BIT #BIT06,(R4) ;IS 'IE' SET ALREADY ?
3355 013232 001023 BNE 8$ ;BR IF IT IS
3356 013234 004737 017704 JSR PC,SET.IE ;SET INTERRUPT
3357 013240 000420 BR 8$ ;RETURN, REQUEST IN QUEUE
3358 013242 105761 012120 6$: TSTB DRVSTA(R1) ;SEE IF DRIVE OFFLINE OR UNSAFE
3359 013246 002412 BLT 7$ ;BR IF UNSAFE
3360 013250 012762 140000 000016 MOV #BIT15!BIT14,16(R2) ;SET OFFLINE ERROR INDICATOR
3361 013256 105761 012130 TSTB DRVTP(R1) ;SEE IF OFFLINE OR NONEXISTENT
3362 013262 001007 BNE 8$ ;BR IF OFFLINE
3363 013264 012762 100002 000016 MOV #BIT15!BIT01,16(R2) ;REPORT DRIVE NONEXISTENT
3364 013272 000403 BR 8$ ;GO TO EXIT
3365 013274 012762 110000 000016 7$: MOV #BIT15!BIT12,16(R2) ;DRIVE IS UNSAFE
3366 013302 104413 8$: RESREG ;RESTORE R0 - R5
3367 013304 005720 TST (R0)+ ;SETUP FOR NORMAL RETURN
3368 013306 000401 BR 10$ ;FINISH UP, THEN EXIT
3369 013310 104413 9$: RESREG ;RESTORE R0 - R5
3370 013312 005720 10$: TST (R0)+ ;CORRECT THE RETURN ADDRESS
3371 013314 105037 012164 CLRB ACTDRV ;CLEAR "ACTIVE DRIVER" FLAG
3372 013320 012637 177776 MOV (SP)+,2#PS ;RETURN "PS" TO USER LEVEL
3373 013324 000200 RTS ;RETURN TO CALLER
3374
3375 ;OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
3376
3377 ;CALL
3378 ;
3379 ;
3380 ;
3381 OPT: SAVREG ;SAVE R0 - R5
3382 013330 013746 177776 MOV 2#PS, -(SP) ;SAVE PROC. STATUS
3383 013334 146137 012234 012162 BICB ATABIT(R1),SRCHWT ;CLEAR "SEARCH WAIT" KEY
3384 013342 004737 020422 JSR PC,GETREQ ;GET "DPB" POINTER OF REQUEST
3385 013346 005702 TST R2 ;IS THERE A REQUEST IN QUEUE?
3386 013350 001505 BEQ 7$ ;NO--BRANCH TO EXIT
3387 013352 032764 004000 000000 BIT #BIT11,RPCS1(R4) ;IS UVA SET?
3388 013360 001407 BEQ 10$ ;BRANCH IF NOT
3389 013362 032764 000100 000012 BIT #BIT6,RPCS1(R4) ;IS VV SET ?
3390 013370 001003 BNE 10$ ;BR IF IT IS
3391 013372 004037 012476 9$: JSR RO,DRVINT ;SEE IF DRIVE STILL ONLINE ?

```

M05

CZRJPC0, RPO4/5/6 FMTR MACY1: 30(1046) 07-NOV-77 10:07 PAGE 65
 CZRJBB.P11 04-NOV-77 12:54

SINGLE/DUAL PORT RH11/RPO4/5/6 DRIVER (REV ' 0)

SEQ 0064

```

3392 013376 000470          BR          6$          ;PARITY ,VA' NOT SET
3393 013400 105761 012120 10$: TSTB     DRVSTA(R1)  ;IS DRIVE ,LINE?
3394 013404 003014          BGT          1$          ;YES--BRANCH
3395 013406 004737 020444          JSR     PC,POPQUE  ;NO--REMOVE REQUEST FROM QUEUE
3396 013412 012762 140000 000016  MOV     #BIT15:BIT14,16(R2) ;SET OFFLINE STATUS/ERROR INDICATOR
3397 013420 105761 012120          TSTB     DRVSTA(R1)  ;IS DRIVE UNSAFE ?
3398 013424 100064          BPL          8$          ;BR TO EXIT IF NOT
3399 013426 012762 110000 000016  MOV     #BIT15:BIT12,16(R2) ;SET UNSAFE STATUS/ERROR INDICATOR
3400 013434 000460          BR          8$          ;BRANCH TO EXIT
3401 013436 012746 000111 1$: MOV     #111,-(SP)  ;LOAD COMMAND ONTO THE STACK
3402 013442 004037 017374          JSR     RO,WRT.RP   ;LOAD THE REGISTER
3403 013446 000000          RPCS1         ;REGISTER INCREMENT
3404 013450 013560          6$          ;ERROR RETURN ADDRESS
3405 013452 032714 004000          BIT     #BIT11,(R4)  ;DRIVE AVAILABLE ?
3406 013456 001427          BEQ          5$          ;BR IF NOT
3407 013460 122762 000150 000002  CMPB     #150,2(R2)  ;IS THE REQUEST FOR I/O?
3408 013466 002403          BLT          2$          ;YES--BRANCH
3409 013470 004737 014022          JSR     PC,C14      ;CALL THE COMMAND INITIATOR
3410 013474 000440          BR          8$          ;BRANCH TO EXIT
3411 013476 005737 012232 2$: TST     DTUW        ;DATA TRANSFER UNDERWAY?
3412 013502 002012          BGE          4$          ;YES--GO START A SEARCH
3413 013504 005737 012210          TST     SEEKFG      ;DO IMPLIED SEEKS?
3414 013510 100404          BMI          3$          ;YES---BRANCH
3415 013512 004037 014772          JSR     RO,LA       ;NO--DO LOOK AHEAD
3416 013516 000427          BR          8$          ;RETURN HERE ON A PARITY ERROR
3417 013520 000403          BR          4$          ;GO START A SEARCH
3418 013522 004737 013606 3$: JSR     PC,C11      ;START A DATA TRANSFER
3419 013526 000423          BR          8$          ;
3420 013530 004737 013714 4$: JSR     PC,C13      ;START A SEARCH
3421 013534 000420          BR          8$          ;GO TO THE EXIT
3422 013536 112761 177777 012150 5$: MOV     #-1,DPRQS(R1) ;SET PORT REQUEST INDICATOR
3423 013544 010103          MOV     R1,R3      ;SET UP TO ADDRESS WORDS
3424 013546 006303          ASL     R3          ;CONVERT TO WORD INDEX
3425 013550 012763 023420 012212  MOV     #10000.,TIMER(R3) ;START 10 SEC TIMER
3426 013556 000402          BR          7$          ;EXIT
3427 013560 004737 014436 6$: JSR     PC,C17      ;PROCESS THE PARITY ERROR
3428 013564 032714 000100 7$: BIT     #BIT06,(R4)  ;SEE IF 'IE' ALREADY SET
3429 013570 001002          BNE          8$          ;BR IF SET
3430 013572 004737 017704          JSR     PC,SET.IE   ;SET "IE" WITHOUT A "TRE"
3431 013576 012637 177776 8$: MOV     (SP)+,2#PS  ;RESTORE PROC. STATUS
3432 013602 104413          RESREG        ;RESTORE RO - R5
3433 013604 000207          RTS     PC
3434
3435          ;COMMAND INITIATOR
3436          ;CALL
3437          ;
3438          MOV     #DRVNUM,R1 ;DRIVE NUMBER
3439          MOV     #DPB,R2    ;ADDRESS OF DPB
3440          JSR     PC,C1?    ;C1?= C11,C13, OR C14
3441          ;WHERE:
3442          ;C11=DATA TRANSFER
3443          ;C12=SEARCH REQUESTED BY DATA XFER
3444          ;C14=NOT DATA TRANSFER
3445
3446 013606 004737 020444 C11: JSR     PC,POPQUE  ;REMOVE REQUEST FROM "DRIVES WAIT" QUEJE
3447 013612 010237 012160          MOV     R2,TRNSWT  ;PUT REQ. IN TRANSFER WAIT QUEJE

```

NOS

CZRJB80 RPO4/5/6 FMTR MACY11 30(1046) U7-NOV-77 10:07 PAGE 66
 CZRJB8.P11 04-NOV-77 12:54 SINGLE/DUAL PORT RH11/RPO4/5/6 DRIVER (REV 1.0)

SEQ 0065

3448	013616	010203			MOV	R2,R3	;DPB ADDRESS TO R3
3449	013620	013704	012246		MOV	RPADR,R4	;RPCS1 ADDRESS
3450	013624	010164	000010		MOV	R1,RPCS2(R4)	;SELECT DRIVE
3451	013630	062703	000004		ADD	#4,R3	;DESIRED WORD COUNT
3452	013634	062704	000002		ADD	#2,R4	;RPWC ADDRESS
3453	013640	012324			MOV	(R3)+,(R4)+	;LOAD WORD COUNT
3454	013642	012324			MOV	(R3)+,(R4)+	;LOAD BUFFER ADDRESS
3455	013644	012346			MOV	(R3)+,-(SF)	;LOAD SECTOR AND TRACK
3456	013646	004037	017374		JSR	RO,WRT.RP	;CALL THE LOAD(WRITE) ROUTINE
3457	013652	000006			RPDA		;INDEX OF REGISTER TO LOAD
3458	013654	014436			CI7		;ERROR RETURN ADDRESS
3459	013656	012346			MOV	(R3)+,-(SP)	;LOAD CYLINDER ADDRESS
3460	013660	004037	017374		JSR	RO,WRT.RP	
3461	013664	000034			RPCF		
3462	013666	014436			CI7		
3463	013670	016246	000002		MOV	2(R2),-(SP)	;LOAD "COMMAND+GO", "A17&A16", AND "PSEL"
3464	013674	004037	017374		JSR	RO,WRT.RP	
3465	013700	000000			RPCS1		
3466	013702	014436			CI7		
3467	013704	010137	012232		MOV	R1,DTUW	;SET "DATA TRANSFER UNDERWAY"
3468	013710	000137	014400		JMP	CI5	
3469	013714	013704	012246		MOV	RPADR,R4	;RPCS1 ADDRESS
3470	013720	010164	000010		MOV	R1,RPCS2(R4)	;SELECT DRIVE
3471	013724	016246	000012		MOV	12(R2),-(SP)	;DESIRED CYLINDER ADDRESS
3472	013730	004037	017374		JSR	RO,WRT.RP	
3473	013734	000034			RPCA		
3474	013736	014436			CI7		
3475	013740	116203	000010		MOV	10(R2),R3	;PICKUP SECTOR ADDRESS
3476	013744	163703	012262		SUB	MXWIND,R3	;BACKUP BY MAX. SEARCH FOR I/O WINDOW
3477	013750	002002			BGE	1\$	
3478	013752	062703	000026		ADD	#22,R3	
3479	013756	010346			MOV	R3,-(SP)	;COMBINE THE ADJUSTED SECTOR WITH
3480	013760	116266	000011	000001	MOV	11(R2),1(SP)	;THE DESIRED TRACK
3481	013766	004037	017374		JSR	RO,WRT.RP	;LOAD DESIRED TRACK & SECTOR
3482	013772	000006			RPDA		
3483	013774	014436			CI7		
3484	013776	012746	000131		MOV	#131,-(SP)	;START A SEARCH
3485	014002	004037	017374		JSR	RO,WRT.RP	
3486	014006	000000			RPCS1		
3487	014010	014436			CI7		
3488	014012	156137	012234	012162	BISB	ATABIT(R1),SRCHWT	;SET "SEARCH WAIT" KEY
3489	014020	000567			BR		
3490	014022	013704	012246		MOV	RPADR,R4	;RPCS1 ADDRESS
3491	014026	010164	000010		MOV	R1,RPCS2(R4)	;SELECT DRIVE
3492	014032	116203	000002		MOV	2(R2),R3	;PICKUP THE REQUESTED COMMAND
3493	014036	122703	000131		CMPB	#131,R3	;IS IT A SEARCH COMMAND?
3494	014042	001007			BNE	1\$;BRANCH IF NO
3495	014044	016246	000010		MOV	10(R2),-(SP)	;LOAD DESIRED TRACK & SECTOR
3496	014050	004037	017374		JSR	RO,WRT.RP	
3497	014054	000006			RPDA		
3498	014056	014436			CI7		
3499	014060	000403			BR		
3500	014062	122703	000105		CMPB	#105,R3	;GO LOAD CYLINDER
3501	014066	001007			BNE	3\$;IS IT A SEEK COMMAND
3502	014070	016246	000012		MOV	12(R2),-(SP)	;BRANCH IF NO
3503	014074	004037	017374		JSR	RO,WRT.RP	;LOAD DESIRED CYLINDER

3504	014100	000034			RPCA			
3505	014102	014436			CI7			
3506	014104	000546			BR	CI6		
3507	014106	122703	000115	3\$:	CMPB	#115,R3		: IS IT AN "OFFSET" COMMAND?
3508	014112	001013			BNE	4\$: BR IF NO
3509	014114	004037	017214		JSR	RO, RD.RP		: MERGE THE OFFSET VALUE INTO RPOF
3510	014120	000032			RPOF			: BUT DON'T CHANGE THE UPPER
3511	014122	014436			CI7			
3512	014124	116216	000001		MOVB	1(R2), (SP)		: BYTE WHEN LOADING THE
3513	014130	004037	017374		JSR	RO, WRT.RP		: REGISTER (RPOF)
3514	014134	000032			RPOF			
3515	014136	014436			CI7			
3516	014140	000530			BR	CI6		: GO START THE COMMAND
3517	014142	122703	000107	4\$:	CMPB	#107,R3		: IS IT A "RECALIBRATE" COMMAND?
3518	014146	001525			BEQ	CI6		: BRANCH IF YES
3519	014150	122703	000117		CMPB	#117,R3		: IS IT A RETURN TO CENTER?
3520	014154	001522			BEQ	CI6		: BRANCH IF YES
3521	014156	122703	000103		CMPB	#103,R3		: IS IT AN "UNLOAD" COMMAND?
3522	014162	001016			BNE	5\$: BRANCH IF NO
3523	014164	112761	000001	012110	MOVB	#1, DRVACT(R1)		: SET THE DRIVE ACTIVE INDICATOR
3524	014172	105061	012120		CLRB	DRVSTA(R1)		: PUT DRIVE STATUS TO OFFLINE
3525	014176	112761	000001	012166	MOVB	#1, ULDFLG(R1)		: SET "UNLOAD IN PROGRESS" FLAG
3526	014204	010346			MOV	R3, -(SP)		: START THE "UNLOAD" COMMAND
3527	014206	004037	017374		JSR	RO, WRT.RP		
3528	014212	000000			RPCS1			
3529	014214	014436			CI7			
3530	014216	000207			RTS	PC		: RETURN TO USER
3531	014220	122703	000143	5\$:	CMPB	#143,R3		: IS IT A "SET FORMAT" COMMAND?
3532	014224	001014			BNE	6\$: BRANCH IF NO
3533	014226	004037	017214		JSR	RO, RD.RP		: READ THE OFFSET REGISTER
3534	014232	000032			RPOF			
3535	014234	014436			CI7			
3536	014236	116266	000001	000001	MOVB	1(R2), 1(SP)		: COMBINE "FMT22" "ECI" AND "HCI"
3537	014244	004037	017374		JSR	RO, WRT.RP		: LOAD "FMT22", "ECI", AND/OR "HCI".
3538	014250	000032			RPOF			
3539	014252	014436			CI7			
3540	014254	000436			BR	12\$		
3541	014256	122703	000141	6\$:	CMPB	#141,R3		: IS IT A "GET REGISTER" COMMAND?
3542	014262	001023			BNE	10\$: BRANCH IF NO
3543	014264	016203	000006	7\$:	MOV	6(R2), R3		: POINTS TO 1ST ADDRESS OF WHERE
3544								: TO PUT THE REGISTER(S)
3545	014270	116237	000010	014306	MOVB	10(R2), 9\$: INIT. THE INDEX FOR THE FIRST REG.
3546	014276	116205	000011		MOVB	11(R2), R5		: INDEX OF LAST REG. TO MOVE
3547	014302	004037	017214	8\$:	JSR	RO, RD.RP		: READ RPO4/5/6 REGISTER
3548	014306	000000		9\$:	RPCS1			: INDEX OF REG. TO READ
3549	014310	014436			CI7			
3550	014312	012623			MOV	(SP)+, (R3)+		: GET THE CONTENTS OF RH11/RPO4/5/6 REG.
3551	014314	023705	014306		CMP	9\$ R5		: LAST REG. BEEN READ?
3552	014320	001414			BEQ	12\$: GET OUT IF YES
3553	014322	062737	000002	014306	ADD	#2, 9\$: INCREASE THE INDEX BY 2
3554	014330	000764			BR	8\$: LOOP--MORE TO READ
3555	014332	122703	000145	10\$:	CMPB	#145,R3		: IS IT A "SELECT DRIVE" COMMAND?
3556	014336	001405			BEQ	12\$: BRANCH IF YES
3557	014340	010346		11\$:	MOV	R3, -(SP)		: LOAD THE COMMAND
3558	014342	004037	017374		JSR	RO, WRT.RP		
3559	014346	000000			RPCS1			

```

3560 014350 014436          CI7
3561 014352 004737 020444 12$: JSR    PC,POPQUE      ;REMOVE REQ. FROM QUEUE
3562 014356 052762 000200 000016 BIS    #BIT07,16(R2)    ;SET THE "DONE" BIT
3563 014364 005737 012206 TST    SAVEFG          ;SAVE THE RH11/RPO4/5/6 REGISTERS?
3564 014370 100002          BPL    13$             ;BRANCH IF NO
3565 014372 004737 017566 JSR    PC,SVRH11       ;YES--GO SAVE THE REGISTERS
3566 014376 000207          RTS    PC              ;RETURN TO USER
3567 014400 006301          CI5: ASL    R1              ;
3568 014402 012761 001750 012212 MOV    #1000.,TIMER(R1) ;SET A ONE SECOND TIMER
3569 014410 006201          ASR    R1              ;
3570 014412 112761 000001 012110 MOVB   #1,DRVACT(R1)   ;SET THE DRIVE ACTIVE
3571 014420 000207          RTS    PC              ;RETURN TO THE USER
3572 014422 010346          CI6: MOV    R3,-(SP)    ;LOAD THE COMMAND
3573 014424 004037 017374 JSR    RO,WRT.RP
3574 014430 000000          RPCS1
3575 014432 014436          CI7
3576 014434 000761          BR    CI5              ;
3577 014436 032764 010000 000010 CI7: BIT    #BIT12,RPCS2(R4) ;DRIVE NON-EXISTENT ?
3578 014444 001034          BNE   CI8              ;BR IF YES
3579 014446 005702          1$: TST    R2           ;ANYTHING IN QUEUE ?
3580 014450 001405          BEQ   CI7B            ;BR IF NOT
3581 014452 012762 104000 000016 MOV    #BIT15!BIT11,16(R2) ;SET "PARITY" ERROR INDICATOR
3582 014460 004737 017566 JSR    PC,SVRH11       ;GO SAVE THE RH11/RPO4/5/6 REGISTERS
3583 014464 012746 000111 CI7B: MOV    #11,-(SP)    ;DO A "DRIVE CLEAR"
3584 014470 004037 017374 JSR    RO,WRT.RP
3585 014474 000000          RPCS1
3586 014476 014536          CI8
3587 014500 004737 020326 JSR    PC,EMPTYQ      ;EMPTY THE QUEUE
3588 014504 105061 012166 CLR    ULDFLG(R1)     ;CLEAR THE UNLOAD IN QUEUE FLAG
3589 014510 105061 012110 CLR    DRVACT(R1)     ;DRIVE IS IDLE
3590 014514 020137 012232 CMP    R1,DTUW        ;IF THIS DRIVE HAD AN I/O REQUEST
3591 014520 001005          1$: BNE   1$           ;IN PROGRESS CLEAR ALL OF THE FLAGS
3592 014522 005037 012160 CLR    TRNSWT
3593 014526 012737 177777 CI2232 MOV    #-1,DTUW
3594 014534 000207          1$: RTS    PC
3595 014536 104412          CI8: SAVREG ;SAVE R0 - R5
3596 014540 032764 010000 000010 BIT    #BIT12,RPCS2(R4) ;IS 'NED' SET ?
3597 014546 001002          1$: BNE   1$           ;BR IF YES
3598 014550 005001          CLR    R1
3599 014552 005003          CLR    R3
3600 014554 105761 012110 1$: TSTB   DRVACT(R1)   ;DRIVE ACTIVE?
3601 014560 001443          BEQ   5$              ;BRANCH IF NO
3602 014562 013702 012160 MOV    TRNSWT,R2      ;GET THE "TRANSFER WAIT" QUEUE
3603 014566 020137 012232 CMP    R1,DTUW        ;DID THIS DRIVE HAVE AN I/O IN PROGRESS?
3604 014572 001402          BEQ   2$              ;BRANCH IF YES
3605 014574 004737 020422 JSR    PC,GETREG      ;GET THE DPB POINTER
3606 014600 005702          2$: TST    R2           ;QUEUE ENTRY FOR DRIVE ?
3607 014602 001413          BEQ   4$              ;BR IF NOT
3608 014604 032764 010000 000010 BIT    #BIT12,RPCS2(R4) ;'NED' SET ?
3609 014612 001404          BEQ   3$              ;BR IF NOT
3610 014614 012762 100002 000016 MOV    #BIT15!BIT01,16(R2) ;SET 'DRIVE NON-EXISTENT' INDICATOR
3611 014622 000405          BR    4$              ;CONTINUE
3612 014624 012762 102000 000016 3$: MOV    #BIT15!BIT10,16(R2) ;SET "NON-CLEARABLE PARITY" ERROR INDICATOR
3613 014632 004737 017566 JSR    PC,SVRH11       ;SAVE RH11/RPO4/5/6 REGISTERS
3614 014636 012763 177777 012212 4$: MOV    #-1,TIMER(R3)  ;STOP THE TIMER
3615 014644 105061 012110 CLR    DRVACT(R1)     ;SET "DRIVE ACTIVE" TO IDLE
    
```

```

3616 014650 020137 012232          CMP      R1,DTUW          ; IS THIS DRIVE SETUP FOR A TRANSFER
3617 014654 001005          BNE      5$              ; BR IF NOT
3618 014656 012737 177777 012232        MOV      #-1,DTUW        ; RESET THE INDICATOR
3619 014664 005037 012160          CLR      TRNSWT         ; CLEAR THE TRANSFER QUEUE
3620 014670 105061 012166          CLR      ULDFLC(R1)     ; CLEAR UNLOAD FLAG
3621 014674 032764 010000 000010    5$:     BIT      #BIT12,RPCS2(R4) ; 'NED' SET ?
3622 014702 001021          BNE      6$              ; BR IF YES
3623 014704 005201          INC      R1              ; MOVE TO THE NEXT DRIVE
3624 014706 062703 000002        ADD      #2,R3
3625 014712 042701 177770        BIC      #1C7,R1
3626 014716 001316          BNE      1$              ; BRANCH IF MORE DRIVES
3627 014720 012737 177777 012232        MOV      #-1,DTUW        ; NO DATA TRANSFERS UNDERWAY
3628 014726 005037 012160          CLR      TRNSWT         ; CLEAR THE 'TRANSFER WAIT' QUEUE
3629 014732 004737 020250          JSR      PC,CLRQUE      ; CLEAR ALL OF THE REQUEST QUEUES
3630 014736 012764 000040 000010    MOV      #BIT05,RPCS2(R4) ; DO A MASSBUS INIT.
3631 014744 000406          BR       7$              ; CONTINUE
3632 014746 004737 020326          JSR      PC,EMPTYQ      ; CLEAR THE DRIVE'S QUEUE
3633 014752 105061 012120          CLR      DRVSTA(R1)     ; SET DRIVE TO OFFLINE
3634 014756 105061 012130          CLR      DRVTP(R1)     ; CLEAR THE DRIVE TYPE INDICATOR
3635 014762 004737 017704          JSR      PC,SET.IE      ; SET "IE" WITHOUT "TRE"
3636 014766 104413          RESREG                    ; RESTORE R0 - R5
3637 014770 000207          RTS      PC              ; RETURN
3638
3639          ; LOOK AHEAD ROUTINE
3640
3641          ; CALL
3642          ;
3643          ;     MOV      #DRVNUM,R1          ; DRIVE NUMBER
3644          ;     MOV      #DPB,R2            ; POINT TO DPB
3645          ;     JSR      RO,LA              ; GO CHECK THE WINDOW
3646          ;     RETURN1                     ; ERROR RETURN
3647          ;     RETURN2                     ; START A SEARCH
3648          ;     RETURN3                     ; START A DATA TRANSFER
3649
3649 014772 013704 012246          LA:     MOV      RPADR,R4          ; GET RPCS1'S ADDRESS
3650 014776 010164 000010          MOV      R1,RPCS2(R4)   ; SELECT DRIVE
3651 015002 004037 017214          JSR      RO,RO.RP       ; READ CURRENT CYLINDER
3652 015006 000036          RPCC
3653 015010 015122          4$
3654 015012 022662 000012        CMP      (SP)+,12(R2)   ; IS CURRENT CYLINDER=DESIRED
3655          ; CYLINDER?
3656 015016 001037          BNE      3$              ; EXIT IF NO
3657 015020 105261 012176          INCB    LACNT(R1)       ; INCREMENT THE LOOK AHEAD COUNT
3658 015024 126137 012176 012254        CMPB    LACNT(R1),MXLACT ; EXCEED MAX?
3659 015032 003026          BGT      2$              ; BRANCH IF YES
3660 015034 116203 000010          MOV      10(R2),R3      ; GET DESIRED SECTOR ADDRESS AND
3661 015040 000303          SWAB    R3              ; MULT. BY 64--ALIGN WITH
3662 015042 006203          ASR      R3              ; LOOK AHEAD REGISTER
3663 015044 006203          ASR      R3
3664 015046 012737 000340 177776        MOV      #340,2#PS      ; PRIORITY LEVEL "7"
3665 015054 004037 017214          JSR      RO,RO.RP       ; READ LOOK AHEAD REGISTER
3666 015060 000020          RPLA
3667 015062 015122          4$
3668 015064 162603          SUB      (SP)+,R3       ; CALCULATE THE DELTA
3669 015066 002002          BGE      1$              ; MAKE THE DELTA POSITIVE
3670 015070 062703 002600          ADD      #122,*64.,R3   ;
3671 015074 023703 012256          1$:     CMP      MXDLTA,R3   ; CHECK THE DELTA TO SEE
    
```

```

3672 015100 002406          BLT      3$          ;IF IT IS WITHIN THE
3673 015102 023703 012260    CMP      MNDLTA,R3   ;WINDOW---IF YES, ZERO
3674 015106 002003          BGE      3$          ;THE LOOK AHEAD COUNT
3675 015110 105061 012176    2$:     CLRB     LACNT(R1) ;AND TAKE THE I/O EXIT
3676 015114 005720          TST      (R0)+
3677 015116 005720          3$:     TST      (R0)+   ;ADJUST THE RETURN ADDRESS
3678 015120 000402          BR       5$          ;EXIT
3679 015122 004737 014436    4$:     JSR      PC,C17  ;PROCESS THE ERROR
3680 015126 000200          5$:     RTS       RO     ;RETURN
3681
3682          ;INTERRUPT SERVICE ROUTINE
3683
3684 015130 112737 000001 012164  ISR:     MOV      #1,ACTDRV   ;SET "ACTIVE DRIVER" FLAG
3685 015136 104412          SAVREG
3686 015140 013704 012246    MOV      RPADR,R4   ;SAVE RO - R5
3687 015144 013701 012232    MOV      DTUW,R1   ;ADDRESS OF RHSCS1
3688 015150 002403          BLT      1$          ;GET "DATA TRANSFER UNDERWAY" INDICATOR
3689 015152 004737 015174    JSR      PC,TD      ;BRANCH IF NO DATA TRANSFER UNDERWAY
3690 015156 000402          BR       2$          ;CALL TRANSFER DONE
3691 015160 004737 015334    1$:     JSR      PC,SC   ;EXIT
3692 015164 104413          2$:     RESREG
3693 015166 105037 012164    CLRB     ACTDRV    ;CALL SPECIAL CONDITIONS
3694 015172 000002          RTI              ;RESTORE RO - R5
3695          ;CLEAR "ACTIVE DRIVER" FLAG
3696          ;RETURN
3697          ;TRANSFER DONE ROUTINE
3698 015174 105061 012110    TD:     CLRB     DRVACT(R1) ;SET DRIVE ACTIVE INDICATOR TO IDLE
3699 015200 012737 177777 012232    MOV      #-1,DTUW  ;NO DATA TRANSFERS UNDERWAY
3700 015206 006301          ASL      R1
3701 015210 012761 177777 012212    MOV      #-1,TIMER(R1) ;CANCEL TIMEOUT
3702 015216 006201          ASR      R1
3703 015220 013702 012160    MOV      TRANSW,R2 ;GET "DPB" ADDRESS FROM THE
3704 015224 005037 012160    CLR      TRANSW    ;TRANSFER WAIT QUEUE--CLEAR QUEUE
3705 015230 052762 000200 000016    BIS      #BIT07,16(R2) ;SET DONE
3706 015236 010164 000010    MOV      R1,RPCS2(R4) ;SELECT THE DRIVE
3707 015242 004037 017214    JSR      RO,RD.PP  ;TRANSFER ERROR(TRE=1)?
3708 015246 000000          RPCS1
3709 015250 014436          CI7
3710 015252 006126          ROL      (SP)+
3711 015254 100413          BMI      3$          ;BR IF YES
3712 015256 005737 012206    TST      SAVEFG    ;SAVE THE RH11/RPO4/5/6 REGISTERS?
3713 015262 100002          BPL      1$          ;BRANCH IF NO
3714 015264 004737 017566    JSR      PC,SVRH11 ;YES--SAVE THE REGISTERS
3715 015270 004737 013326    1$:     JSR      PC,OPT  ;CALL OPTIMIZER
3716 015274 000417          BR       SC          ;CHECK OTHER DRIVES
3717 015276 012714 000113    2$:     MOV      #113,(R4) ;RELEASE THE DRIVE
3718 015302 000414          BR       SC          ;CHECK FOR OTHER DRIVES
3719 015304 052762 100100 000016    3$:     BIS      #BIT15!BIT06,16(R2) ;SET DATA ERROR FLAG
3720 015312 004737 020326    JSR      PC,EMPTYQ ;EMPTY THE "DRIVE'S WAIT" QUEUE
3721 015316 004737 017566    JSR      PC,SVRH11 ;SAVE THE RH11/RPO4/5/6 REGISTERS
3722 015322 012714 040111    MOV      #40111,(R4) ;ISSUE A "DRIVE CLEAR"
3723 015326 012714 000113    MOV      #113,(R4)  ;ISSUE A RELEASE TO THE DRIVE
3724 015332 000400          BR       SC          ;CHECK FOR OTHER DRIVES
3725
3726          ;SPECIAL CONDITION ROUTINE
3727

```

F06

CZRJB80, RPO4/5/6 FMTR MACY11 30(1046) 07-NOV-77 10:07 PAGE 71
 CZRJB8.P11 04-NOV-77 12:54 SINGLE/DUAL PORT RH11/RP04/5/6 DRIVER (REV 1.0)

SEQ 0070

3728	015334	116403	000016	SC:	MOVB	RPAS(R4),R3	;READ "RPAS"
3729	015340	001014			BNE	2\$;BRANCH IF ANY 'ATA' BITS SET
3730	015342	004037	017214		JSR	RO,RO.RP	;READ CONTROL AND STATUS REGISTER
3731	015346	000000			RPCS1		
3732	015350	014536			CIB		
3733	015352	106126			ROLB	(SP)+	;IS "IE"=1?
3734	015354	100405			BMI	1\$;YES, NO DRIVES TO CHECK
3735	015356	004037	020506		JSR	RO,ES.SAV	;SAVE THE ADDRESS IN '\$ESCAPE'
3736	015362	104001			ERROR	1	;REPORT AN ILLEGAL INTERRUPT
3737	015364	004737	017704		JSR	PC,SET.IE	;SET INTERRUPT ENABLE
3738	015370	000207		1\$:	RTS	PC	;RETURN
3739	015372	005046		2\$:	CLR	-(SP)	;PROCESS ALL DRIVES THAT HAVE
3740	015374	110316			MOVB	R3,(SP)	;AN "ATA"=1
3741	015376	012703	000001		MOV	#1,R3	
3742	015402	005001			CLR	R1	
3743	015404	030316		SC3:	BIT	R3,(SP)	;ATA=1?
3744	015406	001005			BNE	SC5	;YES--BRANCH
3745	015410	005201		SC4:	INC	R1	;MOVE TO THE NEXT DRIVE
3746	015412	106303			ASLB	R3	
3747	015414	001373			BNE	SC3	;BRANCH IF MORE TO CHECK?
3748	015416	005726			TST	(SP)+	;CLEAN OFF THE STACK
3749	015420	000207			RTS	PC	;RETURN TO USER
3750	015422	105761	012140	SC5:	TSTB	DPINT(R1)	;INITIALIZING THE DRIVE ?
3751	015426	001402			BEQ	1\$;BR IF NOT
3752	015430	000137	016326		JMP	SC13	;PROCESS THE DRIVE
3753	015434	105761	012150	1\$:	TSTB	DPROS(R1)	;PORT REQUEST OUTSTANDING ?
3754	015440	001402			BEQ	2\$;BR IF NOT
3755	015442	000137	016326		JMP	SC13	;START THE OUTSTANDING COMMAND
3756	015446	105761	012120	2\$:	TSTB	DRVSTA(R1)	;CHECK THE DRIVE STATUS
3757	015452	003025			BGT	5\$;BRANCH IF ONLINE
3758	015454	105761	012166		TSTB	ULDFLG(R1)	;UNLOAD IN PROGRESS?
3759	015460	003422			BLE	5\$;BRANCH IF NOT
3760	015462	004737	020422		JSR	PC,GETREQ	;GET DPB POINTER
3761	015466	004737	017566		JSR	PC,SVR411	;SAVE THE RH11/RP04/5/6 REGISTERS
3762	015472	004737	016256		JSR	PC,SC12	;SAVE RPDS1, RPER1, RPER2, AND RPER3
3763							;ALSO DO A DRVINT
3764	015476	105761	012120		TSTB	DRVSTA(R1)	;DID DRIVE COME ONLINE?
3765	015502	003416			BLE	6\$;NO---BRANCH
3766	015504	032737	040000	012100	BIT	#BIT14,RPERRS	;WAS THERE AN ERROR?
3767	015512	001002			BNE	3\$;BR IF ERROR
3768	015514	000137	016166		JMP	SC11	;NO ERROR
3769	015520	013705	012102	3\$:	MOV	RPERRS+2,R5	;YES -- PICKUP RPER1 AND
3770	015524	000502			BR	SCF9	;GO PROCESS THE ERROR
3771	015526	105761	012110	5\$:	TSTB	DRVACT(R1)	;DRIVE ACTIVE WITH COMMAND OR ERROR RECOVERY ?
3772	015532	001033			BNE	SC6	;BR IF EITHER
3773	015534	004737	016256		JSR	PC,SC12	;SAVE RPDS1, RPER1, RPER2, AND RPER3
3774							;ALSO DO A DRVINT
3775	015540	105761	012140	6\$:	TSTB	DPINT(R1)	;TRYING TO INIT THE DRIVE ?
3776	015544	001321			BNE	SC4	;BR IF YES, CHECK ON MORE DRIVES
3777	015546	105761	012120		TSTB	DRVSTA(R1)	;CHECK ON DRIVE'S STATUS
3778	015552	100412			BMI	7\$;BR IF UNSAFE
3779	015554	032737	020000	012106	BIT	#BIT13,RPERRS+6	;ADDRESS PLUG CHANGED ?
3780	015562	001013			BNE	8\$;BR IF YES
3781	015564	012746	000113		MOV	#113, -(SP)	;RELEASE COMMAND
3782	015570	004037	017374		JSR	RO,WAT.RP	;WRITE THE COMMAND INTO RPCS1
3783	015574	000000			RPCS1		;REGISTER INDE


```

3840 016046 000137 015410          JMP      SC4
3841 015052 052762 100220 000016 3$:  BIS     #BIT15!BIT07!BIT04,16(R2) ;INFORM USER OF ERROR
3842 015060 105061 012110          SC7:  CLRB   DRVACT(R1) ;DRIVE IS IDLE
3843 016064 004737 020326          JSR     PC,EMPTYQ ;DUMP THE QUEUE
3844 016070 105761 012166          TSTB   ULDFLG(R1) ;UNLOAD IN PROGRESS OR QUEUE?
3845 016074 003002          BGT     1$ ;BR IF NOT
3846 016076 105061 012166          CLRB   ULDFLG(R1) ;CLEAR UNLOAD FLAG
3847 016102 116164 012234 000016 1$:  MOVB   ATABIT(R1),RPAS(R4) ;CLEAR ATTENTION BIT
3848 016110 105761 012120          TSTB   DRVSTA(R1) ;IS THE DRIVE UNSAFE ?
3849 016114 100406          BMI     2$ ;BR IF IT IS
3850 016116 012746 000113          MOV     #113,-(SP) ;RELEASE COMMAND
3851 016122 004037 017374          JSR     RD,WRT.RP ;WRITE THE COMMAND INTO RPCS1
3852 016126 000000          RPCS1
3853 016130 016136          SC8:  SCB
3854 016132 000137 015410          JMP     SC4 ;CHECK FOR MORE DRIVES
3855 016136 105761 012110          SC8:  TSTB   DRVACT(R1) ;IS DRIVE IDLE?
3856 016142 001405          BEQ     1$ ;YES--BRANCH
3857 016144 004737 020422          JSR     PC,GETREQ ;GET DPB POINTER
3858 016150 004737 014436          JSR     PC,C17 ;PROCESS THE PARITY ERROR
3859 016154 000402          BR      2$ ;CONTINUE
3860 016156 004737 014464          1$:  JSR     PC,C17B ;PROCESS THE UNCORRECTABLE PARITY ERROR
3861 016162 000137 015410          2$:  JMP     SC4 ;CHECK MORE DRIVES
3862 016166 105761 012166          SC11: TSTB   ULDFLG(R1) ;"UNLOAD IN PROGRESS"?
3863 016172 003402          BLE     1$ ;BRANCH IF NO
3864 016174 105061 012166          CLRB   ULDFLG(R1) ;CLEAR UNLOAD FLAG
3865 016200 105061 012110          1$:  CLRB   DRVACT(R1) ;SET DRIVE IDLE
3866 016204 136137 012234 012162          BITB   ATABIT(R1),SRCHWT ;DOING A SEARCH OPERATION FOR
3867                                     ;AN I/O COMMAND?
3868 016212 001012          BNE     2$ ;BRANCH IF YES
3869 016214 004737 020444          JSR     PC,POPQUE ;REMOVE REQUEST FROM QUEUE
3870 016220 052762 000200 000016          BIS     #BIT07,16(R2) ;SET "DONE" BIT
3871 016226 005737 012206          TST     SAVEFG ;SAVE THE REGISTERS?
3872 016232 100002          BPL     2$ ;BRANCH IF NO
3873 016234 004737 017566          JSR     PC,SVRH11 ;YES--SAVE ALL OF THE RH11/RPO4/5.6 REG'S
3874 016240 116164 012234 000016 2$:  MOVB   ATABIT(R1),RPAS(R4) ;CLEAR ATTENTION BIT
3875 016246 004737 013326          JSR     PC,OPT ;START A REQUEST
3876 016252 000137 015410          JMP     SC4 ;CHECK FOR MORE DRIVES
3877 016256 010164 100010          SC12: MOV     R1,RPCS2(R4) ;SELECT DRIVE
3878 016262 016437 000012 012100          MOV     RPOS1(R4),RPERRS ;SAVE THE FOUR REGISTERS THAT
3879 016270 016437 000014 012102          MOV     RPER1(R4),RPERRS+2 ;WILL TELL US SOMETHING
3880 016276 016437 000040 012104          MOV     RPER2(R4),RPERRS+4
3881 016304 016437 000042 012106          MOV     RPER3(R4),RPERRS+6
3882 016312 004037 012476          JSR     RD,DRVINT ;INIT. THE STATE OF THE DRIVE
3883 016316 000401          BR      1$ ;TAKE ERROR EXIT
3884 016320 000207          RTS     PC ;RETURN
3885 016322 005726          1$:  TST     (SP)+ ;POP PC OFF OF THE STACK
3886 016324 000704          BR      SC8 ;PROCESS THE PARITY ERROR
3887 016326 006301          SC13: ASL     R1 ;SETUP TO ADDRESS WORDS
3888 016330 012761 177777 012212          MOV     #-1,TIMER(R1) ;STOP THE TIMER
3889 016336 006201          ASR     R1
3890 016340 010164 000010          MOV     R1,RPCS2(R4) ;SELECT THE DRIVE
3891 016344 116164 012234 000016          MOVB   ATABIT(R1),RPAS(R4) ;CLEAR THE ATTENTION BIT
3892 016352 032714 004000          BIT     #BIT11,(R4) ;DRIVE AVAILABLE ?
3893 016356 001006          BNE     1$ ;BR IF AVAILABLE
3894 016360 006301          ASL     R1
3895

```

```

3896 016362 012761 023420 012212      MOV      #10000.,TIMER(R1) ;START 10 SEC TIMER AGAIN
3897
3898 016370 006201      ASR      R1
3899 016372 000433      BR      3$
3900 016374 105761 012140      1$: TSTB   DPINT(R1) ;INITIALIZING THE DRIVE ?
3901 016400 001424      BEQ     2$ ;BR IF NOT
3902 016402 105061 012140      CLRB   DPINT(R1) ;CLEAR THE INIT INDICATOR
3903 016406 004037 012476      JSR    RD,DRVINT ;GO INIT THE DRIVE
3904 016412 000240      NOP
3905 016414 105761 012120      TSTB   DRVSTA(R1) ;DRIVE ONLINE ?
3906 016420 003014      BGT    2$ ;BR IF YES -- START ORDER
3907 016422 005702      TST    R2 ;QUEUE ENTRY FOR THE DRIVE
3908 016424 001416      BEQ    3$ ;BR IF NOT
3909 016426 004737 020422      JSR    PC,GETREQ ;GET DPB ADDRESS
3910 016432 052762 140000 000016      BIS    #BIT15:BIT14,16(R2) ;INFORM USER THAT DRIVE OFFLINE
3911 016440 004737 017566      JSR    PC,SVRH11 ;SAVE THE REGISTERS
3912 016444 004737 020326      JSR    PC,EMPTYQ ;EMPTY THE REQUEST QUEUE
3913 016450 000404      BR     3$
3914 016452 105061 012150      2$: CLRB   DPRQS(R1) ;CLEAR THE PORT REQUEST INDICATOR
3915 016456 004737 013326      JSR    PC,OPT ;START THE PENDING REQUEST
3916 016462 000137 015410      3$: JMP    SC4 ;PROCESS OTHER DRIVES
3917
3918 ;RPO4/5/6 TIMER ROUTINE
3919 ;CALL
3920 ;
3921 ;      MOV      #TIME -(SP) ;ELAPSED TIME IN MILLISECONDS ON THE STACK
3922 ;      JSR      PC,RPTMR ;CALL RPO4/5/6 TIME ROUTINE
3923
3923 016466 005737 012164      RPTMR: TST    ACTDRV ;CHECK "ACTDRV & ACTSTR"
3924 016472 001030      BNE    4$ ;IF NON ZERO EXIT
3925 016474 112737 000001 012165      MOVB   #1,ACTSTR ;SET "ACTSTR"
3926 016502 104412      SAVREG ;SAVE R0 - R5
3927 016504 005001      CLR    R1 ;START WITH DRIVE 0
3928 016506 005003      CLR    R3
3929 016510 005763 012212      1$: TST    TIMER(R3) ;IS THE TIMER RUNNING?
3930 016514 002407      BLT    2$ ;BRANCH IF NO
3931 016516 166663 000002 012212      SUB    2(SP),TIMER(R3) ;COUNT THE INTERVAL
3932 016524 003003      BR    2$ ;BR IF NO SOFTWARE TIMEOUT
3933 016526 004737 016560      JSR    PC,STO ;CALL SOFTWARE TIMEOUT ROUTINE
3934 016532 000405      BR    3$ ;GO TO THE EXIT
3935 016534 005201      2$: INC    R1 ;MOVE TO NEXT DRIVE
3936 016536 005723      TST    (R3)+
3937 016540 022701 000010      CMP    #8.,R1 ;OUT OF DRIVES?
3938 016544 003361      BGT    1$ ;BRANCH IF NO
3939 016546 104413      3$: RESREG ;RESTORE R0 - R5
3940 016550 105037 012165      CLRB   ACTSTR ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
3941 016554 012E16      4$: MOV    (SP)+,(SP) ;ADJUST THE STACK
3942 016556 000207      RTS    PC ;RETURN
3943
3944 ;SOFTWARE TIMEOUT ROUTINE
3945 ;
3946 ;NOTE: THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
3947 ;OR GREATER
3948 ;
3949 ;CALL: STO
3950 ;      MOV      #DRYNUM,R1 ;DRIVE NUMBER
3951 ;      JSR      PC,STO ;CALL
    
```

```

3952 ; RETURN
3953
3954 016560 010146 STO: MOV R1,-(SP) ;SAVE R1
3955 016562 010346 MOV R3,-(SP) ;SAVE R3
3956 016564 013704 012246 MOV RPADR,R4 ;GET ADDRESS OF "RPCS1"
3957 016570 010164 000010 MOV R1,RPCS2(R4) ;SELECT THE DRIVE
3958 016574 004037 017214 JSR RO,RO.RP ;READ "DRIVE STATUS REG"
3959 016600 000012 RPDS1
3960 016602 017102 STOS
3961 016604 105726 TSTB (SP)+ ;IS "DRY"=1?
3962 016606 100477 BMI ST02 ;BR IF YES
3963 016610 105761 012140 ST01: TSTB DPINT(R1) ;TRYING TO INITIALIZE THE DRIVE ?
3964 016614 001074 BNE ST02 ;BR IF YES
3965 016616 105761 012150 TSTB DPRQS(R1) ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
3966 016622 001071 BNE ST02 ;BR IF YES
3967 016624 013702 012160 MOV TRANSW,R2 ;PICKUP TRANSFER WAIT QUEUE
3968 016630 020137 012232 CMP R1,DTUW ;TRANSFER UNDERWAY ON THIS DRIVE?
3969 016634 001402 BEQ 1$ ;BRANCH IF YES
3970 016636 004737 020422 JSR PC.GETREQ ;GET DPB ADDRESS
3971 016642 052762 101000 000016 1$: BIS #BIT15:BIT09,16(R2) ;SET THE ERROR FLAGS
3972 016650 004737 017566 JSR PC.SVRH11 ;SAVE RH11/RPO4/5/6 REGISTERS
3973 016654 012764 000040 000010 MOV #BIT05,RPCS2(R4) ;"INIT" THE MASS BUS
3974 016662 105061 012110 CLRB DRVACT(R1) ;DRIVE IS IDLE
3975 016666 105061 012166 CLRB ULDFLG(R1) ;CLEAR THE UNLOAD FLAG
3976 016672 005001 CLR R1 ;START WITH DRIVE 0
3977 016674 005003 CLR R3
3978 016676 004037 012476 2$: JSR RO.DRVINT ;INIT. THIS DRIVE
3979 016702 000477 BR ST05 ;PARITY ERROR RETURN
3980 016704 105761 012110 TSTB DRVACT(R1) ;DRIVE IDLE BEFORE THE INIT.?
3981 016710 001414 BEQ 4$ ;YES--BRANCH
3982 016712 013702 012160 MOV TRANSW,R2 ;GET TRANSFER WAIT QUEUE
3983 016716 023701 012232 CMP DTUW,R1 ;WAS THERE I/O ON THIS DRIVE?
3984 016722 001402 BEQ 3$ ;YES--BRANCH
3985 016724 004737 020422 JSR PC.GETREQ ;GET THE DPB POINTER FROM QUEUE
3986 016730 052762 100400 000016 3$: BIS #BIT15:BIT08,16(R2) ;INFORM USER OF INIT.
3987 016736 105061 012110 CLRB DRVACT(R1) ;SET DRIVE ACTIVE TO IDLE
3988 016742 105061 012166 4$: CLRB ULDFLG(R1) ;NO UNLOAD
3989 016746 012763 177777 012212 MOV #-1,TIMER(R3) ;STOP THE TIMER
3990 016754 005723 TST (R3)+ ;UPDATE THE INDCX
3991 016756 005201 INC R1 ;INCREMENT THE DRIVE NUMBER
3992 016760 022701 000010 CMP #8.,R1 ;LAST DRIVE BEEN CHECKED?
3993 016764 003344 BGT 2$ ;NO--LOOP
3994 016766 012737 177777 012232 MOV #-1,DTUW ;NO DATA TRANSFERS UNDERWAY
3995 016774 005037 012160 CLR TRANSW ;CLEAR TRANSFER WAIT QUEUE
3996 017000 004737 020250 JSR PC.CLRQUE ;CLEAR ALL REQUEST QUEUES
3997 017004 000500 BR ST09 ;EXIT
3998 017006 116405 000016 ST02: MOVB RPAS(R4),R5 ;READ ATTENTION REG
3999 017012 136105 012234 BITB ATABIT(R1),R5 ;IS ATTENTION FOR THIS DRIVE UP ?
4000 017016 001017 BNE ST03 ;YES--BRANCH
4001 017020 105761 012140 TSTB DPINT(R1) ;TRYING TO INITIALIZE THE DRIVE ?
4002 017024 001031 BNE ST06 ;BR IF YES - DRIVE NOT ONLINE
4003 017026 105761 012150 TSTB DPRQS(R1) ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
4004 017032 001045 BNE ST07 ;BR IF YES - NO RESPONSE TO REQUEST
4005 017034 020137 012232 CMP R1,DTUW ;DATA TRANSFER UNDERWAY FOR THIS DRIVE
4006 017040 001263 BNE ST01 ;BR IF NO
4007 017042 004037 017214 JSR RO,RO.RP ;YES--CHECK "RDY"
    
```

K06

```

4008 017046 000000          RPCS1
4009 017050 017102          ST05
4010 017052 105726          TSTB      (SP)+
4011 017054 100255          BPL      ST01          ;BR IF "RDY"=0
4012 017056 105761 012140  ST03:  TSTB      DPINT(R1)      ;INITIALIZING THE DRIVE ?
4013 017062 001003          BNE      IS          ;BR IF INIT PENDING
4014 017064 105761 012150          TSTB      DPRQS(R1)      ;PORT REQUEST PENDING ?
4015 017070 001446          BEQ      ST09          ;BR IF NOT
4016 017072 012763 177777 012212  IS:   MOV      #-1,TIMER(R3) ;STOP THE TIMER
4017 017100 000442          BP       ST09          ;EXIT
4018 017102 004737 014536          ST05:  JSR      PC,C18      ;GO HANDLE THE PARITY ERROR
4019 017106 000437          BR       ST09
4020 017110 105061 012140          ST06:  CLRB      DPINT(R1)      ;CLEAR THE INITIALIZE INDICATOR
4021 017114 105061 012120          CLRB      DRVSTA(R1)      ;SET UNIT OFFLINE
4022 017120 012763 177777 012212          MOV      #-1,TIMER(R3) ;STOP THE TIMER
4023 017126 004737 020422          JSR      PC,GETREQ      ;GET THE DPB ADDRESS
4024 017132 005702          TST      R2          ;REQUEST IN QUEUE ?
4025 017134 001424          BEQ      ST09          ;BR IF NOT
4026 017136 052762 140000 000016          BIS      #BIT15:BIT14,16(R2) ;INFORM THE USER DRIVE NOT AVAILABLE
4027 017144 000414          BR       ST08
4028 017146 012763 177777 012212  ST07:  MOV      #-1,TIMER(R3) ;STOP THE TIMER
4029 017154 105061 012150          CLRB      DPRQS(R1)      ;CLEAR PORT REQUEST INDICATOR
4030 017160 004737 020422          JSR      PC,GETREQ      ;GET DPB ADDRESS
4031 017164 005702          TST      R2          ;QUEUE ENTRY FOR DRIVE ?
4032 017166 001407          BEQ      ST09          ;BR IF NONE
4033 017170 012762 100004 000016          MOV      #BIT15:BIT2,16(R2) ;INFORM USER OF PORT REQUEST ERROR
4034 017176 004737 020326          ST08:  JSR      PC,EMPTYQ      ;CLEAR THE QUEUE FOR THE DRIVE
4035 017202 004737 017566          JSR      PC,SVRH11      ;SAVE THE REGISTERS
4036 017206 012603          ST09:  MOV      (SP)+,R3      ;RESTORE R3
4037 017210 012601          MOV      (SP)+,R1      ;RESTORE R1
4038 017212 000207          RTS      PC          ;RETURN
4039
4040          ;ROUTINE TO READ A RH11/RP04/5/6 REGISTER
4041          ;CALL
4042          ;
4043          JSR      RD,RD.RP      ;GO READ A REGISTER
4044          INDEX      ;REG. INDEX FROM BASE
4045          ERRADR      ;ERROR ADDRESS--PROCESS ERROR STARTING
4046          ;AT THIS ADDRESS
4047          RETURN      ;CONTENTS OF REG. IS ON THE STACK
4048
4049 017214 013737 012244 017362  RD.RP:  MOV      MCPEMX,RD.RP2      ;MAX. RETRYS ALLOWED
4050 017222 011646          MOV      (SP)-,(SP)      ;SAVE RD FOR RETURN
4051 017224 013737 012246 017240          MOV      RPADR,RD.ADR      ;FORM THE DESIRED ADDRESS
4052 017232 062037 017240          ADD      (RD)+,RD.ADR      ;USING THE BASE AND THE INDEX
4053 017236 013727          RD.RP1: MOV      @((PC)+,(PC)+      ;READ THE DESIRED REGISTER OF THE RP04
4054 017240 000000          RD.ADR: .WORD      0      ;ADDRESS IS FORMED HERE
4055 017242 000000          RD.WRD: .WORD      0      ;REG. CONTENTS PUT HERE
4056 017244 013766 017242 000002          MOV      RD.WRD,2(SP)      ;RETURN IT TO THE USER
4057 017252 013746 012246          MOV      RPADR,-(SP)      ;PUT THE ADDRESS ON THE STACK
4058 017256 062716 000010          ADD      #RPCS2,(SP)      ;FORM THE ADDRESS OF RPCS2
4059 017262 032736 010000          BIT      #BIT12,@(SP)+      ;CHECK THE 'NED' BIT
4060 017266 001037          BNE      RD.RP3      ;BR IF DRIVE NON-EXISTENT
4061 017270 017746 172752          MOV      @RPADR,-(SP)      ;READ RPCS1
4062 017274 032716 020000          BIT      #BIT13,(SP)      ;DID MCPE SET?
4063 017300 001002          BNE      IS          ;BRANCH IF YES
    
```

```

4064 017302 022620          CMP      (SP)+,(RO)+      ;ADJUST FOR RETURN
4065 017304 000432          BR       RD.RP4          ;EXIT
4066 017306                1$:
4067 017306 004037 020506      JSR      RD,ES.SAV      ;SAVE THE ADDRESS IN 'SESCAPE'
4068 017312 104003          ERROR    3              ;REPORT "MCPE" ERROR
4069 017314 005737 012232      TST     DTUW           ;DATA TRANSFER UNDERWAY?
4070 017320 100405          BMI     2$              ;NO--BRANCH
4071 017322 032716 040000      BIT     #BIT14,(SP)    ;NO--"TRE"=1?
4072 017326 001402          BEQ     2$              ;NO--BRANCH
4073 017330 005726          TST     (SP)+          ;YES--CLEAN OFF THE STACK AND
4074 017332 000415          BR     RD.RP3          ;TAKE THE FATAL ERROR EXIT
4075 017334 052716 040000      2$:    BIS     #BIT14,(SP)  ;CLEAR "MCPE" BY SENDING A "1" TO "TRE"
4076 017340 000316          SWAB   (SP)            ;POSITION BEFORE WRITING
4077 017342 013737 012246 017356  MOV     RPADR,3$       ;FORM ADDRESS OF HIGH BYTE
4078 017350 005237 017356          INC     3$              ;WRITE THE HIGH BYTE OF RPCS1
4079 017354 112637          MOVB   (SP)+,2(PC)+   ;ADDRESS STORAGE
4080 017356 000000          3$:    .WORD 0           ;EXCEEDED MAX. RETRYS
4081 017360 005327          DEC     (PC)+
4082 017362 000003          RD.RP2: .WORD 3
4083 017364 002324          BGE    RD.RP1         ;BRANCH IF NO
4084 017366 011000          RD.RP3: MOV     (RO),RO ;FATAL ERROR EXIT
4085 017370 012616          MOV     (SP)+,(SP)
4086 017372 000200          RD.RP4: RTS    RO
4087
4088          ;ROUTINE TO WRITE A REGISTER
4089          ;CALL
4090          ;
4091          ;      MOV     DATA -(SP)      ;DATA TO BE LOADED ON THE STACK
4092          ;      JSR     RD,WRT.RP      ;CALL THE ROUTINE TO LOAD(WRITE) THE REG.
4093          ;      INDEX  ERRADR        ;INDEX OF THE REGISTER TO BE LOADED
4094          ;      ERRADR  RETURN      ;ADDRESS TO RETURN TO ON AN ERROR
4095          ;
4096          ;
4097 017374 013737 012244 017550  WRT.RP: MOV     MCPEMX,WRT.R2 ;MAX RETRYS ALLOWED
4098 017402 016637 000002 017462  MOV     2(SP),WRT.WD ;SAVE THE WORD TO WRITE
4099 017410 012616          MOV     (SP)+,(SP)    ;ADJUST THE STACK
4100 017412 012037 017464          MOV     (RO)+,WRT.AD ;GET INDEX OF REGISTER TO BE WRITTEN
4101 017416 001015          BNE    1$              ;BRANCH IF NOT RPCS1
4102 017420 122737 000150 017462  CMPB   #150,WRT.WD    ;IS THE COMMAND FOR DATA TRANSFERS?
4103 017426 002411          BLT    1$              ;YES--DON'T GET THE OLD A16 & A17. & PSEL
4104 017430 004037 017214          JSR     RD,RD.RP      ;NO---COMBINE A16&A17, & PSEL WITH
4105 017434 000000          RPCS1 ;THE COMMAND BEFORE SENDING IT TO
4106 017436 017556          WRT.R3 ;THE RH11/RPO4
4107 017440 000316          SWAB   (SP)
4108 017442 042716 177770          BIC    #1C7,(SP)
4109 017446 112637 017463          MOVB   (SP)+,WRT.WD+1
4110 017452 063737 012246 017464  1$:    ADD     RPADR,WRT.AD ;FORM THE ADDRESS OF THE DISK REG.
4111 017460 012737          WRT.R1: MOV     (PC)+,2(PC)+ ;LOAD THE DESIRED REG.
4112 017462 000000          WRT.WD: .WORD 0      ;WORD TO WRITE GOES HERE
4113 017464 000000          WRT.AD: .WORD 0      ;ADDRESS IS FORMED HERE
4114 017466 013746 012246          MOV     RPADR, -(SP)  ;PUT THE ADDRESS ON THE STACK
4115 017472 062716 000010          ADD     #RPCS2,(SP)  ;FORM THE ADDRESS OF RPCS2
4116 017476 032736 010000          BIT     #BIT12,2(SP)+ ;CHECK THE 'NED' BIT
4117 017502 001025          BNE    WRT.R3         ;BR IF DRIVE NON-EXISTENT
4118 017504 004037 017214          JSR     RD,RD.RP      ;CHECK FOR PARITY ERROR ON WRITE
4119 017510 000014          RPER1
    
```

```

4120 017512 017556          WRT.R3
4121 017514 032726 000010  BIT      #BIT03,(SP)+
4122 017520 001420          BEQ      WRT.R4          ; BRANCH IF "PAR=0"
4123 017522 016037 177776 017534  MOV      -2(R0),1$      ; PICKUP THE INDEX
4124 017530 004037 017214  JSR      RO,RO.RP      ; READ THE REG.
4125 017534 000000 1$:      .WORD      0          ; REG. INDEX
4126 017536 017556          WRT.R3          ; RETURN TO THIS ADDRESS ON ERROR
4127 017540 004037 020506  JSR      RO,ES.SAV     ; SAVE THE ADDRESS IN 'SESCAPE'
4128 017544 104004          ERROR      4          ; REPORT THE PARITY ON WRITE ERROR
4129 017546 005327          DEC      (PC)+        ; DECREMENT THE ERROR COUNT
4130 017550 000003  WRT.R2:  .WORD      3          ; REPLY COUNTER
4131 017552 002342          BGE      WRT.R1      ; TRY AGAIN IF NOT FINISHED
4132 017554 005726          TST      (SP)+        ; CLEAN OFF THE STACK
4133 017556 011000  WRT.R3:  MOV      (RO),RO    ; TAKE THE "PARITY ON WRITE" ERROR EXIT
4134 017560 000401          BR       WRT.R5      ; EXIT
4135 017562 005720  WRT.R4:  TST      (RO)+    ; ADJUST FOR ERROR FREE EXIT
4136 017564 000200  WRT.R5:  RTS       RO
4137
4138          ; ROUTINE TO SAVE THE RH11/RP04/5/6 REGISTERS AS PER DPB+14
4139          ; CALL
4140          ;
4141          MOV      #DPBNUM,R2          ; DPB POINTER TO R2
4142          JSR      PC,SVRH11          ; SAVE THE DRIVES REG'S
4143
4144 017566 104412  SVRH11:  SAV/REG          ; SAVE R0 - R5
4145 017570 005702          TST      R2          ; QUEUE ENTRY FOR THE DRIVE ?
4146 017572 001430          BEQ      4$          ; BR IF NONE
4147 017574 013704 012246  MOV      RPADR,R4
4148 017600 111264 000010  MOV      (R2),RPCS2(R4) ; SELECT DRIVE
4149 017604 016203 000014  MOV      14(R2),R3    ; GET THE ERROR TABLE POINTER
4150 017610 001433          BEQ      6$          ; EXIT IF NO ADDRESS
4151 017612 005037 017646  CLR      3$          ; COUNTER & POINTER
4152 017616 023727 017646 000022 1$:      CMP      3$,#RPDB     ; REACHED THE BUFFER REGISTER ?
4153 017624 001006          BNE      2$          ; BR IF NOT
4154 017626 032764 000200 000010  BIT      #BIT07,RPCS2(R4) ; 'OR' SET ?
4155 017634 001002          BNE      2$          ; BR IF SET
4156 017636 005023          CLR      (R3)+        ; STORE RPDB AS ZEROES
4157 017640 000405          BR       4$          ; CONTINUE
4158 017642 004037 017214 2$:      JSR      RO,RO.RP     ; READ THE SELECTED REGISTER
4159 017646 000000 3$:      .WORD      0          ; REGISTER INDEX
4160 017650 017674          5$:      MOV      (SP,+(R3)+    ; ERROR RETURN ADDRESS
4161 017652 012623          4$:      CMP      3$,#RPEC2    ; STORE THE REGISTER CONTENTS
4162 017654 023727 017646 000046  BEQ      6$          ; REACHED THE END ?
4163 017662 001406          BEQ      6$          ; BR IF YES
4164 017664 062737 000002 017646  ADD      #2,3$        ; INCREMENT THE REGISTER INDEX
4165 017672 000751          BR       1$          ; CONTINUE READING THE REGISTERS
4166 017674 004737 014436 5$:      JSR      PC,C17      ; PROCESS THE UNCORRECTABLE PARITY ERROR
4167 017700 104413 6$:      RESREG          ; RESTORE R0 - R5
4168 017702 000207          RTS       PC          ; RETURN
4169
4170          ; ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A "TRE"
4171          ; CALL
4172          ;
4173          MOV      #DRYNUM,R1          ; DRIVE NUMBER TO R1
4174          JSR      PC,SET.IE          ; SET "IE"
4175          RETURN
    
```

```

4176 017704 010446          SET.IE: MOV      R4 -(SP)          ;SAVE R4
4177 017706 013704 012246  MOV      RPADR,R4          ;PICKUP ADDRESS OF RPCS1
4178 017712 010164 000010  MOV      R1,RPCS2(R4)      ;SELECT DRIVE
4179 017716 011446          MOV      (R4)-(SP)        ;READ RPCS1
4180 017720 052716 040000  BIS      #BIT14,(SP)      ;SET THE "TRE" BIT OF THE WORD READ
4181 017724 000316          SWAB      (SP)            ;ADJUST FOR DATO
4182 017726 112714 000100  MOVB     #BIT06,(R4)      ;SET "IE"
4183 017732 032764 010000 000010 BIT      #BIT12,RPCS2(R4) ;IS "NED"=1?
4184 017740 001002          BNE      1$              ;YES--CLEAR "TRE"
4185 017742 005726          TST      (SP)+           ;CLEAN OFF THE STACK
4186 017744 000402          BR       2$              ;
4187 017746 112664 000001  1$:     MOVB    (SP)+,1(R4)  ;CLEAR "TRE"
4188 017752 012604          2$:     MOV      (SP)+,R4    ;RESTORE R4
4189 017754 000207          RTS      PC              ;RETURN TO CALLER

;QUEUE COUNT
4191          QCNT:
4192 017756          .BYTE 0 ;DRIVE 0
4193 017757          .BYTE 0 ;DRIVE 1
4194 017760          .BYTE 0 ;DRIVE 2
4195 017761          .BYTE 0 ;DRIVE 3
4196 017762          .BYTE 0 ;DRIVE 4
4197 017763          .BYTE 0 ;DRIVE 5
4198 017764          .BYTE 0 ;DRIVE 6
4199 017765          .BYTE 0 ;DRIVE 7

;QUEUE INPUT POINTERS
4201          QINPT:
4202
4203 017766 020050          .WORD QDRV0 ;DRIVE 0
4204 017770 020070          .WORD QDRV1 ;DRIVE 1
4205 017772 020110          .WORD QDRV2 ;DRIVE 2
4206 017774 020130          .WORD QDRV3 ;DRIVE 3
4207 017776 020150          .WORD QDRV4 ;DRIVE 4
4208 020000 020170          .WORD QDRV5 ;DRIVE 5
4209 020002 020210          .WORD QDRV6 ;DRIVE 6
4210 020004 020230          .WORD QDRV7 ;DRIVE 7

;QUEUE OUTPUT POINTERS
4211          QOUTPT:
4212
4213
4214 020006 020050          .WORD QDRV0 ;DRIVE 0
4215 020010 020070          .WORD QDRV1 ;DRIVE 1
4216 020012 020110          .WORD QDRV2 ;DRIVE 2
4217 020014 020130          .WORD QDRV3 ;DRIVE 3
4218 020016 020150          .WORD QDRV4 ;DRIVE 4
4219 020020 020170          .WORD QDRV5 ;DRIVE 5
4220 020022 020210          .WORD QDRV6 ;DRIVE 6
4221 020024 020230          .WORD QDRV7 ;DRIVE 7

;DRIVE 0 START ADDRESS
4222
4223 020026 020050          QSTART: .WORD QDRV0 ;DRIVE 0 START ADDRESS
4224 020030 020070          QSTOP:  .WORD QDRV1 ;DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
4225 020032 020110          .WORD QDRV2 ;STOP DRIVE 1--START DRIVE 2
4226 020034 020130          .WORD QDRV3 ;STOP DRIVE 2--START DRIVE 3
4227 020036 020150          .WORD QDRV4 ;STOP DRIVE 3--START DRIVE 4
4228 020040 020170          .WORD QDRV5 ;STOP DRIVE 4--START DRIVE 5
4229 020042 020210          .WORD QDRV6 ;STOP DRIVE 5--START DRIVE 6
4230 020044 020230          .WORD QDRV7 ;STOP DRIVE 6--START DRIVE 7
4231 020046 020250          .WORD QTERM ;STOP DRIVE 7
    
```



```

4288 ; RETURN2 ; RETURN HERE IF REQUEST IS IN QUEUE
4289
4290 020346 122761 000010 017756 DRVQUE: CMPB #10, QCNT(R1) ; IS QUEUE FULL?
4291 020354 001421 BEQ 2$ ; BR IF YES-TAKE RETURN1
4292 020356 105261 017756 INCB QCNT(R1) ; INCREMENT QUEUE COUNT
4293 020362 006301 ASL R1
4294 020364 010271 017766 MOV R2, QINPT(R1) ; PUT THIS REQUEST IN QUEUE
4295 020370 062761 000002 017766 ADD #2, QINPT(R1) ; UPDATE THE QUEUE POINTER
4296 020376 026161 017766 020030 CMP QINPT(R1), QSTOP(R1) ; TIME TO RESET THE POINTER
4297 020404 001003 BNE 1$ ; BRANCH IF NO
4298 020406 016161 020026 017766 MOV QSTART(R1), QINPT(R1) ; YES--RESET POINTER
4299 020414 006201 1$: ASR R1
4300 020416 005720 TST (R0)+ ; TAKE RETURN 2
4301 020420 000200 2$: RTS R0 ; RETURN TO USER
4302
4303 ; ROUTINE TO GET THE "DPB" ADDRESS OF NEXT REQUEST IN QUEUE
4304
4305 ; CALL
4306 ; MOV #DRVNUM, R1 ; DRIVE NUMBER TO R1
4307 ; JSR PC, GETREQ ; GO GET THE REQUEST
4308 ; RETURN ; R2="DPB" ADDRESS OF THE REQUEST
4309 ; ; R2=0 IF NO REQUEST IN QUEUE
4310
4311 020422 005002 GETREQ: CLR R2
4312 020424 105761 017756 TSTB QCNT(R1) ; IS THERE ANY REQUEST IN QUEUE?
4313 020430 001404 BEQ 2$ ; NO---BRANCH
4314 020432 006301 1$: ASL R1
4315 020434 017102 020006 MOV QOUTPT(R1), R2 ; PICKUP "DPB" POINTER FOR THIS DRIVE
4316 020440 006201 ASR R1
4317 020442 000207 2$: RTS PC ; RETURN TO USER
4318
4319 ; ROUTINE TO "POP" THE REQUEST FROM QUEUE
4320
4321 ; CALL
4322 ; MOV #DRVNUM, R1 ; DRIVE NUMBER TO R1
4323 ; JSR PC, POPQUE ; CALL TO REMOVE REQUEST
4324 ; RETURN ; R2=ADDRESS OF DPB REMOVED
4325
4326 020444 105361 017756 POPQUE: DECB QCNT(R1) ; DECREMENT QUEUE COUNT
4327 020450 006301 ASL R1
4328 020452 017102 020006 MOV QOUTPT(R1), R2 ; GET THE "DPB" POINTER
4329 020456 062761 000002 020006 020006 ADD #2, QOUTPT(R1) ; UPDATE THE QUEUE POINTER
4330 020464 026161 020006 020030 CMP QOUTPT(R1), QSTOP(R1) ; TIME TO RESET THE POINTER?
4331 020472 001003 BNE 1$ ; NO--BRANCH TO EXIT
4332 020474 016161 020026 020006 MOV QSTART(R1), QOUTPT(R1) ; YES--RESET THE POINTER
4333 020502 006201 1$: ASR R1
4334 020504 000207 RTS PC ; RETURN TO USER
4335
4336 ; ROUTINE TO SAVE THE CONTENTS OF 'SESCAPE' WHEN THE DRIVER
4337 ; REPORTS AN ERROR DIRECTLY.
4338
4339 ; CALL
4340 ; JSR R0, ES.SAV ; THE ERROR CALL
4341 ; ERROR N ; THE RETURN IS PAST THE ERROR CALL
4342 ; RETURN
4343

```

4344	020506	012037	020522
4345	020512	013746	001160
4346	020516	005037	001160
4347	020522	000000	
4348	020524	012637	001160
4349	020530	000200	

```

ES.SAV: MOV      (R0)+,1$      ;GET THE ERROR CALL
          MOV      $ESCAPE,-(SP) ;SAVE THE ADDRESS IN '$ESCAPE'
          CLR      $ESCAPE      ;CLEAR THE ESCAPE RETURN
1$:      .WORD    0             ;THE ERROR CALL IS MOVED HERE
          MOV      (SP)+,$ESCAPE ;RESTORE THE ESCAPE ADDRESS
          RTS      R0           ;RETURN

```

;*****

4350
4351
4352
4353
4354
4355

.SBTTL TELETYPE MESSAGES

4356	020532	047440	043106	044514
4357	020540	042516	000	
4358	020543	040	047117	044514
4359	020550	042516	000	
4360	020553	040	047516	020124
4361	020560	047101	051040	030120
4362	020566	027464	027465	000066
4363	020574	047040	052117	050040
4364	020602	042522	042523	052116
4365	020610	000		
4366	020611	040	047125	040523
4367	020616	042506	000	
4368	020621	122	030120	000064
4369	020626	050122	032460	000
4370	020633	122	030120	000066
4371	020640	047125	052111	051440
4372	020646	040524	052524	006523
4373	020654	005012	000	
4374	020657	015	042012	044522
4375	020664	042526	020072	000
4376	020671	040	020057	000
4377	020675	103	000	
4378	020677	124	000	
4379	020701	060	000	
4380	020703	040	040	
4381	020705	040	000040	
4382	020710	047105	042524	020122
4383	020716	042101	051104	051505
4384	020724	020123	044514	044515
4385	020732	051524	006472	000012
4386	020740	020040	051104	053111
4387	020746	021105	043117	046106
4388	020754	047111	006505	000012
4389	020762	042040	044522	042526
4390	020770	047040	052117	050040
4391	020776	042522	042523	052116
4392	021004	005015	000	
4393	021007	040	051104	053111
4394	021014	020105	047516	020124
4395	021022	053101	044501	040514
4396	021030	046102	006505	000012
4397	021036	042040	044522	042526
4398	021044	047040	052117	040440
4399	021052	020116	050122	032060

```

UNTOFF: .ASCIZ / OFFLINE/
UNTON:  .ASCIZ / ONLINE/
NOTRP:  .ASCIZ @ NOT AN RPO4/5/6@
NOTPRS: .ASCIZ / NOT PRESENT/
NOTSAF: .ASCIZ / UNSAFE/
RPO4B:  .ASCIZ /RPO4/
RPO5:   .ASCIZ /RPO5/
RPO6:   .ASCIZ /RPO6/
SYSTAT: .ASCIZ /UNIT STATUS/<CR><LF><LF>
MUNIT:  .ASCIZ <CR><LF>/DRIVE: /
SLASH:  .ASCIZ @ / @
C:       .ASCIZ /C/
T:       .ASCIZ /T/
MORVD:  .ASCIZ /O/
LIN4SP: .ASCII / /
LINSR:  .ASCIZ / /
ENTADR: .ASCIZ /ENTER ADDRESS LIMITS:/<CR><LF>
MOFFLN. .ASCIZ / DRIVE OFFLINE/<CR><LF>
MORNP:  .ASCIZ / DRIVE NOT PRESENT/<CR><LF>
MER11:  .ASCIZ / DRIVE NOT AVAILABLE/<CR><LF>
MNRPO4: .ASCIZ @ DRIVE NOT AN RPO4/5/6@<CR><LF>

```

4400	021060	032457	033057	005015	
4401	021066	000			
4402	021067	040	051104	053111	MUSDR: .ASCIZ / DRIVE UNSAFE/<CR><LF>
4403	021074	020105	047125	040523	
4404	021102	042506	005015	000	
4405	021107	040	042523	042514	MSELD: .ASCIZ / SELECTED/
4406	021114	052103	042105	000	
4407	021121	015	050012	047522	MMODE: .ASCIZ <CR><LF>/PROGRAM MODE (C OR F): /
4408	021126	051107	046501	046440	
4409	021134	042117	020105	041450	
4410	021142	047440	020122	024506	
4411	021150	020072	000		
4412	021153	040	047506	046522	MFORMAT: .ASCIZ / FORMAT & VERIFY/
4413	021160	052101	023040	053040	
4414	021166	051105	043111	000131	
4415	021174	044103	041505	020113	MHECK: .ASCIZ /CHECK ONLY/
4416	021202	047117	054514	000	
4417	021207	106	051117	040515	MORMAT: .ASCIZ /FORMAT & VERIFY/
4418	021214	020124	020046	042526	
4419	021222	044522	054506	000	
4420	021227	015	005012	050117	MSIZE: .ASCIZ <CR><LF><LF>/OPERATE IN 22 SECTOR MODE (Y OR N) ? /
4421	021234	051105	052101	020105	
4422	021242	047111	031040	020062	
4423	021250	042523	052103	051117	
4424	021256	046440	042117	020105	
4425	021264	054450	047440	020122	
4426	021272	024516	037440	000040	
4427	021300	050117	051105	052101	MSEC22: .ASCIZ /OPERATION WILL BE IN 22 SECTOR (16 BIT) MODE/<CR><LF>
4428	021306	047511	020116	044527	
4429	021314	046114	041040	020105	
4430	021322	047111	031040	020062	
4431	021330	042523	052103	051117	
4432	021336	024040	033061	041040	
4433	021344	052111	020051	047515	
4434	021352	042504	005015	000	
4435	021357	117	042520	040522	MSEC20: .ASCIZ /OPERATION WILL BE IN 20 SECTOR (18 BIT) MODE/<CR><LF>
4436	021364	044524	047117	053440	
4437	021372	046111	020114	042502	
4438	021400	044440	020116	030062	
4439	021406	051440	041505	047524	
4440	021414	020122	030450	020070	
4441	021422	044502	024524	046440	
4442	021430	042117	006505	000012	
4443	021436	047111	040526	044514	BADENT: .ASCIZ /INVALID ENTRY/<CR><LF>
4444	021444	020104	047105	051124	
4445	021452	006531	000012		
4446	021456	047105	044504	043516	MADRER: .ASCII /ENDING DSK ADRS MUST BE EQUAL TO OR GREATER/<CR><LF>
4447	021464	042040	045523	040440	
4448	021472	051104	020123	052515	
4449	021500	052123	041040	020105	
4450	021506	050505	040525	020114	
4451	021514	047524	047440	020122	
4452	021522	051107	040505	042524	
4453	021530	006522	012		
4454	021533	124	040510	020116	.ASCIZ /THAN STARTING ADRS/<CR><LF>
4455	021540	052123	051101	044524	

4456	021546	043516	040440	051104	
4457	021554	006523	000012		
4458	021560	042523	042514	052103	MSELP: .ASCII /SELECT DATA PATTERN/<CR><LF>
4459	021566	042040	052101	020101	
4460	021574	040520	052124	051105	
4461	021602	006516	012		
4462	021605	040	030050	020051	.ASCII / (0) ZERO'S/<CR><LF>
4463	021612	042532	047522	051447	
4464	021620	005015			
4465	021622	024040	024461	047440	.ASCII / (1) ONE'S/<CR><LF>
4466	021630	042516	051447	005015	
4467	021636	024040	024462	053440	.ASCIZ / (2) WORST CASE: /
4468	021644	051117	052123	041440	
4469	021652	051501	035105	000040	
4470					
4471	021660	047527	051522	020124	MPATD: .ASCIZ /WORST CASE/
4472	021666	040503	042523	000	
4473	021673	015	005012	052123	MSFOU: .ASCIZ <CR><LF><LF>/STARTING FORMAT ON DRIVE /
4474	021700	051101	044524	043516	
4475	021706	043040	051117	040515	
4476	021714	020124	047117	042040	
4477	021722	044522	042526	000040	
4478	021730	005015	051412	040524	MSCHK: .ASCIZ <CR><LF><LF>/STARTING CHECK ON DRIVE /
4479	021736	052122	047111	020107	
4480	021744	044103	041505	020113	
4481	021752	047117	042040	044522	
4482	021760	042526	000040		
4483	021764	005015	043012	051117	MFCMPT: .ASCIZ <CR><LF><LF>/FORMAT COMPLETE, /
4484	021772	040515	020124	047503	
4485	022000	050115	042514	042524	
4486	022006	020054	000		
4487	022011	015	005012	044103	MCCMPT: .ASCIZ <CR><LF><LF>/CHECK COMPLETE, /
4488	022016	041505	020113	047503	
4489	022024	050115	042514	042524	
4490	022032	020054	000		
4491	022035	124	052117	046101	NUMERR: .ASCIZ /TOTAL ERRORS DETECTED: /
4492	022042	042440	051122	051117	
4493	022050	020123	042504	042524	
4494	022056	052103	042105	020072	
4495	022064	000			
4496	022065	120	042522	042523	ADDRIS: .ASCIZ /PRESENT ADDRESS IS: /
4497	022072	052116	040440	042104	
4498	022100	042522	051523	044440	
4499	022106	035123	000040		
4500					
4501					::*****
4502					.SBTTL ERROR MESSAGES
4503					::*****
4504					
4505					
4506					
4507	022112	044122	030461	044440	EMI: .ASCIZ /RH11 INTERRUPT OCCURRED (RPAS=0)/
4508	022120	052116	051105	052522	
4509	022126	052120	047440	041503	
4510	022134	051125	042522	020104	
4511	022142	051050	040520	036523	

4512	022150	024460	000			
4513	022153	125	042516	050130	EM2:	.ASCIZ /UNEXPECTED ATTENTION OCCURRED/
4514	022160	041505	042524	020104		
4515	022166	052101	042524	052116		
4516	022174	047511	020116	041517		
4517	022202	052503	051122	042105		
4518	022210	000				
4519	022211	115	051501	041123	EM3:	.ASCIZ /MASSBUS PARITY ERROR (MCPE=1)/
4520	022216	051525	050040	051101		
4521	022224	052111	020131	051105		
4522	022232	047522	020122	046450		
4523	022240	050103	036505	024461		
4524	022246	000				
4525	022247	115	051501	041123	EM4:	.ASCIZ /MASSBUS PARITY ERROR (PAR=1)/
4526	022254	051525	050040	051101		
4527	022262	052111	020131	051105		
4528	022270	047522	020122	050050		
4529	022276	051101	030475	000051		
4530	022304	042101	051104	051505	EM5:	.ASCIZ /ADDRESS PLUG CHANGE BIT SET/
4531	022312	020123	046120	043525		
4532	022320	041440	040510	043516		
4533	022326	020105	044502	020124		
4534	022334	042523	000124			
4535	022340	044122	030461	042040	EM6:	.ASCIZ /RH11 DIDN'T RESPOND TO ADDRESSING/
4536	022346	042111	023516	020124		
4537	022354	042522	050123	047117		
4538	022362	020104	047524	040440		
4539	022370	042104	042522	051523		
4540	022376	047111	000107			
4541	022402	051104	053111	020105	EM10:	.ASCIZ /DRIVE OFFLINE/
4542	022410	043117	046106	047111		
4543	022416	000105				
4544	022420	042520	051522	051511	EM11:	.ASCIZ /PERSISTENT DRIVE UNSAFE ERROR/
4545	022426	042524	052116	042040		
4546	022434	044522	042526	052440		
4547	022442	051516	043101	020105		
4548	022450	051105	047522	000122		
4549	022456	047125	047503	051122	EM12:	.ASCIZ /UNCORRECTABLE MASSBUS PARITY ERROR/
4550	022464	041505	040524	046102		
4551	022472	020105	040515	051523		
4552	022500	052502	020123	040520		
4553	022506	044522	054524	042440		
4554	022514	051122	051117	000		
4555	022521	123	043117	053524	EM13:	.ASCIZ /SOFTWARE TIMEOUT/
4556	022526	051101	020105	044524		
4557	022534	042515	052517	000124		
4558	022542	051104	053111	020105	EM14:	.ASCIZ /DRIVE UNSAFE ERROR/
4559	022550	047125	040523	042506		
4560	022556	042440	051122	051117		
4561	022564	000				
4562	022565	103	047117	051124	EM15:	.ASCIZ @CONTROLLER/DRIVE ERROR DURING WRITED
4563	022572	046117	042514	027522		
4564	022600	051104	053111	020105		
4565	022606	051105	047522	020122		
4566	022614	052504	044522	043516		
4567	022622	053440	044522	042524		

4568	022630	000			
4569	022631	103	047117	051124	EM16: .ASCIZ @CONTROLLER/DRIVE ERROR DURING WRITE CHECK@
4570	022636	046117	042514	027522	
4571	022644	051104	053111	020105	
4572	022652	051105	047522	020122	
4573	022660	052504	044522	043516	
4574	022666	053440	044522	042524	
4575	022674	041440	042510	045503	
4576	022702	000			
4577	022703	122	052105	044522	EM17: .ASCIZ @RETRIES NOT SUCCESSFUL - SECTOR NOT ACCEPTABLE@
4578	022710	051505	047040	052117	
4579	022716	051440	041525	051505	
4580	022724	043123	046125	026440	
4581	022732	051440	041505	047524	
4582	022740	020122	047516	020124	
4583	022746	041501	042503	052120	
4584	022754	041101	042514	000	
4585	022761	104	052101	020101	EM20: .ASCIZ @DATA ERROR DURING WRITE CHECK@
4586	022766	051105	047522	020122	
4587	022774	052504	044522	043516	
4588	023002	053440	044522	042524	
4589	023010	041440	042510	045503	
4590	023016	000			
4591	023017	103	047117	051124	EM21: .ASCIZ @CONTROLLER/DRIVE ERROR VERIFYING HEADERS@
4592	023024	046117	042514	027522	
4593	023032	051104	053111	020105	
4594	023040	051105	047522	020122	
4595	023046	042526	044522	054506	
4596	023054	047111	020107	042510	
4597	023062	042101	051105	000123	
4598	023070	042510	042101	051105	EM22: .ASCIZ @HEADER COMPARE ERROR VERIFYING HEADERS@
4599	023076	041440	046517	040520	
4600	023104	042522	042440	051122	
4601	023112	051117	053040	051105	
4602	023120	043111	044531	043516	
4603	023126	044040	040505	042504	
4604	023134	051522	000		
4605	023137	103	046131	047111	EM23: .ASCIZ @CYLINDER FIELD IN HEADER IS NOT CORRECT@
4606	023144	042504	020122	044506	
4607	023152	046105	020104	047111	
4608	023160	044040	040505	042504	
4609	023166	020122	051511	047040	
4610	023174	052117	041440	051117	
4611	023202	042522	052103	000	
4612	023207	127	044522	042524	EM24: .ASCIZ @WRITE CHECK ERROR@
4613	023214	041440	042510	045503	
4614	023222	042440	051122	051117	
4615	023230	000			
4616					
4617					
4618					
4619	023231	122	040520	000123	DH1: .ASCIZ /RPAS/
4620	023236	051104	053111	020105	DH2: .ASCIZ /DRIVE RPOS1 RPER1 RPER2 RPER3 RPAS/
4621	023244	020040	050122	051504	
4622	023252	020061	020040	050122	
4623	023260	051105	020061	020040	

;;*****

4736	024444	052040	040522	045503			
4737	024452	020040	051440	041505			
4738	024460	047524	020122	042040			
4739	024466	052101	020101	020040			
4740	024474	042040	052101	000101			
4741							
4742							
4743	024502	001276			DT1:	.WORD	ATTN
4744	024504	001274	012100	012102	DT2:	.WORD	DDRIVE, RPERRS, RPERRS+2, RPERRS+4, RPERRS+6, ATTN
4745	024512	012104	012106	001276			
4746	024520	001274	017240	017242	DT3:	.WORD	DDRIVE, RD. ADR, RD. WRD
4747	024526	001274	017464	017462	DT4:	.WORD	DDRIVE, WRT. AD, WRT. WD, RD. WRD
4748	024534	017242					
4749	024536	001172			DT6:	.WORD	\$RPADR
4750	024540	001214	001116	001304	DT10:	.WORD	DRIVE, \$ERRPC, RP. REG, RP. REG+10, RP. REG+12, RP. REG+14, RP. REG+40, RP. REG+42
4751	024546	001314	001316	001320			
4752	024554	001344	001346				
4753	024560	001350	001352	001306	.WORD		RP. REG+44, RP. REG+46, RP. REG+2, RP. REG+4, RP. REG+6, RP. REG+16, RP. REG+20
4754	024566	001310	001312	001322			
4755	024574	001324					
4756	024576	001326	001330	001332	.WORD		RP. REG+22, RP. REG+24, RP. REG+26, RP. REG+30, RP. REG+32, RP. REG+34, RP. REG+36
4757	024604	001334	001336	001340			
4758	024612	001342					
4759	024614	001214	001116	001270	DT17:	.WORD	DRIVE, \$ERRPC, DS. CYL, DS. TRK, SAVSEC
4760	024622	001272	001252				
4761	024626	001214	001116	001270	DT20:	.WORD	DRIVE, \$ERRPC, DS. CYL, DS. TRK, SAVSEC
4762	024634	001272	001252				
4763	024640	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
4764	024646	001320	001344	001346			
4765	024654	001350	001352				
4766	024660	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
4767	024666	001322	001324	001326			
4768	024674	001330	001332				
4769	024700	001334	001336	001340	.WORD		RP. REG+30, RP. REG+32, RP. REG+34, RP. REG+36
4770	024706	001342					
4771	024710	001214	001116	025130	DT23:	.WORD	DRIVE, \$ERRPC, BUFP, RBUF, RBUF+2, RBUF+4, RBUF+6
4772	024716	025120	025122	025124			
4773	024724	025126					
4774	024726	001214	001116	001270	DT24:	.WORD	DRIVE, \$ERRPC, DS. CYL, DS. TRK, SAVSEC, \$GDDAT, RP. REG+22
4775	024734	001272	001252	001124			
4776	024742	001326					
4777	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
4778	024752	001320	001344	001346			
4779	024760	001350	001352				
4780	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
4781	024772	001322	001324	001326			
4782	025000	001330	001332				
4783	025004	001334	001336	001340	.WORD		RP. REG+30, RP. REG+32, RP. REG+34, RP. REG+36
4784	025012	001342					
4785							
4786	025014	000001			DF1:	.WORD	1
4787	025016	001	000			.BYTE	1,0
4788	025020	000001			DF2:	.WORD	1
4789	025022	006	000			.BYTE	6,0
4790	025024	000001			DF3:	.WORD	1
4791	025026	003	000			.BYTE	3,0

4792	025030	000001		
4793	025032	004	000	
4794	025034	000003		
4795	025036	010	000	
4796	025040	023511		
4797	025042	007	000	
4798	025044	023576		
4799	025046	007	000	
4800	025050	000001		
4801	025052	005	034	
4802	025054	000004		
4803	025056	005	034	
4804	025060	024001		
4805	025062	010	000	
4806	025064	024077		
4807	025066	010	000	
4808	025070	024174		
4809	025072	004	000	
4810	025074	000001		
4811	025076	007	000	
4812	025100	000004		
4813	025102	007	034	
4814	025104	024001		
4815	025106	010	000	
4816	025110	024077		
4817	025112	010	000	
4818	025114	024174		
4819	025116	004	000	
4820				
4821				
4822				
4823				
4824				
4825				
4826				
4827	025120	000000	000000	000000
4828	025126	000000		
4829				
4830	025130			
4831				
4832	025130	005015	055012	026532
4833	025136	055103	045122	026502
4834	025144	006502	012	
4835	025147	122	030120	027464
4836	025154	027465	020066	047506
4837	025162	046522	052101	042524
4838	025170	020122	051120	043517
4839	025176	040522	006515	005012
4840	025204	000		
4841	025205	015	052012	020117
4842	025212	043047	051117	040515
4843	025220	023524	047440	020122
4844	025226	041447	042510	045503
4845	025234	020047	051104	053111
4846	025242	020105	026060	051040
4847	025250	050105	040514	042503

```

DF4: .WORD 1
      .BYTE 4,0
DF10: .WORD 3,0
      .BYTE 8,0
      .WORD DH10A
      .BYTE 7,0
      .WORD DH10B
      .BYTE 7,0
DF17: .WORD 1
      .BYTE 5,34
DF20: .WORD 4
      .BYTE 5,34
      .WORD DH20A
      .BYTE 8,0
      .WORD DH20B
      .BYTE 8,0
      .WORD DH20C
      .BYTE 4,0
DF23: .WORD 1
      .BYTE 7,0
DF24: .WORD 4
      .BYTE 7,34
      .WORD DH20A
      .BYTE 8,0
      .WORD DH20B
      .BYTE 8,0
      .WORD DH20C
      .BYTE 4,0

```

;;*****

;BUFFER STARTS HERE

;;*****

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

BUFP: ;FORMAT AND CHECK BUFFER STARTS HERE

TITLE: .ASCII <CR><LF><LF>/ZZ-CZRJB-B/<CR><LF>

.ASCIZ RP04/5/6 FORMATTER PROGRAM<CR><LF><LF>

LOADRV: .ASCII <CR><LF>/TO 'FORMAT' OR 'CHECK' DRIVE 0, REPLACE THE 'XXDP'<CR><LF>

4848	025256	052040	042510	023440
4849	025264	054130	050104	006447
4850	025272	012		
4851	025273	120	041501	020113
4852	025300	047117	042040	044522
4853	025306	042526	030040	053440
4854	025314	052111	020110	047101
4855	025322	052117	042510	020122
4856	025330	040520	045503	020054
4857	025336	046103	040503	020122
4858	025344	042515	047515	054522
4859	025352	046040	041517	052101
4860	025360	047511	020116	030064
4861	025366	006454	012	
4862	025371	101	042116	051040
4863	025376	051505	040524	052122
4864	025404	052040	042510	050040
4865	025412	047522	051107	046501
4866	025420	005015	000	
4867				
4868				
4869				
4870				
4871				
4872				
4873				
4874				
4875				
4876				
4877				
4878				
4879				
4880	025424	177700		
4881	025426	000776		
4882	025430	000000		
4883	025432	005737	001302	
4884	025436	001460		
4885	025440	005037	001302	
4886	025444	012700	001172	
4887	025450	104401	025652	
4888	025454	012046		
4889	025456	104402		
4890	025460	104401	020705	
4891	025464	104411		
4892	025466	012601		
4893	025470	013737	025424	025430
4894	025476	004537	025674	
4895	025502	025522		
4896	025504	025600		
4897	025506	025444		
4898	025510	025516		
4899	025512	025444		
4900	025514	025574		
4901	025516	010260	177776	
4902	025522	104401	025663	
4903	025526	012700	001174	

.ASCII /PACK ON DRIVE 0 WITH ANOTHER PACK, CLEAR MEMORY LOCATION 40, /<CR><LF>

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

```

.SBTTL 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
;
HIAD: .WORD 177700
HIVEC: .WORD 776
LIMIT: .WORD
BUSADR: TST CHGADR ; INPUT FROM TTY REQUESTED?
        BEQ 7$ ; NO--BRANCH
        CLR CHGADR ; YES--CLEAR THE REQUEST FLAG
        MOV #SRPADR,R0 ; FIRST ADDRESS
        TYPE MRPCS1 ; "RPCS1="
        MOV (R0)+,-(SP) ; PRESENT RPCS1 ADDRESS
        TYPOC ; TYPE IT
        TYPE ,LINSF ; 2 SPACES
        RDLIN ; GET THE ENTRY
        MOV (SP)+,R1 ; ADDRESS OF ASCII TEXT
        MOV HIAD,LIMIT ; THIS IS THE ADDRSS HIGH LIMIT
        JSR R5,CK.NUM ; CHECK THE NUMBER
        3$ ; CARRIAGE RETURN ONLY ENTERED
        7$ ; PERIOD ONLY ENTERED
        1$ ; ILLEGAL INPUT
        2$ ; TERMINATED WITH A CARRIAGE RETURN
        1$ ; TERMINATED WITH A "."
        4$ ; TERMINATED WITH A "!"
        2$: MOV R2,-2(R0) ; SAVE NEW RPCS1
        3$: TYPE MRHVEC ; "RHVEC="
        MOV #SRPVEC,R0 ; FIRST VECTOR

```

```

4904 025532 012046          MOV      (R0)+, -(SP)      ;PRESENT RH11 VECTOR ADDRESS ON THE STACK
4905 025534 104402          TYP0C          ;TYPE IT
4906 025536 104401 020705     TYPE          ,LINSF      ;2 SPACES
4907 025542 104411          RDLIN          ;READ THE ENTRY
4908 025544 012601          MOV      (SP)+, R1        ;ASCII TEXT ADDRESS
4909 025546 013737 025426 025430  MOV      HIVEC, LIMIT     ;VECTOR LIMIT
4910 025554 004537 025674  JSR      R5, CK.NUM      ;CHECK THE NUMBER
4911 025560 025600          7$          ;CARRIAGE RETURN ONLY ENTERED
4912 025562 025600          7$          ;PERIOD ONLY ENTERED
4913 025564 025522          3$          ;ILLEGAL INPUT
4914 025566 025574          4$          ;TERMINATED WITH A CARRIAGE RETURN
4915 025570 025522          3$          ;TERMINATED WITH A "."
4916 025572 025574          4$          ;TERMINATED WITH A "!"
4917 025574 010260 177776     4$: MOV      R2, -2(R0)      ;SAVE INPUT
4918 025600 013701 000004     7$: MOV      ERV0C, R1    ;SAVE THE ERROR VECTOR
4919 025604 012737 025640 000004  MOV      #B$, ERV0C     ;SETUP FOR TRAP
4920 025612 005777 153354     JSR      #SAPADR        ;CHECK FOR RH11
4921 025616 010137 000004     MOV      R1, ERV0C      ;RESTORE ERROR VECTOR
4922 025622 012700 001172     MOV      #SAPADR, R0    ;FIRST ADDRESS OF NEW PARAMETERS
4923 025626 012701 012246     MOV      #RPADR, R1     ;FIRST ADDRESS OF WHERE TO PUT THEM
4924 025632 012021          MOV      (R0)+, (R1)+   ;BUS ADDRESS
4925 025634 012021          MOV      (R0)+, (R1)+   ;VECTOR ADDRESS
4926 025636 000207          RTS          PC          ;RETURN
4927 025640 010137 000004     8$: MOV      R1, ERV0C   ;RESTORE ERROR VECTOR
4928 025644 022626          CMP      (SP)+, (SP)+   ;CLEAN OFF THE STACK
4929 025646 104006          ERROR      6          ;REPORT THE ERROR
4930 025650 000675          BR       1$            ;ASK FOR BUS ADDRESS
4931
4932 025652 050122 051503 020061  MRPCS1: .ASCIZ  @RPCS1 = @
4933 025660 020075          000
4934 025663 122 053110 041505  MRHVEC: .ASCIZ  @RHVEC = @
4935 025670 036440 000040
4936
4937          ;SBTTL CK.NUM - CHECK NUMBER (OCTAL)
4938          ;THIS ROUTINE CHECKS AN ASCIZ STRING FOR LEGAL CHARACTERS
4939          ;AND FORMS AN OCTAL NUMBER IN R2
4940          ;CALL:
4941          ;      MOV      #ADR, R1      ;ADDRESS OF ASCIZ STRING
4942          ;      MOV      #NUM, R2     ;MAX SIZE OF INPUT NUMBER
4943          ;      JSR      R5, CK.NUM   ;GO FORM THE NUMBER
4944          ;      RETURN  ADR1        ;"CR" ONLY ENTERED -- R2 = 0
4945          ;      RETURN  ADR2        ;"PERIOD" ONLY ENTERED -- R2 = 0
4946          ;      RETURN  ADR3        ;ILLEGAL CHARACTER IN THE INPUT STRING
4947          ;      RETURN  ADR4        ;"CR" ENTERED -- R2 = NUMBER
4948          ;      RETURN  ADR5        ;"COMMA" -- R2 = NUMBER
4949          ;      RETURN  ADR6        ;"PERIOD" -- R2 = NUMBER
4950
4951 025674 010446          CK.NUM: MOV      R4, -(SP)  ;SAVE R4
4952 025676 010346          MOV      R3, -(SP)    ;SAVE R3
4953 025700 010246          MOV      R2, -(SP)    ;SAVE R2
4954 025702 005004          CLR      R4          ;RETURN POINTER
4955 025704 005003          CLR      R3          ;START NUMBER A' ZERO
4956 025706 005002          CLR      R2          ;STORE RESULT
4957 025710 004537 006426  JSR      R5, CK.CHR   ;CHECK ONE CHARACTER
4958 025714 026014          6$          ;ILLEGAL CHARACTER
4959 025716 116020          8$          ;CARRIAGE RETURN

```


ABS = 000200
 ACK = 000123
 ACL = 000040
 ACTDRV 012164
 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
 AT5 = 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
 BIT5 = 000040
 BIT6 = 000100
 BIT7 = 000200

BIT8 = 000400
 BIT9 = 001000
 BPTVEC = 000014
 BUFP 025130
 BUSADR 025432
 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 025674
 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
 DFF20 = 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
 OPR = 000400
 DPRQS 012150
 DRIVE 001214
 DRQ = 004000
 DRVACT 012110
 DRVCLR = 000111
 DRVINT 012476
 DRVQUE 020346
 DRVSTA 012120
 DRVSTYP 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
 OVA = 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
 GO = 000001
 GPV = 000010
 GTSWR = 104406
 HCE = 000200

HCI = 002000
 HCRC = 000400
 HDREAD 004650
 HEDERR 001262
 HIAD 025424
 HIVEC 025426
 HT = 000011
 IAE = 002000
 IE = 000100
 ILF = 000001
 ILR = 000002
 IOTVEC = 000020
 IR = 000100
 ISR 015130
 IXE = 004000
 LA 014772
 LACNT 012176
 LF = 000012
 LIMIT 025430
 LINSR 020705
 LIN4SP 020703
 LOADRV 025205
 LOP.CK 005334
 LST = 002000
 MADRER 021456
 MAXSEC 001266
 MCCMPT 022011
 MCLK = 000002
 MCPE = 020000
 MCPMX 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 025663
 MRPCS1 025652
 MSCHK 021730
 MSE = 000020

MSEC20 021357
MSEC22 021300
MSELO 021107
MSELP 021560
MSFOU 021673
MSIZE 021227
MSTCK = 000010
MUNIT 020657
MUSDR 021067
MWC 001260
MWR = 000040
MXDLTA 012256
MXF = 001000
MXLACT 012254
MXWINDW 012262
M0 003116
M1 002644
M1A 002734
M1B 003062
M2 003404
M4 003444
M5 003526
MBA = 100000
MED = 010000
MEM = 004000
MHS = 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
OPT 013326
OP = 000200
PAR = 000010
PARENT 006236
PAR1 001426
PAR2 001441
PAR3 001454
PAR4 001465
PAT = 000020
PATA 001242
PATE 001244

PATSEL 001240
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
ROCHR = 104410
RODAT = 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
RPOB = 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
RPO5 020626
RPO6 020633
RTC = 000117
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
SEFK = 000105
SEEKFG 012210
SELDIV = 000145
SETFMT = 000143
SETHOR 005602
SETPAT 003700
SETTBL 005402
SETVEC 002410
SET.IE 017704
SKI = 040000
SLASH 020671
SOFSW 001216
SACHWT 012162
STACK = 001100
STCLK1 006020
STCLK2 006070
STCLK3 006074
STKLMT = 177774
STO 016560
STO1 016610
STO2 017006
STO3 017056
STO5 017102
STO6 017110
STO7 017146
STO8 017176
STO9 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 001326
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
UL.FLG 012166
UNLOAD = 000103
UNS = 040000
UNTOFF 020532
UNTON 020543
UPDACY 005662
UPDATK 005712
UPE = 020000
US1 = 000001
US2 = 000002
US4 = 000004

JWR = 000010	WRT.WD = 017462	\$ERROR = 006636	\$OCNT = 007736	\$TKSRV = 010254
VUF = 000002	WRU = 000400	\$ERRPC = 001116	\$OMODE = 007740	\$TN = 000000
VU30 = 010000	WSU = 000004	\$ERRTB = 001626	\$PASS = 001100	\$TNPWR = 011634
VV = 000100	\$AUTOB = 001134	\$ERTTL = 001112	\$QUES = 001166	\$TPB = 001152
WAO = 000002	\$BDADR = 001122	\$ESCAP = 001160	\$RDCHR = 011034	\$TPFLG = 001157
WC = 001256	\$BDDAT = 001126	\$FILLC = 001156	\$RDLIN = 011124	\$TPS = 001150
WCE = 040000	\$BELL = 001162	\$FILLS = 001155	\$RDSZ = 000007	\$TRAP = 012014
WCF = 000040	\$CHARC = 007510	\$GDADR = 001120	\$RESRE = 011756	\$TRAP2 = 012036
WCKD = 000151	\$CKSWR = 010472	\$GDDAT = 001124	\$RPADR = 001172	\$TRP = 000014
WCKHD = 000153	\$CMTAG = 001100	\$GET42 = 005164	\$RPVEC = 001174	\$TRPAD = 012050
WCTBL = 001552	\$CM3 = 000000	\$GTSWR = 010562	\$RTNAD = 005206	\$TSTNM = 001102
WCU = 000001	\$CNTLC = 011367	\$HD = 000000	\$SAVRE = 011720	\$TTYIN = 011360
WLE = 004000	\$CNTLG = 011401	\$ICNT = 001104	\$SB2D = 011470	\$TYPDS = 007742
WRL = 004000	\$CNTLU = 011374	\$INTAG = 001135	\$SETUP = 000106	\$TYPE = 007274
WRTDAT = 000161	\$CRLF = 001167	\$ITEMB = 001114	\$STUP = 177777	\$TYPEC = 007444
WRTHD = 000163	\$DBLK = 010156	\$LF = 001170	\$SUPRS = 011430	\$TYPEX = 007512
WRTRK = 004004	\$DB2D = 011524	\$LKCSB = 001200	\$SVPC = 000200	\$TYPC = 007540
WRTRK1 = 004060	\$DECVL = 011704	\$LKCSR = 001176	\$SWR = 123000	\$TYPON = 007554
WRTRK2 = 004116	\$DOAGN = 005204	\$LKS = 001206	\$TKB = 001146	\$TYPOS = 007514
WRT.R0 = 017464	\$DTBL = 010146	\$LLVEC = 001210	\$TKCNT = 010166	\$GET4 = 000000
WRT.RP = 017374	\$ENDAD = 005174	\$LPADR = 001106	\$TKINT = 010204	\$OFILL = 007737
WRT.R1 = 017460	\$ENDCT = 005160	\$LPERR = 001110	\$TKQEN = 010203	. = 026036
WRT.R2 = 017550	\$EOP = 005100	\$LPVEC = 001202	\$TKQIN = 010170	
WRT.R3 = 017556	\$EOPCT = 005152	\$MNEW = 011417	\$TKQOU = 010172	
WRT.R4 = 017562	\$ERFLG = 001103	\$MSWR = 011406	\$TKQSR = 010174	
WRT.R5 = 017564	\$ERMAX = 001115	\$NULL = 001154	\$TKS = 001144	

. ABS. 026036 000

ERRORS DETECTED: 0

RM03:CZRJBB,CZRJBB.SEG/SOL/LI:ME/NL:MC:MD:CND=RM03:RPO456.011,CZRJBB.P11
RUN-TIME: 22 14 .7 SECONDS
RUN-TIME RATIO: 777/37=20.7
CORE USED: 36K (71 PAGES)

F08