

RP04/5/6

RP04/5/6 FMTR
CZRJBDO

AH-9187D-MC

COPYRIGHT 74-79

FICHE 1 OF 1

SEP 1979

digital

MADE IN USA

.REM @

I D E N T I F I C A T I O N

PRODUCT CODE: AC-9185D-MC

PRODUCT NAME: CZRJBD0 RP04/5/6 FORMATTER PROGRAM

PRODUCT DATE: MAY, 1979

MAINTAINER: DIAGNOSTIC ENGINEERING

AUTHOR: C. HESS

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

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

COPYRIGHT (C) 1974, 1979 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGIAL
DEC

PDP
DECUS

UNIBUS
DECTAPE

MASSBUS

CONTENTS

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

1. ABSTRACT

THE RP04/5/6 FORMATTER PROGRAM FORMATS THE DISK PACK AND PERFORMS A CURSORY CHECK OF THE PACK'S SURFACE. THE PROGRAM ALLOWS THE OPERATOR TO SPECIFY ADDRESS LIMITS, PATTERNS, AND EITHER 16 BIT OR 18 BIT FORMAT MODE. THE PROGRAM VERIFIES EACH TRACK WRITTEN AS WELL AS VERIFYING THE FORMAT OPERATION.

2. REQUIREMENTS

2.1 EQUIPMENT

PDP-11 PROCESSOR
8K MEMORY
TELETYPE
PROGRAM LOAD DEVICE
KW11-L OR KW11-P CLOCK
RH11 OR RH70 WITH 1 - 8 RP04, RP05, RP06 DISK DRIVES

2.2 PRELIMINARY PROGRAMS

RP04/5/6 DISKLESS CONTROLLER TEST
PART 1 (MAINDEC-11-DZRJG)
PART 2 (MAINDEC-11-DZRJH)

RP04/5/6 FUNCTIONAL CONTROLLER TEST
PART 1 (MAINDEC-11-DZRJI)
PART 2 (MAINDEC-11-DZRJJ)

2.3 PROGRAMMABLE DRIVES (DUAL PORT ENABLED)

THIS REV INCORPORATES A SAFEGUARD TO PREVENT INADVERTENT CORRUPTION OF DISK PACKS IN PROGRAMMABLE DRIVES. THIS IS A POTENTIAL HAZARD IN RUNNING THIS PROGRAM IN A MULTIPROCESSOR SYSTEM. FOR THE STANDARD STARTING ADDRESS OF 200 THE PROGRAM HAS BEEN MODIFIED TO PREVENT INITIALIZING DRIVES FOUND TO BE PROGRAMMABLE. THIS MODIFICATION APPLIES ONLY TO THE FIELD ENVIRONMENT (XXDP CHAIN, STANDALONE) WHERE LOCATION 42 DOES NOT EQUAL LOCATION 46. FOR THE MANUFACTURING ENVIRONMENT (WHERE LOCATION 42 EQUALS LOCATION 46) PROGRAMMABLE DRIVES WILL NOT BE INHIBITED. IF THE OPERATOR DESIRES TO RUN THIS PROGRAM USING PROGRAMMABLE DRIVES IN A FIELD ENVIRONMENT USE STARTING ADDRESS 220, WHERE 220 IS THE SAME AS 200 WITHOUT INHIBITING PROGRAMMABLE DRIVES. SEE SECTION 4.1 FOR A SUMMARY OF ALL STARTING ADDRESSES.

3. LOADING PROCEDURES

138 THE PROGRAM MAY BE LOADED FROM PAPER TAPE USING THE ABSOLUTE
139 LOADER OR IT MAY BE LOADED FROM THE APPROPRIATE 'XXDP' MEDIA
140 USING THE ASSOCIATED LOADER. THE PROGRAM MAY NOT BE INCLUDED
141 IN AN 'XXDP' CHAIN.
142
143
144
145
146

147 4. STARTING PROCEDURES
148 -----
149

150 4.1 STARTING ADDRESSES
151

152 THE PROGRAM IS STARTED FROM LOCATION 200(8) IF THE ADDRESS OF THE
153 RH11 OR RH70 WILL NOT BE CHANGED
154 PROGRAMMABLE DRIVES ARE INHIBITED, STANDALONE AND XXDP CHAIN-SEE SECTION 2.3
155

156 THE PROGRAM IS STARTED FROM LOCATION 204(8) IF THE ADDRESS OF
157 THE RH11 OR RH70 IS TO BE CHANGED FROM THE PRELOADED VALUE.
158 (SEE SECTION 4.3) PROGRAMMABLE DRIVES ARE NOT INHIBITED.
159

160 STARTING ADDRESS 220 IS THE SAME AS 200 BUT WITH NO INHIBITIONS.
161

162 4.2 OPERATION ACTION
163

- 164 1. LOAD THE PROGRAM INTO MEMORY (SEE SECTION 3).
165
166 2. LOAD THE STARTING ADDRESS - 200(8) OR 204(8) OR 220(8).
167
168 3. SET THE SWITCHES AS REQUIRED AND PRESS 'START'.
169

170 IF THIS IS THE PROGRAM'S FIRST START, THE STATUS OF THE DRIVES
171 ON THE SELECTED MASSBUS SUBSYSTEM WILL BE TYPED OUT. THIS TYPEOUT
172 MAY BE INHIBITED ON SUBSEQUENT STARTS BY SETTING SW<02>.
173

- 174 4. THE PROGRAM WILL THEN TYPE THE FOLLOWING MESSAGE:
175

176 'PROGRAM MODE (C OR F):'
177

178 ENTER THE APPROPRIATE CODE: 'C' FOR 'CHECK' OPERATION OR 'F'
179 FOR 'FORMAT & VERIFY'. IF A 'CARRIAGE RETURN' IS ENTERED IN
180 RESPONSE TO THE REQUEST, THE PROGRAM WILL ASSUME 'FORMAT &
181 VERIFY'.
182

- 183 5. THE PROGRAM WILL THEN ASK FOR THE FORMATTING MODE:
184

185 'OPERATE IN 22 SECTOR (16 BIT) MODE (Y OR N)'
186

187 ENTER THE APPROPRIATE CHARACTER: 'Y' FOR 16 BIT MODE OR 'N' FOR
188 18 BIT MODE. IF A 'CARRIAGE RETURN' IS ENTERED IN RESPONSE TO
189 THIS REQUEST, THE PROGRAM WILL ASSUME 16 BIT MODE.
190

- 191 6. THE PROGRAM WILL THEN ASK FOR A DRIVE:
192

193 'DRIVE: '
194

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

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

'ENTER ADDRESS LIMITS: '

THE PREVIOUSLY SELECTED OR DEFAULT VALUES FOR BEGINNING CYLINDER AND TRACK AND FOR ENDING CYLINDER AND TRACK WILL BE TYPED OUT. IF A 'CARRIAGE RETURN' IS TYPED AS A RESPONSE, THE PRESENT VALUE WILL BE USED; IF A 'PERIOD' IS TYPED, THE PROGRAM WILL BYPASS THE REMAINING ENTRIES AND WILL USE THEIR PRESENT VALUES. NOTE THAT THE CYLINDER AND TRACK VALUES ARE D E C I M A L NUMBERS. THE ADDRESS SPECIFIED BY THE BEGINNING CYLINDER AND TRACK MUST BE LESS THAN THE ADDRESS SPECIFIED BY THE ENDING CYLINDER AND TRACK ADDRESS.

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

'SELECT DATA PATTERN

(0) ZERO'S

(1) ONES

(2) WORST CASE:'

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

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

165555
133333

8. THE PROGRAM WILL THEN TYPE:

'STARTING FORMAT ON DRIVE N'

OR

'STARTING CHECK ON DRIVE N'

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

'PRESENT ADDRESS IS: CXXX TXX'

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

10. IF THE OPERATOR DOES NOT SELECT A DIFFERENT DRIVE WHEN THE FORMAT OR CHECK OPERATION HAS COMPLETED, THE PROGRAM WILL NOT ALTER THE ADDRESS LIMITS SPECIFIED AT THE START OF THE PREVIOUS OPERATION. IF THE FORMAT OR CHECK OPERATION IS TERMINATED BY A 'CONTROL C' OR IF A DIFFERENT DRIVE IS SELECTED, THE PROGRAM WILL RESET THE ADDRESS LIMITS TO THE VALUES APPROPRIATE FOR THE DRIVE TYPE.
11. TO CHANGE EITHER THE ADDRESSING MODE OR THE OPERATION MODE, THE PROGRAM MUST BE STARTED FROM LOCATION 200(8) OR 204(8) AGAIN.

4.3 RH11 - RH70 UNIBUS ADDRESS

THE PROGRAM ASSUMES THAT THE RH11 OR RH70 ADDRESSES START AT 176700 AND THAT THE VECTOR ADDRESS IS 254. THESE ADDRESSES MAY BE CHANGED WHEN THE PROGRAM IS STARTED FROM LOCATION 204(8). IF THE RH11 - RH70 IS NOT AT THE DEFAULT ADDRESS, THE PROGRAM MUST BE STARTED FROM 204(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.

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<07>=1...PRINT SOFT ERROR REPORTS AS THEY OCCUR
 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

306 RECOGNIZE THE 'CONTROL G' AT ANY TIME EXCEPT WHEN THE PROGRAM
307 IS AT A HIGHER PRIORITY PROCESSING AN RP04/5/6 INTERRUPT. THE
308 'SOFTWARE' SWITCH VALUES ARE ENTERED AS AN OCTAL NUMBER IN RESPONSE
309 TO THE PROMPT FROM THE SWITCH ENTRY ROUTINE:
310

311 'SWR = NNNNNN NEW ='
312

313 EACH TIME SWITCH SETTING ARE ENTERED, THE ENTIRE SWITCH REGISTER
314 IMAGE MUST BE ENTERED. LEADING ZEROS ARE NOT REQUIRED., 'RUBOUT' AND
315 'CONTROL U' FUNCTIONS MAY BE USED TO CORRECT TYPING ERRORS
316 DURING SWITCH ENTRY.
317

318 ON PROCESSORS WITH HARDWARE SWITCH REGISTERS, THE 'SOFTWARE' SWITCH
319 REGISTER MAY BE USED. IF THE PROGRAM FINDS ALL 16 SWITCHES IN THE
320 'UP' POSITION, ALL SWITCH REGISTER REFERENCES WILL BE TO THE
321 'SOFTWARE' REGISTER AND THE PROCEDURES DESCRIBED ABOVE MUST
322 BE FOLLOWED.
323

324
325
326
327 6. ERROR MESSAGES
328 -----
329

- 330 1. 'RH11 INTERRUPT OCCURRED (RPAS=0) - AN INTERRUPT OCCURRED, BUT
331 NOTHING ON THE MASSBUS IS INDICATING AN ATTENTION.
332
333 2. 'UNEXPECTED ATTENTION OCCURRED' - THE INDICATED DRIVE INTERRUPTED,
334 BUT NO INTERRUPT WAS EXPECTED FROM THE INDICATED DRIVE.
335
336 3. 'MASSBUS PARITY ERROR (MCPE=1)' - A CONTROL BUS PARITY ERROR
337 WAS DETECTED BY THE RH11 WHEN THE INDICATED REGISTER WAS READ.
338
339 4. 'MASSBUS PARITY ERROR (PAR=1)' - A CONTROL BUS PARITY ERROR
340 OCCURRED WHEN THE INDICATED REGISTER WAS WRITTEN.
341
342 5. 'ADDRESS PLUG CHANGE BIT SET' - THE PROGRAM FOUND THE 'OPE'
343 BIT SET FOR THE INDICATED DRIVE.
344
345 6. 'RH11 DIDN'T RESPOND TO ADDRESSING' - THE PROGRAM ACCESSED THE
346 RH11 AT THE INDICATED ADDRESS AND RECEIVED NO RESPONSE.
347
348 7. 'DRIVE OFFLINE' - THE INDICATED DRIVE HAS GONE OFFLINE
349
350 8. 'PERSISTENT DRIVE UNSAFE ERROR' - THE INDICATED DRIVE HAS BECOME
351 UNSAFE AND THE CONDITION CANNOT BE CLEARED BY ISSUING 'DRIVE
352 CLEAR' INSTRUCTIONS.
353
354 9. 'UNCORRECTABLE MASSBUS PARITY ERROR' - THE PROGRAM ATTEMPTED
355 TO PERFORM AN OPERATION AND DETECTED 3 SUCESSIVE MASSBUS PARITY
356 ERRORS OR THE PROGRAM ATTEMPTED TO CLEAR A 'PAR' ERROR IN THE
357 DRIVE AND A PARITY ERROR OCCURRED.
358
359 10. 'SOFTWARE TIMEOUT' - THE OPERATION FAILED TO COMPLETE WITHIN
360 1 SECOND.
361

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 A 'DCK' ERROR.
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 ERROR OCCURRED DURING THE VERIFICATION PASS AFTER FORMATTING.
18. 'CYLINDER FIELD IN HEADER IS NOT CORRECT' - THE CYLINDER FIELD FROM A HEADER READ DURING THE VERIFICATION PASS IS NOT CORRECT.
19. 'WRITE CHECK ERROR' - THE RH11 REPORTED A WRITE CHECK ERROR AND NO DRIVE ERRORS WERE SET.
20. 'HARDWARE ERROR DURING WRITE CHECK' - AN ERROR THAT WAS NEITHER A DATA ERROR, WRITE CHECK ERROR, NOR CONTROLLER ERROR OCCURRED DURING A WRITE CHECK. THESE ERRORS COULD BE ONE OF THE FOLLOWING: 'OPI', 'DTE', 'HCRC', 'HCE', OR 'FER'.

7. MISCELLANEOUS

7.1 FORMAT TIME

IT TAKES APPROXIMATELY 8 MINUTES TO FORMAT AN ENTIRE RP04/5 PACK AND APPROXIMATELY 16 MINUTES TO FORMAT AN RP06 PACK. THE 'CHECK' MODE TIME FOR AN ENTIRE PACK IS 4 MINUTES FOR RP04/5'S AND 8 MINUTES FOR RP06'S.

7.2 HALTING THE PROGRAM

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

7.3 SURFACE VERIFICATION

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

8. PROGRAM DESCRIPTION

8.1 FORMAT OPERATION

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

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

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

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

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

8.2 CHECK OPERATION

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

8.3 POSITIONER VERIFICATION

AFTER THE PROGRAM COMPLETES THE FORMAT OPERATION, THE

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

9. PROGRAM LISTING

```

@
.TITLE CZRJBDO, RP04/5/6 FMTR
;*COPYRIGHT (C) 1976,1978
;*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.
;*
```

.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
.=174
DISPREG: .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
```

```

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      000000
515
516
517
518      000174
519      000174 000000
520      000176 000000
521
522
523
524
525
526      000200
527      000046
528      000046 005422
529      000052
```



```

530 000052 020000      .WORD 20000      ;;2)SET LOC.52 TO 20000
531      000200      .= $SVPC      ;; RESTORE PC
532
533      .SBTTL STARTING ADDRESS = 200
534      . =200
535 000200 000137 002132  JMP BEGIN1      ;NORMAL STARTING ADDRESS (INHIBIT PROGRAMMABLE DRIVES
536      ;SEE SEC 2.3)
537
538      .SBTTL STARTING ADDRESS TO CHANGE THE RH11 ADDRESS = 204
539 000204 000137 002114  JMP BEGIN      ;CHANGE THE RH11 ADDRESS (NO INHIBITIONS)
540
541      . =220
542 000220 000137 002100  JMP BEGIN3      ;SAME AS 200 WITH NO INHIBITIONS
543
544      .SBTTL BASIC DEFINITIONS
545
546      ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
547      001100  STACK= 1100
548      .EQUIV EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
549      .EQUIV IOT,SCOPE      ;;BASIC DEFINITION OF SCOPE CALL
550
551      ;*MISCELLANEOUS DEFINITIONS
552      000011  HT= 11      ;;CODE FOR HORIZONTAL TAB
553      000012  LF= 12      ;;CODE FOR LINE FEED
554      000015  CR= 15      ;;CODE FOR CARRIAGE RETURN
555      000200  CRLF= 200      ;;CODE FOR CARRIAGE RETURN-LINE FEED
556      177776  PS= 177776      ;;PROCESSOR STATUS WORD
557      .EQUIV PS,PSW
558      177774  STKLM= 177774      ;;STACK LIMIT REGISTER
559      177772  PIRQ= 177772      ;;PROGRAM INTERRUPT REQUEST REGISTER
560      177570  DSWR= 177570      ;;HARDWARE SWITCH REGISTER
561      177570  DDISP= 177570      ;;HARDWARE DISPLAY REGISTER
562
563      ;*GENERAL PURPOSE REGISTER DEFINITIONS
564      000000  R0= %0      ;;GENERAL REGISTER
565      000001  R1= %1      ;;GENERAL REGISTER
566      000002  R2= %2      ;;GENERAL REGISTER
567      000003  R3= %3      ;;GENERAL REGISTER
568      000004  R4= %4      ;;GENERAL REGISTER
569      000005  R5= %5      ;;GENERAL REGISTER
570      000006  R6= %6      ;;GENERAL REGISTER
571      000007  R7= %7      ;;GENERAL REGISTER
572      000006  SP= %6      ;;STACK POINTER
573      000007  PC= %7      ;;PROGRAM COUNTER
574
575      ;*PRIORITY LEVEL DEFINITIONS
576      000000  PR0= 0      ;;PRIORITY LEVEL 0
577      000040  PR1= 40      ;;PRIORITY LEVEL 1
578      000100  PR2= 100      ;;PRIORITY LEVEL 2
579      000140  PR3= 140      ;;PRIORITY LEVEL 3
580      000200  PR4= 200      ;;PRIORITY LEVEL 4
581      000240  PR5= 240      ;;PRIORITY LEVEL 5
582      000300  PR6= 300      ;;PRIORITY LEVEL 6
583      000340  PR7= 340      ;;PRIORITY LEVEL 7
584
585      ;*'SWITCH REGISTER' SWITCH DEFINITIONS

```


586	100000	SW15=	100000
587	040000	SW14=	40000
588	020000	SW13=	20000
589	010000	SW12=	10000
590	004000	SW11=	4000
591	002000	SW10=	2000
592	001000	SW09=	1000
593	000400	SW08=	400
594	000200	SW07=	200
595	000100	SW06=	100
596	000040	SW05=	40
597	000020	SW04=	20
598	000010	SW03=	10
599	000004	SW02=	4
600	000002	SW01=	2
601	000001	SW00=	1

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

613 ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)

614	100000	BIT15=	100000
615	040000	BIT14=	40000
616	020000	BIT13=	20000
617	010000	BIT12=	10000
618	004000	BIT11=	4000
619	002000	BIT10=	2000
620	001000	BIT09=	1000
621	000400	BIT08=	400
622	000200	BIT07=	200
623	000100	BIT06=	100
624	000040	BIT05=	40
625	000020	BIT04=	20
626	000010	BIT03=	10
627	000004	BIT02=	4
628	000002	BIT01=	2
629	000001	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

641 ;*BASIC 'CPU' TRAP VECTOR ADDRESSES


```

642      000004      ERRVEC= 4      ::TIME OUT AND OTHER ERRORS
643      000010      RESVEC= 10     ::RESERVED AND ILLEGAL INSTRUCTIONS
644      000014      TBITVEC=14     ::'T' BIT
645      000014      TRTVEC= 14     ::TRACE TRAP
646      000014      BPTVEC= 14     ::BREAKPOINT TRAP (BPT)
647      000020      IOTVEC= 20     ::INPUT/OUTPUT TRAP (IOT) **SCOPE**
648      000024      PWRVEC= 24     ::POWER FAIL
649      000030      EMTVEC= 30     ::EMULATOR TRAP (EMT) **ERROR**
650      000034      TRAPVEC=34     ::'TRAP' TRAP
651      000060      TKVEC= 60      ::TTY KEYBOARD VECTOR
652      000064      TPVEC= 64      ::TTY PRINTER VECTOR
653      000240      PIRQVEC=240    ::PROGRAM INTERRUPT REQUEST VECTOR
654
655      ;:*****
656
657      .SBTTL RH11 REGISTERS
658
659      ;:*****
660
661      ;CONTROL AND STATUS REGISTER 1 (RPCS1)
662
663      000100      IE= 100          ;INTERRUPT ENABLE (BIT #6)
664      000200      RDY= 200        ;READY (BIT #7)
665      000400      A16= 400        ;HIGH ORDER BUS ADDRESS BIT (BIT #8)
666      001000      A17= 1000      ;HIGH ORDER BUS ADDRESS BIT (BIT #9)
667      002000      PSEL= 2000     ;PORT SELECT (BIT #10)
668      020000      MCPE= 20000    ;MASSBUS PARITY ERROR (BIT #13)
669      040000      TRE= 40000     ;TRANSFER ERROR (BIT #14)
670      ;SC= 100000              ;SPECIAL CONDITION (BIT #15)
671
672      ;WORD COUNT REGISTER (RPWC)
673      ;(EACH BIT IS CALLED BY BIT NUMBER)
674
675      ;BUS ADDRESS REGISTER (RPBA)
676      ;(EACH BIT IS CALLED BY BIT NUMBER)
677
678      ;CONTROL AND STATUS REGISTER 2 (RPCS2)
679
680      000001      US1= 1          ;UNIT SELECT (BIT #0)
681      000002      US2= 2          ;UNIT SELECT (BIT #1)
682      000004      US4= 4          ;UNIT SELECT (BIT #2)
683      000010      BAI= 10        ;BUS ADDRESS INCREMENT INHIBIT (BIT #3)
684      000020      PAT= 20        ;MASSBUS PARITY TEST (BIT #4)
685      000040      CLR= 40        ;CLEAR (BIT #5)
686      000100      IR= 100        ;INPUT READY (BIT #6)
687      000200      OR= 200        ;OUTPUT READY (BIT #7)
688      000400      MPE= 400       ;MASS BUS PARITY ERROR (BIT #8)
689      001000      MXF= 1000      ;MISSED TRANSFER ERROR (BIT #9)
690      002000      PGE= 2000     ;PROGRAM ERROR (BIT #10)
691      004000      NEM= 4000      ;NON EXISTENT MEMORY (BIT #11)
692      010000      NED= 10000    ;NON EXISTENT DRIVE (BIT #12)
693      020000      UPE= 20000    ;UNIBUS PARITY ERROR (BIT #13)
694      040000      WCE= 40000    ;WRITE CHECK ERROR (BIT #14)
695      100000      DLT= 100000   ;DATA LATE (BIT #15)
696
697      ;DATA BUFFER REGISTER (RPDB)

```



```

698      ;(EACH BIT IS CALLED BY BIT NUMBER)
699
700
701      ;*****
702
703      .SBTTL  RP04/5/6 REGISTERS
704
705      ;*****
706
707      ;CONTROL AND STATUS 1 REGISTER. (#00)
708
709      000001      GO=      1      ;GO BIT (BIT #0)
710      000002      F1=      2      ;FUNCTION CODE BIT #1
711      000004      F2=      4      ;FUNCTION CODE BIT #2
712      000010      F3=     10      ;FUNCTION CODE BIT #3
713      000020      F4=     20      ;FUNCTION CODE BIT #4
714      000040      F5=     40      ;FUNCTION CODE BIT #5
715      004000      DVA=    4000    ;DEVICE AVAILABLE (BIT #11)
716
717      ;DRIVE STATUS REGISTER (RPDS1) (#01)
718
719      ;DF5=      1      DRIVE FORWARD 5'/SEC. (BIT #0)
720      000002      DFF20= 2      ;DRIVE FORWARD 20'/SEC. (BIT #1)
721      000004      DIGB= 4      ;DRIVE TO INNER GUARD BAND (BIT #2)
722      000010      GRV= 10      ;GO REVERSE (BIT #3)
723      000020      DL64= 20      ;DIFFERENCE LESS THAN 64 (BIT #4)
724      000040      DE1= 40      ;DIFFERENCE EQUALS 1 (BIT #5)
725      000100      VV= 100      ;VOLUME VALID (BIT #6)
726      000200      DRY= 200      ;DRIVE READY (BIT #7)
727      000400      DPR= 400      ;DRIVE PRESENT (BIT #8)
728      001000      PGM= 1000     ;PROGRAMABLE (BIT #9)
729      002000      LST= 2000     ;LAST SECTOR TRANSFERRED (BIT #10)
730      004000      WRL= 4000     ;WRITE LOCK (BIT #11)
731      010000      MOL= 10000    ;MEDIUM ON-LINE (BIT #12)
732      020000      PIP= 20000    ;POSITIONING OPERATION IN PROGRESS (BIT #13)
733      040000      ERR= 40000    ;COMPOSITE ERROR (BIT #14)
734      100000      ATA= 100000   ;ATTENTION ACTIVE (BIT #15)
735
736      ;ERROR REGISTER #01 (RPER1) (#02)
737
738      000001      ILF=      1      ;ILLEGAL FUNCTION (BIT #0)
739      000002      ILR=      2      ;ILLEGAL REGISTER (BIT #1)
740      000004      RMR=      4      ;REGISTER MODIFICATION REFUSED (BIT #2)
741      000010      PAR=     10      ;PARITY ERROR (BIT #3)
742      000020      FER=     20      ;FORMAT ERROR (BIT #4)
743      000040      WCF=     40      ;WRITE CLOCK FAIL (BIT #5)
744      000100      ECH=    100      ;ECC HARD ERROR (BIT #6)
745      000200      HCE=    200      ;HEADER COMPARE ERROR (BIT #7)
746      000400      HCRC=   400      ;HEADER CRC ERROR (BIT #8)
747      001000      AOE=   1000     ;ADDRESS OVERFLOW ERROR (BIT #9)
748      002000      IAE=   2000     ;INVALID ADDRESS ERROR (BIT #10)
749      004000      WLE=   4000     ;WRITE LOCK ERROR (BIT #11)
750      010000      DTE=  10000     ;DRIVE TIMING ERROR (BIT #12)
751      020000      OPI=  20000     ;OPERATION INCOMPLETE (BIT #13)
752      040000      UNS=  40000     ;DRIVE UNSAFE (BIT #14)
753      100000      DCK= 100000    ;DATA CHECK ERROR (BIT 15)
  
```


754			
755			
756		;MAINTAINABILITY REGISTER (RPMR) (#03)	
757	000001	DMD= 1	;DIAGINOSTIC MODE (BIT #0)
758	000002	MCLK= 2	;MAINTAINABILITY CLOCK (BIT #1)
759	000004	MINX= 4	;MAINTAINABILITY INDEX (BIT #2)
760	000010	MSTCK= 10	;MAINTAINABILITY SECTOR CLOCK (BIT #3)
761	000020	MRD= 20	;MAINTAINABILITY READ (BIT #4)
762	000040	MWR= 40	;MAINTAINABILITY WRITE (BIT #5)
763	000200	DTSY= 200	;MAINTAINABILITY SYNC DETECTED (BIT #7)
764			
765		;ATTENTION SUMMARY PSEUDO-REGISTER (RPAS) (#04)	
766			
767	000001	AT0= 1	;DEVICE 0 (BIT #0)
768	000002	AT1= 2	;DEVICE 1 (BIT #1)
769	000004	AT2= 4	;DEVICE 2 (BIT #2)
770	000010	AT3= 10	;DEVICE 3 (BIT #3)
771	000020	AT4= 20	;DEVICE 4 (BIT #4)
772	000040	AT5= 40	;DEVICE 5 (BIT #5)
773	000100	AT6= 100	;DEVICE 6 (BIT #6)
774	000200	AT7= 200	;DEVICE 7 (BIT #7)
775			
776		;DESIRED SECTOR/TRACK ADDRESS REGISTER (RPDA) (#05)	
777		;(EACH BIT IS CALLED BY BIT NUMBER)	
778			
779		;DRIVE TYPE REGISTER (RPDT) (#06)	
780			
781	000001	DT00= 1	;DRIVE TYPE NUMBER BIT 1
782	000002	DT01= 2	;DRIVE TYPE NUMBER BIT 2
783	000004	DT02= 4	;DRIVE TYPE NUMBER BIT 3
784	000010	DT03= 10	;DRIVE TYPE NUMBER BIT 4
785	000020	DT04= 20	;DRIVE TYPE NUMBER BIT 5
786	000040	DT05= 40	;DRIVE TYPE NUMBER BIT 6
787	000100	DT06= 100	;DRIVE TYPE NUMBER BIT 7
788	000200	DT07= 200	;DRIVE TYPE NUMBER BIT 8
789	000400	DT08= 400	;DRIVE TYPE NUMBER BIT 9
790	004000	DRQ= 4000	;DRIVE REQUEST REQUIRED (BIT #11)
791	020000	MOH= 20000	;MOVING HEAD (BIT #13)
792	040000	TAP= 40000	;TAPE DRIVE (BIT #14)
793	100000	NBA= 100000	;NOT BLOCK ADDRESSED (BIT #15)
794			
795		;LOOK-AHEAD REGISTER (RPLA) (#07)	
796			
797	000001	EXT1= 1	;EXTENSION 1 (BIT #0)
798	000002	EXT2= 2	;EXTENSION 2 (BIT #1)
799	000004	EXT4= 4	;EXTENSION 3 (BIT #2)
800	000010	EXT10= 10	;EXTENSION 4 (BIT #3)
801	000020	EXT20= 20	;EXTENSION 5 (BIT #4)
802	000040	EXT40= 40	;EXTENSION 6 (BIT #5)
803	000100	SC1= 100	;SECTOR COUNT FIELD 0 (BIT #6)
804	000200	SC2= 200	;SECTOR COUNT FIELD 1 (BIT #7)
805		SC4= 400	;SECTOR COUNT FIELD 2 (BIT #8)
806	001000	SC10= 1000	;SECTOR COUNT FIELD 3 (BIT #9)
807	002000	SC20= 2000	;SECTOR COUNT FIELD 4 (BIT #10)
808	004000	TRK1= 4000	;TRACK FIELD 1 (BIT #11)
809	010000	TRK2= 10000	;TRACK FIELD 2 (BIT #12)

810	020000	TRK4=	20000	;TRACK FIELD 3 (BIT #13)
811	040000	TRK10=	40000	;TRACK FIELD 4 (BIT #14)
812	100000	TRK20=	100000	;TRACK FIELD 5 (BIT #15)
813				
814		;RP04 ERROR REGISTER #2 (RPER2) (#10)		
815				
816	000001	WCU=	1	;WRITE CURRENT UNSAFE (BIT #0)
817	000002	CSF=	2	;CURRENT SINK FAILURE (BIT #1)
818	000004	WSU=	4	;WRITE SELECT UNSAFE (BIT #2)
819	000010	CSU=	10	;CURRENT SWITCH UNSAFE (BIT #3)
820	000020	MSE=	20	;MOTOR SEQUENCE ERROR (BIT #4)
821	000040	TDF=	40	;TRANSITIONS DETECTOR FAILURE (BIT #5)
822	000100	TUF=	100	;TRANSITIONS UNSAFE (BIT #6)
823	000200	FEN=	200	;FAILSAFE ENABLED (BIT #7)
824	000400	WRU=	400	;WRITE READY UNSAFE (BIT #8)
825	001000	MHS=	1000	;MULTIPLE HEAD SELECT (BIT #9)
826	002000	NHS=	2000	;NO HEAD SELECTION (BIT #10)
827	004000	IXE=	4000	;INDEX ERROR (BIT #11)
828	010000	VU30=	10000	;30VOLT UNSAFE (BIT #12)
829	020000	PLU=	20000	;PLO UNSAFE (BIT #13)
830	100000	ACU=	100000	;AC UNSAFE (BIT #15)
831				
832		;RP05/6 ERROR REGISTER #02 (RPER2) (#10)		
833				
834	000001	WCU=	1	;WRITE CURRENT UNSAFE (BIT #0)
835	000002	CSF=	2	;CURRENT SINK FAILURE (BIT #1)
836	000004	WSU=	4	;WRITE SELECT UNSAFE (BIT #2)
837	000010	CSU=	10	;CURRENT SWITCH UNSAFE (BIT #3)
838	000020	RAW=	20	;READ AND WRITE (BIT #4)
839	000040	TDF=	40	;TRANSITIONS DETECTOR FAILURE (BIT #5)
840	000100	TUF=	100	;TRANSITIONS UNSAFE (BIT #6)
841	000200	ABS=	200	;ABNORMAL STOP (BIT #7)
842	000400	WRU=	400	;WRITE READY UNSAFE (BIT #8)
843	001000	MHS=	1000	;MULTIPLE HEAD SELECT (BIT #9)
844	002000	NHS=	2000	;NO HEAD SELECTION (BIT #10)
845	004000	IXE=	4000	;INDEX ERROR (BIT #11)
846	020000	PLU=	20000	;PLO UNSAFE (BIT #12)
847				
848		;OFFSET REGISTER (RPOF) (#11)		
849				
850	000001	OF25=	1	;OFFSET 25 MICRO INCHES (BIT #0)
851	000002	OF50=	2	;OFFSET 50 MICRO INCHES (BIT #1)
852	000004	OF100=	4	;OFFSET 100 MICRO INCHES (BIT #2)
853	000010	OF200=	10	;OFFSET 200 MICRO INCHES (BIT #3)
854	000020	OF400=	20	;OFFSET 400 MICRO INCHES (BIT #4)
855	000040	OF800=	40	;OFFSET 800 MICRO INCHES (BIT #5)
856	000200	OFREV=	200	;OFFSET NEGATIVE (REVERSE) (BIT #5)
857	002000	HCI=	2000	;HEADER COMPARE INHIBIT (BIT #10)
858	004000	ECI=	4000	;ERROR CORRECTION CODE INHIBIT (BIT #11)
859	010000	FMT22=	10000	;FORMAT BIT (BIT #12)
860				
861		;DESIRED CYLINDER ADDRESS (RPCA) (#12)		
862		;(EACH BIT IS CALLED BY BIT NUMBER)		
863				
864		;CURRENT CYLINDER ADDRESS (RPCC) (#13)		
865		;(EACH BIT IS CALLED BY BIT NUMBER)		


```

866
867 ;SERIAL NUMBER REGISTER (RPSN) (#14)
868 ;(EACH IS CALLED BY BIT NUMBER)
869
870 ;RP04 ERROR REGISTER #03 (RPER3) (#15)
871
872 000001 PSU= 1 ;PACK SPEED UNSAFE (BIT #0)
873 000002 VUF= 2 ;VELOCITY UNSAFE (BIT #1)
874 000010 UWR= 10 ;ANY UNSAFE EXCEPT READ/WRITE (BIT #3)
875 000040 ACL= 40 ;AC LOW (BIT #5)
876 000100 DCL= 100 ;DC LOW (BIT #6)
877 040000 SKI= 40000 ;SEEK INCOMPLETE (BIT #14)
878 100000 OCYL= 100000 ;OFF CYLINDER (BIT #15)
879
880 ;RP05/6 ERROR REGISTER #03 (RPER3) (#15)
881
882 000001 DCU= 1 ;DC UNSAFE (BIT #0)
883 000002 WAO= 2 ;WRITE AND OFFSET (BIT #1)
884 000040 ACL= 40 ;AC LOW (BIT #5)
885 000100 DCL= 100 ;DC LOW (BIT #6)
886 020000 OPE= 20000 ;OPERATOR PLUG ERROR (BIT #13)
887 040000 SKI= 40000 ;SEEK INCOMPLETE (BIT #14)
888 100000 OCYL= 100000 ;OFF CYLINDER ERROR (BIT #15)
889
890 ;ECC POSITION REGISTER (RPEC1) (#16)
891 ;(EACH BIT IS CALLED BY BIT NUMBER)
892
893 ;ECC PATTERN REGISTER (RPEC2) (#17)
894 ;(EACH BIT IS CALLED BY BIT NUMBER)
895
896 ;:*****
897
898 .SBTTL RP04/5/6 DRIVER COMMANDS
899
900 ;:*****
901
902 000101 RNOP = 101 ;NO OPERATION
903 000103 UNLOAD = 103 ;UNLOAD
904 000105 SEEK = 105 ;SEEK
905 000107 RECAL = 107 ;RECALIBRATE
906 000111 DRVCLR = 111 ;DRIVE CLEAR
907 000113 RELSE = 113 ;RELEASE
908 000115 OFFSET = 115 ;OFFSET
909 000117 RTC = 117 ;RETURN TO CENTER LINE
910 000121 READIN = 121 ;READ IN PRESET
911 000123 ACK = 123 ;PACK ACKNOWLEDGE
912 000131 SEARCH = 131 ;SEARCH
913 000141 GETREG = 141 ;GET REGISTERS
914 000143 SETFMT = 143 ;SET FORMAT (& ECI OR HCI)
915 000145 SELDRV = 145 ;SELECT DRIVE
916 000151 WCKD = 151 ;WRITE CHECK DATA
917 000153 WCKHD = 153 ;WRITE CHECK HEADER & DATA
918 000161 WRTDAT = 161 ;WRITE DATA
919 000163 WRTHD = 163 ;WRITE HEADER & DATA
920 000171 RDDAT = 171 ;READ DATA
921 000173 RDHD = 173 ;READ HEADER & DATA
  
```


CZRJBDO, RP04/5/6 FMTR MACY11 30A(1052) 24-MAY-79 14:10 PAGE 20^F 2
CZRJBD.P11 28-MAR-79 11:23 RP04/5/6 DRIVER COMMANDS

SEQ 0018

922
923


```

924      .SBTTL  COMMON TAGS
925
926      ;*****
927      ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
928      ;*USED IN THE PROGRAM.
929
930      001100      .=1100
931      001100      $CMTAG:
932      001100      $PASS: .WORD 0
933      001102      $TSTNM: .BYTE 0
934      001103      $ERFLG: .BYTE 0
935      001104      $ICNT: .WORD 0
936      001106      $LPADR: .WORD 0
937      001110      $LPERR: .WORD 0
938      001112      $ERTTL: .WORD 0
939      001114      $ITEMB: .BYTE 0
940      001115      $ERMAX: .BYTE 1
941      001116      $ERRPC: .WORD 0
942      001120      $GDADR: .WORD 0
943      001122      $BDADR: .WORD 0
944      001124      $GDDAT: .WORD 0
945      001126      $BDDAT: .WORD 0
946      001130      .WORD 0
947      001132      .WORD 0
948      001134      $AUTOB: .BYTE 0
949      001135      $INTAG: .BYTE 0
950      001136      .WORD 0
951      001140      SWR: .WORD DSWR
952      001142      DISPLAY: .WORD DDISP
953      001144      $TKS: 177560
954      001146      $TKB: 177562
955      001150      $TPS: 177564
956      001152      $TPB: 177566
957      001154      $NULL: .BYTE 0
958      001155      $FILLS: .BYTE 2
959      001156      $FILLC: .BYTE 12
960      001157      $TPFLG: .BYTE 0
961      001160      $ESCAPE: 0
962      001162      $BELL: .ASCIIZ <207><377><377>
963      001166      $QUES: .ASCII /?/
964      001167      $CRLF: .ASCII <15>
965      001170      $LF: .ASCIIZ <12>
966
967      000015      CR = 15
968      000012      LF = 12
969      001172      $RPADR: .WORD 176700
970      001174      $RPVEC: .WORD 254
971      001176      $LKCSR: .WORD 172540
972      001200      $LKCSB: .WORD 172542
973      001202      $LPVEC: .WORD 104,106
974      001206      $LKS: .WORD 177546
975      001210      $LLVEC: .WORD 100,102
976      001214      DRIVE: .WORD 0
977      001216      SOFSW: .WORD 0
978      001220      MODE: .WORD 0
979      001222      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/RP04/5/6 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

```


980	001224	000000	BEGCYL: .WORD	0	:STARTING CYLINDER
981	001226	000022	ENDTRK: .WORD	18.	:ENDING TRACK
982	001230	000000	BEGTRK: .WORD	0	:STARTING TRACK
983	001232	000000	TTRKS: .WORD	0	:TOTAL # OF TRACKS TO BE FORMATTED
984	001234	000000	TTRKSC: .WORD	0	:TOTAL # OF TRACKS COUNTER
985	001236	000000	TRKCNT: .WORD	0	:COUNTS TRKS FROM 0-18 PER CYL
986	001240	000000	PATSEL: .WORD	0	:CONTAINS PATTERN SELECTED
987	001242	000000	PATA: .WORD	0	:1ST WORD OF PATTERN
988	001244	000000	PATB: .WORD	0	:2ND WORD OF PATTERN
989	001246	000000	CYLCK: .WORD	0	:STORE CYLINDER ADDRESS FOR FORMAT VERIFICATION
990	001250	000000	RETRY: .WORD	0	:MAINTAINS # OF WRITE RETRIES MADE
991	001252	000000	SAVSEC: .WORD	0	:CONTAINS LAST BAD SECTOR ON FORMAT
992	001254	000000	SAVWC: .WORD	0	:CONTAINS WC FOR REMAINING SECTORS ON ERROR
993	001256	000000	WC: .WORD	0	:20 OR 22 SECTOR TRACK SIZE (IN WORDS)
994	001260	000000	MWC: .WORD	0	:2'S COMPLEMENT OF 'WC'
995	001262	000000	HEDERR: .WORD	0	:POSITIONING ERROR DURING FORMAT INDICATOR
996	001264	000000	SEC20: .WORD	0	:20 OR 22 SECTOR MODE INDICATOR
997					:0 = 20 SECTOR MODE
998					:1'S = 22 SECTOR MODE
999	001266	000000	MAXSEC: .WORD	0	:MAXIMUM SECTOR ADDRESS (FOR EITHER 20 OR 22 SECTOR
1000					:FORMAT)
1001	001270	000000	DS.CYL: .WORD	0	:ADDRESS OF CURRENT CYLINDER
1002	001272	000000	DS.TRK: .WORD	0	:ADDRESS OF CURRENT TRACK
1003	001274	000000	DDRIVE: .WORD	0	:DRIVE NUMBER OF 'DRIVER' ERROR MESSAGES
1004	001276	000000	ATTN: .WORD	0	:ATTENTION REGISTER IMAGE FOR 'DRIVER'
1005					:ERROR MESSAGES
1006	001300	000000	CNTLC: .WORD	0	:ADDRESS OF '^C' RETURN
1007	001302	000000	CHGADR: .WORD	0	: 'CHANGE RH11 ADDRESS' FLAG
1008	001304	000000	EFLG: .WORD	0	: 'ERROR DURING WRITE CHECK' FLAG
1009					
1010					:RH11/RP04/5/6 REGISTERS STORED HERE AFTER AN OPERATION
1011					
1012	001306	000000	RP.REG: .WORD	0	:RPCS1
1013	001310	000000	.WORD	0	:RPWC
1014	001312	000000	.WORD	0	:RPBA
1015	001314	000000	.WORD	0	:RPDA
1016	001316	000000	.WORD	0	:RPCS2
1017	001320	000000	.WORD	0	:RPDS1
1018	001322	000000	.WORD	0	:RPER1
1019	001324	000000	.WORD	0	:RPAS
1020	001326	000000	.WORD	0	:RPLA
1021	001330	000000	.WORD	0	:RPDB
1022	001332	000000	.WORD	0	:RPMR
1023	001334	000000	.WORD	0	:RPDT
1024	001336	000000	.WORD	0	:RPSN
1025	001340	000000	.WORD	0	:RPOF
1026	001342	000000	.WORD	0	:RPCA
1027	001344	000000	.WORD	0	:RPCC
1028	001346	000000	.WORD	0	:RPER2
1029	001350	000000	.WORD	0	:RPER3
1030	001352	000000	.WORD	0	:RPEC1
1031	001354	000000	.WORD	0	:RPEC2
1032					
1033					
1034					:DATA/PARAMETER BLOCK - USED FOR ALL DRIVE OPERATION
1035					

1036	001356	000	FMTDPB: .BYTE	0	:DRIVE NUMBER
1037	001357	000	.BYTE	0	:OFFSET VALUE OR FMT22,ECI, AND HCI
1038	001360	000	.BYTE	0	:COMMAND
1039	001361	000	.BYTE	0	:PSEL AND A17 AND A16
1040	001362	000000	.WORD	0	:WORD COUNT (NEG)
1041	001364	000000	.WORD	0	:BUFFER ADDRESS
1042	001366	000	.BYTE	0	:SECTOR ADDRESS
1043	001367	000	.BYTE	0	:TRACK ADDRESS
1044	001370	000000	.WORD	0	:CYLINDER ADDRESS
1045	001372	001306	.WORD	RP.REG	:ERROR TABLE POINTER
1046	001374	000000	.WORD	0	:STATUS-ERROR INDICATOR
1047					:BIT 15 = 1: ERROR OCCURRED
1048					:BIT 07 = 1: DONE
1049					:BIT 14-10 AND BIT 06-03
1050					:INDICATE TYPE OF ERROR

:PARAMETER POINTER TABLE

1055	001376	001430	000000	001224	TABLE: PAR1,0,BEGCYL
1056	001404	001443	000022	001230	PAR2,18.,BEGTRK
1057	001412	001456	000000	001222	PAR3,0,ENDCYL
1058	001420	001467	000022	001226	PAR4,18.,ENDTRK,0
1059	001426	000000			

:ASCII MESSAGES FOR ADDRESS PARAMETERS

1063	001430	052123	051101	020124	PAR1: .ASCIIZ @START CYL @
1064	001436	054503	020114	000	
1065	001443	123	040524	052122	PAR2: .ASCIIZ @START TRK @
1066	001450	052040	045522	000040	
1067	001456	047105	020104	054503	PAR3: .ASCIIZ @END CYL @
1068	001464	020114	000		
1069	001467	105	042116	052040	PAR4: .ASCIIZ @END TRK @
1070	001474	045522	000040		

:SECTOR BUFFER ADDRESS TABLE

1074	001500	025574	ADRTBL: .WORD	BUFP	:ADDRESS OF SECTOR 0
1075	001502	026604	.WORD	BUFP+<520.*1.>	:ADDRESS OF SECTOR 1
1076	001504	027614	.WORD	BUFP+<520.*2.>	:ADDRESS OF SECTOR 2
1077	001506	030624	.WORD	BUFP+<520.*3.>	:ADDRESS OF SECTOR 3
1078	001510	031634	.WORD	BUFP+<520.*4.>	:ADDRESS OF SECTOR 4
1079	001512	032644	.WORD	BUFP+<520.*5.>	:ADDRESS OF SECTOR 5
1080	001514	033654	.WORD	BUFP+<520.*6.>	:ADDRESS OF SECTOR 6
1081	001516	034664	.WORD	BUFP+<520.*7.>	:ADDRESS OF SECTOR 7
1082	001520	035674	.WORD	BUFP+<520.*8.>	:ADDRESS OF SECTOR 8
1083	001522	036704	.WORD	BUFP+<520.*9.>	:ADDRESS OF SECTOR 9
1084	001524	037714	.WORD	BUFP+<520.*10.>	:ADDRESS OF SECTOR 10
1085	001526	040724	.WORD	BUFP+<520.*11.>	:ADDRESS OF SECTOR 11
1086	001530	041734	.WORD	BUFP+<520.*12.>	:ADDRESS OF SECTOR 12
1087	001532	042744	.WORD	BUFP+<520.*13.>	:ADDRESS OF SECTOR 13
1088	001534	043754	.WORD	BUFP+<520.*14.>	:ADDRESS OF SECTOR 14
1089	001536	044764	.WORD	BUFP+<520.*15.>	:ADDRESS OF SECTOR 15
1090	001540	045774	.WORD	BUFP+<520.*16.>	:ADDRESS OF SECTOR 16
1091	001542	047004	.WORD	BUFP+<520.*17.>	:ADDRESS OF SECTOR 17

1092	001544	050014	.WORD	BUFP+<520.*18.>	;ADDRESS OF SECTOR 18
1093	001546	051024	.WORD	BUFP+<520.*19.>	;ADDRESS OF SECTOR 19
1094	001550	052034	.WORD	BUFP+<520.*20.>	;ADDRESS OF SECTOR 20
1095	001552	053044	.WORD	BUFP+<520.*21.>	;ADDRESS OF SECTOR 21

1096
1097
1098

;REMAINING WORD COUNT TABLE

1099	001554	000404	WCTBL: .WORD	260.	;REMAINING WORD COUNT AFTER SECTOR 0
1100	001556	001010	.WORD	260.+<260.*1.>	;REMAINING WORD COUNT AFTER SECTOR 1
1101	001560	001414	.WORD	260.+<260.*2.>	;REMAINING WORD COUNT AFTER SECTOR 2
1102	001562	002020	.WORD	260.+<260.*3.>	;REMAINING WORD COUNT AFTER SECTOR 3
1103	001564	002424	.WORD	260.+<260.*4.>	;REMAINING WORD COUNT AFTER SECTOR 4
1104	001566	003030	.WORD	260.+<260.*5.>	;REMAINING WORD COUNT AFTER SECTOR 5
1105	001570	003434	.WORD	260.+<260.*6.>	;REMAINING WORD COUNT AFTER SECTOR 6
1106	001572	004040	.WORD	260.+<260.*7.>	;REMAINING WORD COUNT AFTER SECTOR 7
1107	001574	004444	.WORD	260.+<260.*8.>	;REMAINING WORD COUNT AFTER SECTOR 8
1108	001576	005050	.WORD	260.+<260.*9.>	;REMAINING WORD COUNT AFTER SECTOR 9
1109	001600	005454	.WORD	260.+<260.*10.>	;REMAINING WORD COUNT AFTER SECTOR 10
1110	001602	006060	.WORD	260.+<260.*11.>	;REMAINING WORD COUNT AFTER SECTOR 11
1111	001604	006464	.WORD	260.+<260.*12.>	;REMAINING WORD COUNT AFTER SECTOR 12
1112	001606	007070	.WORD	260.+<260.*13.>	;REMAINING WORD COUNT AFTER SECTOR 13
1113	001610	007474	.WORD	260.+<260.*14.>	;REMAINING WORD COUNT AFTER SECTOR 14
1114	001612	010100	.WORD	260.+<260.*15.>	;REMAINING WORD COUNT AFTER SECTOR 15
1115	001614	010504	.WORD	260.+<260.*16.>	;REMAINING WORD COUNT AFTER SECTOR 16
1116	001616	011110	.WORD	260.+<260.*17.>	;REMAINING WORD COUNT AFTER SECTOR 17
1117	001620	011514	.WORD	260.+<260.*18.>	;REMAINING WORD COUNT AFTER SECTOR 18
1118	001622	012120	.WORD	260.+<260.*19.>	;REMAINING WORD COUNT AFTER SECTOR 19
1119	001624	012524	.WORD	260.+<260.*20.>	;REMAINING WORD COUNT AFTER SECTOR 20
1120	001626	013130	.WORD	260.+<260.*21.>	;REMAINING WORD COUNT AFTER SECTOR 21

1121
1122
1123

.EVEN


```

1124 .SBTTL ERROR POINTER TABLE
1125
1126 ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
1127 ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
1128 ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
1129 ;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
1130 ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
1131
1132 ;*      EM      ;;POINTS TO THE ERROR MESSAGE
1133 ;*      DH      ;;POINTS TO THE DATA HEADER
1134 ;*      DT      ;;POINTS TO THE DATA
1135 ;*      DF      ;;POINTS TO THE DATA FORMAT
1136
1137
1138 001630 $ERRTB:
1139 ;ERROR 1
1140
1141 001630 022512 EM1 ;RH11 INTERRUPT OCCURRED (RPAS=0)
1142 001632 023674 DH1
1143 001634 025146 DT1
1144 001636 025460 DF1
1145
1146 ;ERROR 2
1147
1148 001640 022553 EM2 ;UNEXPECTED ATTENTION OCCURRED
1149 001642 023701 DH2
1150 001644 025150 DT2
1151 001646 025464 DF2
1152
1153 ;ERROR 3
1154
1155 001650 022611 EM3 ;MASSBUS PARITY ERROR (MCPE=1)
1156 001652 023756 DH3
1157 001654 025164 DT3
1158 001656 025470 DF3
1159
1160 ;ERROR 4
1161
1162 001660 022647 EM4 ;MASSBUS PARITY ERROR (PAR=1)
1163 001662 024004 DH4
1164 001664 025172 DT4
1165 001666 025474 DF4
1166
1167 ;ERROR 5
1168
1169 001670 022704 EM5 ;ADDRESS PLUG CHANGE BIT SET
1170 001672 023701 DH2
1171 001674 025150 DT2
1172 001676 025464 DF2
1173
1174 ;ERROR 6
1175
1176 001700 022740 EM6 ;RH11 DIDN'T RESPOND TO ADDRESSING
1177 001702 024043 DH6
1178 001704 025202 DT6
1179 001706 025460 DF1
  
```


1180				
1181				
1182			;ERROR 7	
1183	001710	000000	0	;UNUSED
1184	001712	000000	0	
1185	001714	000000	0	
1186	001716	000000	0	
1187				
1188			;ERROR 10	
1189				
1190	001720	023002	EM10	;DRIVE OFFLINE
1191	001722	024056	DH10	
1192	001724	025204	DT10	
1193	001726	025500	DF10	
1194				
1195			;ERROR 11	
1196				
1197	001730	023020	EM11	;PERSISTENT DRIVE UNSAFE ERROR
1198	001732	024056	DH10	
1199	001734	025204	DT10	
1200	001736	025500	DF10	
1201				
1202			;ERROR 12	
1203				
1204	001740	023056	EM12	;UNCORRECTABLE MASSBUS PARITY ERROR
1205	001742	024056	DH10	
1206	001744	025204	DT10	
1207	001746	025500	DF10	
1208				
1209			;ERROR 13	
1210				
1211	001750	023121	EM13	;SOFTWARE TIMEOUT
1212	001752	024056	DH10	
1213	001754	025204	DT10	
1214	001756	025500	DF10	
1215				
1216			;ERROR 14	
1217				
1218	001760	023142	EM14	;DRIVE UNSAFE ERROR
1219	001762	024056	DH10	
1220	001764	025204	DT10	
1221	001766	025500	DF10	
1222				
1223			;ERROR 15	
1224				
1225	001770	023165	EM15	;CONTROLLER/DRIVE ERROR DURING WRITE
1226	001772	024056	DH10	
1227	001774	025204	DT10	
1228	001776	025500	DF10	
1229				
1230			;ERROR 16	
1231				
1232	002000	023231	EM16	;CONTROLLER/DRIVE ERROR DURING WRITE CHECK
1233	002002	024056	DH10	
1234	002004	025204	DT10	
1235	002006	025500	DF10	

1236				
1237				
1238				
1239	002010	023303	EM17	
1240	002012	024326	DH17	
1241	002014	025260	DT17	
1242	002016	025514	DF17	
1243				
1244				
1245				
1246	002020	023361	EM20	
1247	002022	024375	DH20	
1248	002024	025272	DT20	
1249	002026	025520	DF20	
1250				
1251				
1252				
1253	002030	023417	EM21	
1254	002032	024056	DH10	
1255	002034	025204	DT10	
1256	002036	025500	DF10	
1257				
1258				
1259				
1260	002040	023470	EM22	
1261	002042	024056	DH10	
1262	002044	025204	DT10	
1263	002046	025500	DF10	
1264				

;ERROR 17

;RETRIES NOT SUCCESSFUL - SECTOR NOT ACCEPTABLE

;ERROR 20

;DATA ERROR DURING WRITE CHECK

;ERROR 21

;CONTROLLER/DRIVE ERROR VERIFYING HEADERS

;ERROR 22

;'HCE' ERROR VERIFYING HEADERS

1265			:ERROR 23	
1266				
1267	002050	023537	EM23	:CYLINDER FIELD IN HEADER IS NOT CORRECT
1268	002052	024674	DH23	
1269	002054	025354	DT23	
1270	002056	025540	DF23	
1271				
1272			:ERROR 24	
1273				
1274	002060	023607	EM24	:WRITE CHECK ERROR
1275	002062	024772	DH24	
1276	002064	025372	DT24	
1277	002066	025544	DF24	
1278				
1279			:ERROR 25	
1280				
1281	002070	023631	EM25	:HARDWARE ERROR DURING WRITE CHECK
1282	002072	024375	DH20	
1283	002074	025272	DT20	
1284	002076	025520	DF20	
1285				
1286			:*****	
1287				
1288			.SBTTL MAIN PROGRAM	


```
1289
1290
1291      ;;*****
1292 002100 005037 001302      BEGIN3: CLR      CHGADR      ;CLEAR THE INDICATOR
1293 002104 012737 000001 013310      MOV      #1,      TSTPGM ;ENABLE PROGRAMMABLE DRIVES
1294 002112 000413          BR      BEGIN2
1295 002114 012737 177777 001302      BEGIN: MOV      #-1,CHGADR ;SET CHANGE 'RH11 BUS ADDRESS' INDICATOR
1296 002122 012737 000001 013310      MOV      #1,      TSTPGM ;ENABLE PROGRAMMABLE DRIVES
```


1297	002130	000404			BR	BEGIN2	:START THE PROGRAM
1298	002132	005037	001302		BEGIN1: CLR	CHGADR	:CLEAR THE 'CHANGE RH11 ADDRESS' INDICATOR
1299	002136	005037	013310		CLR	TSTPGM	:DISABLE PROGRAMMABLE DRIVES
1300	002142	000005			BEGIN2: RESET		:CLEAR THE BUS
1301	002144	005037	001304		CLR	EFLG	:CLEAR ERROR DURING WRITE CHECK FLAG
1302					.SBTTL	INITIALIZE THE COMMON TAGS	
1303					::CLEAR	THE COMMON TAGS (\$CMTAG) AREA	
1304	002150	012706	001100		MOV	#\$CMTAG,R6	:FIRST LOCATION TO BE CLEARED
1305	002154	005026			CLR	(R6)+	:CLEAR MEMORY LOCATION
1306	002156	022706	001140		CMP	#\$SWR,R6 ;:DONE?	
1307	002162	001374			BNE	.-6	:LOOP BACK IF NO
1308	002164	012706	001100		MOV	#\$STACK,SP	:SETUP THE STACK POINTER
1309					::INITIALIZE	A FEW VECTORS	
1310	002170	012737	007064	000030	MOV	#\$ERROR,@#EMTVEC	:EMT VECTOR FOR ERROR ROUTINE
1311	002176	012737	000340	000032	MOV	#\$340,@#EMTVEC+2	:LEVEL 7
1312	002204	012737	012242	000034	MOV	#\$TRAP,@#TRAPVEC	:TRAP VECTOR FOR TRAP CALLS
1313	002212	012737	000340	000036	MOV	#\$340,@#TRAPVEC+2	:LEVEL 7
1314	002220	005037	001160		CLR	\$ESCAPE	:CLEAR THE FESCAPE ON ERROR ADDRESS
1315	002224	112737	000001	001115	MOVB	#1,\$ERMAX	:ALLOW ONE ERROR PER TEST
1316					::SIZE	FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS	
1317					::EQUAL	TO A '-1', SETUP FOR A SOFTWARE SWITCH REGISTER.	
1318	002232	013746	000004		MOV	@#ERRVEC,-(SP)	:SAVE ERROR VECTOR
1319	002236	012737	002272	000004	MOV	#\$64\$,@#ERRVEC	:SET UP ERROR VECTOR
1320	002244	012737	177570	001140	MOV	#\$DSWR,\$SWR	:SETUP FOR A HARDWARE SWICH REGISTER
1321	002252	012737	177570	001142	MOV	#\$DDISP,\$DISPLAY	:AND A HARDWARE DISPLAY REGISTER
1322	002260	022777	177777	176652	CMP	#\$-1,@\$SWR	:TRY TO REFERENCE HARDWARE SWR
1323	002266	001012			BNE	66\$	:BRANCH IF NO TIMEOUT TRAP OCCURRED
1324							:AND THE HARDWARE SWR IS NOT = -1
1325	002270	000403			BR	65\$	:BRANCH IF NO TIMEOUT
1326	002272	012716	002300		64\$: MOV	#\$65\$,(SP)	:SET UP FOR TRAP RETURN
1327	002276	000002			RTI		
1328	002300	012737	000176	001140	65\$: MOV	#\$SWREG,\$SWR	:POINT TO SOFTWARE SWR
1329	002306	012737	000174	001142	MOV	#\$DISPREG,\$DISPLAY	
1330	002314	012637	000704		66\$: MOV	(SP)+,@#ERRVEC	:RESTORE ERROR VECTOR
1331							
1332	002320	000005			RESET		:CLEAR WORLD
1333	002322	012737	000240	000036	MOV	#\$PR5,@#TRAPVEC+2	:CHANGE TRAP PRIORITY BACK TO 5
1334	002330	012737	000240	000032	MOV	#\$PR5,@#EMTVEC+2	:CHANGE EMT PRIORITY BACK TO 5
1335	002336	012737	002132	001300	MOV	#\$BEGIN1,CNTLC	:CONTROL C' ADDRESS

1336	002344	005227	177777		INC	#-1	:FIRST START ?
1337	002350	001025			BNE	1\$	:BR IF NOT
1338	002352	023737	000042	000046	CMP	@#42,@#46	:ACT11 AUTOMATIC MODE?
1339	002360	001004			BNE	4\$	:BRANCH IF NO
1340	002362	012737	000001	013310	MOV	#1, TSTPGM	:ENABLE PROGRAMMABLE DRIVES
1341	002370	000402			BR	5\$	
1342	002372	104401	025574		4\$: TYPE	,TITLE	:TYPE THE PROGRAM'S TITLE
1343	002376	005737	013310		5\$: TST	TSTPGM	:CAN WE USE PROGRAMMABLE DRIVES?
1344	002402	001402			BEQ	6\$	:BRANCH IF NO
1345	002404	104401	021010		TYPE	,USE	:TYPE MSG
1346	002410				6\$:		
1347	002410	122737	000011	000041	CMPB	#11,41	:LOADED FROM AN RP04/5/6 ?
1348	002416	001002			BNE	1\$	:BR IF NOT
1349	002420	104401	025651		TYPE	,LOADRV	:INSTRUCT OPERATOR TO REMOVE 'XXDP'
1350							:PACK FROM DRIVE 0
1351	002424	004737	010432		1\$: JSR	PC,\$TKINT	:TURN ON THE KEYBOARD INTERRUPT
1352					.SBTTL	GET VALUE FOR SOFTWARE	SWITCH REGISTER
1353	002430	005737	000042		TST	@#42	:ARE WE RUNNING UNDER XXDP/ACT?
1354	002434	001006			BNE	67\$	:BRANCH IF YES
1355	002436	023727	001140	000176	CMP	SWR,#SWREG	:SOFTWARE SWITCH REG SELECTED?
1356	002444	001005			BNE	68\$	:BRANCH IF NO
1357	002446	104406			GTSWR		:GET SOFT-SWR SETTINGS
1358	002450	000403			BR	68\$	
1359	002452	112737	000001	001134	67\$: MOV	#1,\$AUTOB	:SET AUTO-MODE INDICATOR
1360	002460				68\$:		
1361	002460	005227	177777		INC	#-1	:CHECK FOR FIRST START
1362	002464	001010			BNE	SETVEC	:BR IF NOT
1363	002466	004737	026076		JSR	PC,BUSADR	:CHECK THE RH11 ADDRESS
1364	002472	013737	001172	012474	MOV	\$RPADR,RPADR	:RH11 ADDRESS
1365	002500	013737	001174	012476	MOV	\$RPVEC,RPVEC	:RH11 VECTOR ADDRESS
1366							
1367							:DISPLAY DRIVE STATUS AND SET UP THE OTHER UNITS THE
1368					:	PROGRAM WILL USE	
1369							
1370	002506	004737	006170		SETVEC: JSR	PC,ST.CLK	:START THE CLOCK
1371	002512	004737	012512		JSR	PC,RPINIT	:INITIALIZE THE RP04/5/6 DRIVER
1372	002516	012737	177777	012434	MOV	#-1,SAVEFG	:SET THE SAVE REGISTERS FLAG
1373	002524	005227	177777		INC	#-1	:SEE IF FIRST START
1374	002530	001404			BEQ	11\$	:BR IF YES
1375	002532	032777	000004	176400	BIT	#SW02,@SWR	:TYPEOUT THE DRIVE STATUS TABLE ?
1376	002540	001105			BNE	10\$	:BR IF NOT
1377	002542	012737	000340	177776	11\$: MOV	#PR7,PS	:SET PRIORITY TO 7
1378	002550	005004			CLR	R4	:DRIVE TABLE POINTER
1379	002552	104401	001167		TYPE	,\$CRLF	:CR-LF
1380	002556	104401	021240		TYPE	,\$SYSTAT	:TYPE STATUS HEADING
1381	002562				1\$:		
1382	002562	010446			MOV	R4,-(SP)	:SAVE R4 FOR TYPEOUT
1383							:TYPE DRIVE NUMBER
1384	002564	104403			TYPOS		:GO TYPE--OCTAL ASCII
1385	002566	002			.BYTE	2	:TYPE 2 DIGIT(S)
1386	002567	000			.BYTE	0	:SUPPRESS LEADING ZEROS
1387	002570	104401	021303		TYPE	,LIN4SP	:SPACES
1388	002574	105764	012346		TSTB	DRVSTA(R4)	:CHECK DRIVE'S STATUS
1389	002600	100425			BMI	4\$	:BR IF UNSAFE
1390	002602	001027			BNE	5\$	:BR IF ONLINE
1391	002604	105764	012356		TSTB	DRVTYP(R4)	:SEE IF OFFLINE OR NONEXISTENT

1392	002610	001404				BEQ	2\$		:BR IF NONEXISTENT
1393	002612	100006				BPL	3\$		:BR IF OFFLINE
1394	002614	104401	021153			TYPE	,NOTRP		:DRIVE NOT AN RP04/5/6
1395	002620	000447				BR	9\$		:CHECK NEXT DRIVE
1396	002622	104401	021174		2\$:	TYPE	,NOTPRS		:DRIVE NOT PRESENT
1397	002626	000444				BR	9\$		:CHECK NEXT DRIVE
1398	002630	132764	000010	012356	3\$:	BITB	#BIT03, DRV TYP(R4)		:DRIVE PROGRAMMABLE?
1399	002636	001403				BEQ	12\$		:BRANCH IF NO
1400	002640	104401	021065			TYPE	,NOUSE		:PRINT MSG
1401	002644	000410				BR	6\$		:PRINT DRIVE TYPE
1402	002646	104401	021132		12\$:	TYPE	,UNTOFF		:DRIVE OFFLINE
1403	002652	000405				BR	6\$		:PRINT DRIVE TYPE
1404	002654	104401	021211		4\$:	TYPE	,NOTSAF		:DRIVE UNSAFE
1405	002660	000402				BR	6\$		:PRINT DRIVE TYPE
1406	002662	104401	021143		5\$:	TYPE	,UNTON		:DRIVE ONLINE
1407	002666	104401	021305		6\$:	TYPE	,LINSF		:SPACES
1408	002672	012737	021221	002736		MOV	#RP04B,8\$		:ADDRESS OF RP04 MESSAGE
1409	002700	132764	000001	012356		BITB	#BIT00, DRV TYP(R4)		:RP04 ?
1410	002706	001012				BNE	7\$		:BR IF YES
1411	002710	012737	021226	002736		MOV	#RP05,8\$		:ADDRESS OF RP05 MESSAGE
1412	002716	132764	000002	012356		BITB	#BIT01, DRV TYP(R4)		:RP05 ?
1413	002724	001003				BNE	7\$		:BR IF YES
1414	002726	012737	021233	002736		MOV	#RP06,8\$		:ADDRESS OF RP06 MESSAGE
1415	002734	104401			7\$:	TYPE			:TYPE THE DRIVE TYPE MESSAGE
1416	002736	000000			8\$:	.WORD	0		:MESSAGE ADDRESS HERE
1417	002740	104401	001167		9\$:	TYPE	, \$CRLF		:CR-LF
1418	002744	005204				INC	R4		:INCREMENT DRIVE NUMBER/TABLE POINTER
1419	002746	020427	000010			CMP	R4, #8.		:FINISHED ?
1420	002752	001303				BNE	1\$		:BR IF NOT
1421	002754	104401	001167		10\$:	TYPE	, \$CRLF		:CR-LF

:SEE WHICH MODE THE PROGRAM IS TO BE RUN IN.
 :MODES ARE: 'FORMAT & VERIFY' OR 'CHECK FORMAT'

1427	002760	005037	001220		M1:	CLR	MODE		:SET MODE TO 'CHECK FORMAT'
1428	002764	104401	021521			TYPE	,MMODE		:TYPE 'PROGRAM MODE'
1429	002770	104411				RDLIN			:READ THE KEYBOARD
1430	002772	012601				MOV	(SP)+,R1		:GET ADDRESS OF INPUT
1431	002774	105711				TSTB	(R1)		: 'CR' ENTERED (DEFAULT)
1432	002776	001414				BEQ	2\$		:BR IF YES
1433	003000	122711	000103			CMPB	#'C,(R1)		: 'CHECK' ?
1434	003004	001406				BEQ	1\$		:BR IF YES
1435	003006	122711	000106			CMPB	#'F,(R1)		: 'FORMAT' ?
1436	003012	001411				BEQ	3\$		:BR IF YES
1437	003014	104401	001166			TYPE	, \$QUES		:NO CORRECT ENTRY
1438	003020	000757				BR	M1		:TRY AGAIN
1439	003022	104401	021574		1\$:	TYPE	,MHECK		:TYPE REST OF 'CHECK'
1440	003026	000410				BR	M1A		:GET STARTING ADDRESS
1441	003030	104401	021553		2\$:	TYPE	,MFORMT		:TYPE DEFAULT MESSAGE
1442	003034	000402				BR	4\$		:SET UP MODE
1443	003036	104401	021607		3\$:	TYPE	,MORMAT		:TYPE REST OF 'FORMAT'
1444	003042	012737	177777	001220	4\$:	MOV	#-1,MODE		:SET MODE TO 'FORMAT & VERIFY'

:FIND OUT IF FORMAT IS TO BE IN 20 OR 22 SECTOR MODE


```

1448 003050 104401 021627 M1A: TYPE ,MSIZE ;TYPE FORMAT MODE REQUEST
1449 003054 104411 RDLIN ;READ THE KEYBOARD
1450 003056 012601 MOV (SP)+,R1 ;ADDRESS OF ENTRY
1451 003060 122711 000131 CMPB #'Y',(R1) ;IS ENTRY A 'Y' ?
1452 003064 001410 BEQ 1$ ;BR IF IT IS
1453 003066 122711 000116 CMPB #'N',(R1) ;IS ENTRY A 'N' ?
1454 003072 001424 BEQ 2$ ;BR IF IT IS
1455 003074 105711 TSTB (R1) ;DEFAULT MODE ?
1456 003076 001403 BEQ 1$ ;BR IF IT IS
1457 003100 104401 001166 TYPE ,SQUES ;TYPE '?'
1458 003104 000761 BR M1A ;TRY AGAIN
1459 003106 104401 021700 1$: TYPE ,MSEC22 ;TYPEOUT MODE SELECTED
1460 003112 012737 013130 001256 MOV #<260.*22.>,WC ;TRACK SIZE IN 22 SECTOR MODE
1461 003120 012737 164650 001260 MOV #-<260.*22.>,MWC ;2'S COMPLEMENT WORD COUNT
1462 003126 012737 177777 001264 MOV #-1,SEC20 ;22 SECTOR INDICATOR
1463 003134 012737 000025 001266 MOV #21.,MAXSEC ;MAX SECTOR ADDRESS IN 22 SECTOR MODE
1464 003142 000415 BR M1B ;CONTINUE
1465 003144 104401 021757 2$: TYPE ,MSEC20 ;TYPE OUT 20 SECTOR MODE SELECTED
1466 003150 012737 012120 001256 MOV #<260.*20.>,WC ;TRACK SIZE IN 20 SECTOR MODE
1467 003156 012737 165660 001260 MOV #-<260.*20.>,MWC ;2'S COMPLEMENT WORD COUNT
1468 003164 005037 001264 CLR SEC20 ;20 SECTOR INDICATOR
1469 003170 012737 000023 001266 MOV #19.,MAXSEC ;MAX SECTOR ADDRESS IN 20 SECTOR MODE
1470 003176 005037 001230 M1B: CLR BEGTRK ;CLEAR STARTING TRACK ADDRESS
1471 003202 005037 001224 CLR BEGCYL ;CLEAR BEGINNING CYLINDER ADDRESS
1472 003206 012737 000022 001226 MOV #18.,ENDTRK ;SETUP END TRACK ADDRESS
1473 003214 012737 000002 001240 MOV #2,PATSEL ;SETUP FOR WORST CASE PATTERN
1474 003222 112737 177777 001356 MOVB #-1,FMTDPB ;SETUP INVALID DRIVE NUMBER
1475 003230 000400 BR MO ;BRANCH TO START
1476
1477 ;GO FIND OUT WHAT DRIVE
1478
1479 003232 012737 006344 001300 MC: MOV #OENTER,CNTLC ;CONTROL C ABORT ENTRANCE
1480 003240 005037 001112 CLR $ERTTL ;CLEAR THE ERROR ACCUMULATOR
1481 003244 004737 010432 JSR PC,$TKINT ;INITIALIZE THE TTY KEYBOARD
1482 003250 104401 021257 TYPE ,MUNIT ;ASK FOR DRIVE NUMBER
1483 003254 104411 RDLIN ;READ THE KEYBOARD
1484 003256 012601 MOV (SP)+,R1 ;ADDRESS OF ENTRY
1485 003260 004537 006654 JSR R5,CK.CHR ;CHECK ONE CHARACTER
1486 003264 003312 3$ ;ILLEGAL CHARACTER
1487 003266 003300 2$ ;CARRIAGE RETURN
1488 003270 003312 3$ ;...
1489 003272 003300 2$ ;...
1490 003274 003320 4$ ;DIGIT 0-7
1491 003276 003312 3$ ;DIGIT 8-9
1492 003300 005037 001214 2$: CLR DRIVE ;SELECT DRIVE ZERO
1493 003304 104401 021301 TYPE ,MDRVD ;GO TYPE DEFAULT DRIVE NUMBER
1494 003310 000413 BR 5$ ;GO SEE IF DRIVE ZERO IS THERE
1495 003312 104401 001166 3$: TYPE ,SQUES ;TYPE '?'
1496 003316 000745 BR MO ;ASK FOR DRIVE NUMBER AGAIN
1497 003320 010237 001214 4$: MOV R2,DRIVE ;SAVE DRIVE NUMBER
1498 003324 005702 TST R2 ;SEE IF DRIVE 0
1499 003326 001004 BNE 5$ ;BR IF NOT
1500 003330 122737 000011 000041 CMPB #11,41 ;PROGRAM LOADED FROM AN RP04/5/6 ?
1501 003336 001431 BEQ 7$ ;BR IF IT WAS, CAN'T FORMAT THE DRIVE
1502 003340 004737 006170 5$: JSR PC,ST.CLK ;START THE CLOCK
1503 003344 004737 012512 JSR PC,RPINIT ;GO SEE WHAT DRIVES ARE AVAILABLE

```


1504	003350	012737	177777	012434		MOV	#-1,SAVEFG	:SAVE THE REGISTERS
1505	003356	012737	177777	012436		MOV	#-1,SEEKFG	:SET 'NO OPTIMIZATION' FLAG
1506	003364	005037	177776			CLR	PS	:SET PRIORITY BACK TO ZERO
1507	003370	105762	012346			TSTB	DRVSTA(R2)	:LOOK AT DRIVE STATUS
1508	003374	003015				BGT	8\$	:BRANCH IF ONLINE
1509	003376	001403				BEQ	6\$	:BR IF DRIVE NOT AVAILABLE
1510	003400	104401	021467			TYPE	,MUSDR	: 'DRIVE UNSAFE'
1511	003404	000712				BR	M0	:GO GET DRIVE NUMBER AGAIN
1512	003406	105762	012356		6\$:	TSTB	DRVTYP(R2)	:A DRIVE PRESENT?
1513	003412	001003				BNE	7\$	:BR IF SO
1514	003414	104401	021362			TYPE	,MDRNP	:TYPE 'DRIVE NOT PRESENT'
1515	003420	000704				BR	M0	:GO GET DRIVE NUMBER AGAIN
1516	003422	104401	021407		7\$:	TYPE	,MER11	: 'DRIVE NOT AVAILABLE'
1517	003426	000701				BR	M0	:GO GET DRIVE NUMBER AGAIN
1518	003430	123737	001214	001356	8\$:	CMPB	DRIVE,FMTDPB	:SAME DRIVE AS LAST TIME ?
1519	003436	001430				BEQ	M2	:BR IF IT IS
1520	003440	113737	001214	001356		MOVB	DRIVE,FMTDPB	:SETUP DRIVE ADDRESS
1521	003446	012737	000632	001222		MOV	#410.,ENDCYL	:SETUP FOR RP04/5
1522	003454	132762	000003	012356		BITB	#BIT00!BIT01,DRVTYP(R2)	:SEE IF DRIVE RP04/5
1523	003462	001003				BNE	9\$	:BR IF EITHER
1524	003464	012737	001456	001222		MOV	#814.,ENDCYL	:SETUP ENDING CYLINDER FOR RP06
1525	003472	005037	001230		9\$:	CLR	BEGTRK	:CLEAR STARTING TRACK ADDRESS
1526	003476	005037	001224			CLR	BEGCYL	:CLEAR STARTING CYLINDER ADDRESS
1527	003502	013737	001222	001400		MOV	ENDCYL,TABLE+2	:ENTRY LIMIT FOR BEGINNING CYLINDER
1528	003510	013737	001222	001414		MOV	ENDCYL,TABLE+16	:ENTRY LIMIT FOR END CYLINDER
1529	003516	000400				BR	M2	:GET ADDRESS LIMITS FROM THE OPERATOR
1530								
1531								:GET ADDRESS LIMITS
1532								
1533	003520	104401	021310		M2:	TYPE	,ENTADR	: 'ENTER ADDRESS LIMITS'
1534	003524	004737	006464			JSR	PC,PARENT	:GET THE ADDRESS LIMITS
1535	003530	023737	001222	001224		CMP	ENDCYL,BEGCYL	:SEE IF ENDING CYLINDER EQ TO GT THAN BEGINNING
1536	003536	101010				BHI	M4	:BR IF HIGHER
1537	003540	103404				BLO	3\$	:BR IF LESS
1538	003542	023737	001226	001230	2\$:	CMP	ENDTRK,BEGTRK	:SEE IF ENDING TRACK EQ OR GT THAN BEGINNING
1539	003550	103003				BHIS	M4	:BR IF YES
1540	003552	104401	022056		3\$:	TYPE	,MADRER	:INVALID ADDRESS ENTERED
1541	003556	000760				BR	M2	:TRY AGAIN
1542								
1543								:GO GET DATA PATTERN FOR FORMAT
1544								
1545	003560	104401	022160		M4:	TYPE	,MSELP	:GO TYPE 'SELECT PATTERN'
1546	003564	104411				RDLIN		:READ THE KEYBOARD
1547	003566	012601				MOV	(SP)+,R1	:ENTRY ADDRESS
1548	003570	004537	006654			JSR	R5,CK.CHR	:CHECK ONE CHARACTER
1549	003574	003630				3\$		:ILLEGAL CHARACTER
1550	003576	003610				1\$		:CARRIAGE RETURN
1551	003600	003630				3\$		:...
1552	003602	003610				1\$		:...
1553	003604	003622				2\$		:DIGIT 0-7
1554	003606	003630				3\$		:DIGIT 8-9
1555	003610	104401	022260		1\$:	TYPE	,MPATD	:TYPE DEFAULT PATTERN
1556	003614	012702	000002			MOV	#2,R2	:WORST CASE PATTERN
1557	003620	000406				BR	4\$	:GO SAVE PATTERN
1558	003622	020227	000002		2\$:	CMP	R2,#2	:IS # LARGER THAN 2
1559	003626	101403				BLOS	4\$	:BRANCH IF NOT


```

1560 003630 104401 001166      3$:  TYPE  ,SQUES      ;TYPE '?'
1561 003634 000751              BR      M4          ;RETYPE LINE
1562 003636 010237 001240      4$:  MOV      R2,PATSEL  ;SAVE PATTERN SELECTED
1563
1564      ;GO TYPE 'STARTING FORMAT ON DRIVE N'
1565
1566 003642 012737 006364 001300 M5:  MOV      #TYPADR,CNTLC ;CHANGE ^C ENTRANCE
1567 003650 005737 001220      TST      MODE      ;'FORMAT' OR 'CHECK' MODE ?
1568 003654 001403              BEQ      1$          ;BR IF 'CHECK' MODE
1569 003656 104401 022273      TYPE  ,MSFOU      ;TYPE 'STARTING FORMAT ON DRIVE N'
1570 003662 000402              BR      2$          ;
1571 003664 104401 022330      1$:  TYPE  ,MSCHK      ;TYPE 'STARTING CHECK ON DRIVE N'
1572 003670              2$:
1573 003670 013746 001214      MOV      DRIVE,-(SP)  ;;SAVE DRIVE FOR TYPEOUT
1574      ;TYPE DRIVE NUMBER
1575 003674 104403              TYPOS
1576 003676      001          .BYTE  1          ;;GO TYPE--OCTAL ASCII
1577 003677      000          .BYTE  0          ;;TYPE 1 DIGIT(S)
1578 003700 104401 001167      TYPE  ,SCRLF      ;;SUPPRESS LEADING ZEROS
1579      ;CR-LF
1580      ;SETUP TOTAL TRACK COUNT FOR FORMAT
1581
1582 003704 023737 001222 001224 CKADRS: CMP      ENDCYL,BEGCYL ;STARTING AND ENDING CYLINDERS THE SAME ?
1583 003712 001011              BNE      1$          ;BRANCH IF BEGCYL NOT EQUAL TO ENDCYL
1584 003714 013737 001226 001232      MOV      ENDTRK,TTRKS ;END TRACK ADDRESS
1585 003722 163737 001230 001232      SUB      BEGTRK,TTRKS ;SUBTRACT THE STARTING TRACK ADDRESS
1586 003730 005237 001232              INC      TTRKS      ;MAKE THE NUMBER OF TRACKS INCLUSIVE
1587 003734 000427              BR      SETPAT      ;SETUP THE DATA PATTERN
1588 003736 013702 001222      1$:  MOV      ENDCYL,R2      ;ENDING CYLINDER
1589 003742 163702 001224      SUB      BEGCYL,R2      ;SUBTRACT THE STARTING CYLINDER
1590 003746 013737 001226 001232      MOV      ENDTRK,TTRKS ;CALCULATE THE RESIDUAL TRACKS
1591 003754 163737 001230 001232      SUB      BEGTRK,TTRKS ;SUBTRACT THE STARTING TRACK
1592 003762 100004              BPL      2$          ;BR IF ENDING TRACK GREATER
1593 003764 062737 000023 001232      ADD      #19.,TTRKS ;CORRECT THE VALUE
1594 003772 005302              DEC      R2          ;COUNT THE PARTIAL TRACK
1595 003774 005237 001232      2$:  INC      TTRKS      ;MAKE THE NUMBER INCLUSIVE
1596 004000 005302              3$:  DEC      R2          ;DECREMENT THE CYLINDER COUNT
1597 004002 100404              BMI      SETPAT      ;BR IF DONE
1598 004004 062737 000023 001232      ADD      #19.,TTRKS ;ADD CYLINDER WORTH OF TRACKS
1599 004012 000772              BR      3$          ;CONTINUE
1600
1601      ;THIS CODE SETS UP THE SELECTED DATA PATTERN TO BE WRITTEN ON EACH SECTOR IMAGE
1602
1603 004014 005037 001242      SETPAT: CLR      PATA      ;CLEAR DATA PATTERN A
1604 004020 005037 001244      CLR      PATB      ;CLEAR DATA PATTERN B
1605 004024 005737 001240      TST      PATSEL      ;SEE IF PATTERN OF ONES
1606 004030 001416              BEQ      1$          ;BR IF SO
1607 004032 005137 001242      COM      PATA      ;SET PATA TO ONES
1608 004036 005137 001244      COM      PATB      ;SET PATB TO ONES
1609 004042 022737 000001 001240      CMP      #1,PATSEL ;SEE IF PATTERN OF ZEROS
1610 004050 001406              BEQ      1$          ;BRANCH IF SO
1611 004052 012737 165555 001242      MOV      #165555,PATA ;SET UP WORST CASE
1612 004060 012737 133333 001244      MOV      #133333,PATB ;SET UP WORST CASE
1613 004066 013703 001256      1$:  MOV      WC,R3      ;SET UP COUNTER
1614 004072 012700 025574      MOV      #BUFP,R0 ;SET UP MEMORY POINTER
1615 004076 013701 001242      MOV      PATA,R1 ;SET UP PATTERN IN R1
  
```



```

1616 004102 013702 001244      MOV      PATB,R2      ;SET UP PATTERN IN R2
1617 004106 010120      2$:    MOV      R1,(R0)+      ;MOV 1ST PAT INTO MEM
1618 004110 010220      MOV      R2,(R0)+      ;MOV 2ND PAT INTO MEM
1619 004112 005303      DEC      R3      ;KEEP COUNT
1620 004114 005303      DEC      R3      ;KEEP COUNT
1621 004116 001373      BNE      2$      ;DO IT AGAIN IF R3 NOT 0
1622
1623      ;THIS CODE SETS UP FOR THE ACTUAL FORMAT
1624
1625 004120 004737 005630      WRTRK:  JSR      PC,SETTBL      ;GO SET UP DRIVER TABLE
1626 004124 004737 006030      JSR      PC,SETHDR      ;GO INITIALIZE HEADERS IN THE SECTOR BUFFER IN CORE
1627 004130 112737 000020 001357      MOVB     #20,FMTDPB+1      ;LOAD FMT22 BIT
1628 004136 005737 001264      TST      SEC20      ;18 BIT MODE ?
1629 004142 001002      BNE      1$      ;BR IF NOT
1630 004144 105037 001357      CLRB     FMTDPB+1      ;CLEAR THE FMT22 BIT
1631 004150 112737 000143 001360 1$:    MOVB     #SETFMT,FMTDPB+2      ;'LOAD FORMAT' COMMAND
1632 004156 004037 013312      2$:    JSR      R0,RP04      ;START THE COMMAND
1633 004162 001356      FMTDPB      ;DPB ADDRESS
1634 004164 000774      BR       2$      ;QUEUE FULL RETURN
1635 004166 005737 001374      3$:    TST      FMTDPB+16      ;LOOK FOR DONE
1636 004172 001775      BEQ      3$      ;LOOP UNTIL FINISHED
1637 004174 005037 001216      WRTRK1: CLR      SOFSW      ;CLEAR ERROR COUNTER
1638 004200 005037 001250      CLR      RETRY      ;ZERO THE RETRY COUNTER
1639 004204 005037 001304      CLR      EFLG      ;CLEAR ERROR DURING WRITE CHECK FLAG
1640 004210 105037 001366      CLRB     FMTDPB+10      ;RESTORE SECTOR
1641 004214 013737 001260 001362      MOV      MWC,FMTDPB+4      ;RESTORE WC
1642 004222 012737 025574 001364      MOV      #BUFP,FMTDPB+6      ;RESTORE BA
1643 004230 005737 001220      TST      MODE      ;'FORMAT' OR 'CHECK' MODE ?
1644 004234 001450      BEQ      CKTRK      ;BR IF 'CHECK' MODE
1645 004236 112737 000163 001360  WRTRK2: MOVB     #WRTHD,FMTDPB+2      ;SET WRITE HEADER & DATA COMMAND IN TBL
1646 004244 012737 004244 001110      MOV      #,$LPERR      ;SETUP LOOP ON ERROR ADDRESS
1647 004252 012706 001100      MOV      #STACK,SP      ;LOAD STACK POINTER
1648 004256 004037 013312      1$:    JSR      R0,RP04      ;GO FORMAT A TRACK
1649 004262 001356      FMTDPB      ;ADRS OF PARAMETERS - TBL
1650 004264 000774      BR       1$      ;WAIT FOR QUEUE IF FULL
1651 004266 005737 001374      2$:    TST      FMTDPB+16      ;WAIT FOR COMMAND TO COMPLETE
1652 004272 001775      BEQ      2$      ;BRANCH IF NOT DONE
1653 004274 100024      BPL      4$      ;BRANCH IF NO ERROR
1654 004276 012737 004324 001160      MOV      #3$, $ESCAPE      ;ESCAPE TO 3$ ON ERROR
1655 004304 004737 005462      JSR      PC,ERINDX      ;SEE WHICH ERROR
1656 004310 104010      ERROR    10      ;DRIVE OFFLINE
1657 004312 104011      ERROR    11      ;PERSISTENT DRIVE UNSAFE ERROR
1658 004314 104012      ERROR    12      ;UNCORRECTABLE MASSBUS PARITY ERROR
1659 004316 104013      ERROR    13      ;SOFTWARE TIMEOUT
1660 004320 104014      ERROR    14      ;DRIVE UNSAFE ERROR
1661 004322 104015      ERROR    15      ;DRIVE/CONTROLLER ERROR DURING WRITE
1662 004324 004737 005562 001250 3$:    JSR      PC,LOP.CK      ;LOOP ON THE ERROR ?
1663 004330 022737 000003      CMP      #3,RETRY      ;ERROR RETRY LIMIT ?
1664 004336 001403      BEQ      4$      ;BR IF REACHED
1665 004340 005237 001250      INC      RETRY      ;COUNT THE ERROR
1666 004344 000744      BR       1$      ;TRY AGAIN
1667 004346 005037 001160      4$:    CLR      $ESCAPE      ;CLEAR ERROR ESCAPE ADDRESS
1668 004352 005037 001250      CLR      RETRY      ;CLEAR THE RETRY COUNTER
1669
1670      ;CHECK THE TRACK JUST WRITTEN
1671

```


1672	004356	112737	000153	001360	CKTRK:	MOVB	#WCKHD,FMTDPB+2	:SET WRITE CHECK HEADER & DATA COMMAND IN TBL
1673	004364	012737	004364	001110		MOV	#, \$LPERR	:SETUP LOOP ON ERROR ADDRESS
1674	004372	012706	001100			MOV	#STACK, SP	:LOAD STACK POINTER
1675	004376	004037	013312		1\$:	JSR	R0, RP04	:GO CHECK THE TRACK JUST FORMATTED
1676	004402	001356				FMTDPB		:ADRS OF PARAMETERS - TBL
1677	004404	000774				BR	1\$	:WAIT FOR QUEUE IF FULL
1678	004406	005737	001374		2\$:	TST	FMTDPB+16	:WAIT FOR COMMAND TO COMPLETE
1679	004412	001775				BEQ	2\$	:BRANCH IF NOT DONE
1680	004414	100147				BPL	8\$	:BRANCH IF NO ERROR
1681	004416	113737	001314	001252		MOVB	RP.REG+RPDA, SAVSEC	:GET THE SECTOR ADDRESS
1682	004424	001005				BNE	3\$	:BRANCH IF NOT SECTOR 0
1683	004426	013737	001266	001252		MOV	MAXSEC, SAVSEC	:RESTORE TO LAST SECTOR +1
1684	004434	005237	001252			INC	SAVSEC	:INCREMENT TO LAST SECTOR +1
1685	004440	005337	001252		3\$:	DEC	SAVSEC	:ADJUST SECTOR TO THE ONE THAT FAILED
1686	004444	012737	005010	001160		MOV	#10\$, \$ESCAPE	:ESCAPE TO 10\$ ON ERROR
1687	004452	004737	005462			JSR	PC, ERINDX	:SEE WHICH ERROR
1688	004456	104010				ERROR	10	:DRIVE OFFLINE
1689	004460	104011				ERROR	11	:PERSISTENT DRIVE UNSAFE ERROR
1690	004462	104012				ERROR	12	:UNCORRECTABLE MASSBUS PARITY ERROR
1691	004464	104013				ERROR	13	:SOFTWARE TIMEOUT
1692	004466	104014				ERROR	14	:DRIVE UNSAFE ERROR
1693	004470	032737	040000	001316		BIT	#WCE, RP.REG+RPCS2	:WRITE CHECK ERROR ?
1694	004476	001005				BNE	4\$	:BR IF SET
1695	004500	033727	001322			BIT	RP.REG+RPER1, (PC)+	:CHECK FOR DATA ERRORS
1696	004504	130620				.WORD	DCK!OPI!DTE!HCRC!HCE!FER	:DATA ERROR BITS
1697	004506	001001				BNE	4\$	:BR IF SET
1698	004510	104016				ERROR	16	:CONTROLLER/DRIVE ERROR DURING WRITE CHECK
1699	004512	005737	001220		4\$:	TST	MODE	:FORMAT OR CHECK MODE ?
1700	004516	001416				BEQ	5\$	:BR IF CHECK
1701	004520	005737	001216			TST	SOF SW	:RETRYING THE SECTOR ?
1702	004524	001413				BEQ	5\$	:BR IF NOT
1703	004526	012737	005034	001160		MOV	#11\$, \$ESCAPE	:ESCAPE TO 11\$ ON ERROR
1704	004534	005737	001304			TST	EFLG	:ERROR DURING WRITE CHECK REPORTED ?
1705	004540	001402				BEQ	21\$	:NO, INCREMENT ERROR COUNT
1706	004542	005337	001112			DEC	\$ERTTL	:YES, DECREMENT REPORTED ERROR COUNT
1707	004546	005037	001304		21\$:	CLR	EFLG	:CLEAR ERROR DURING WRITE CHECK FLAG
1708	004552	104017				ERROR	17	:SECTOR NOT ACCEPTABLE
1709	004554	005037	001160		5\$:	CLR	\$ESCAPE	:CLEAR THE ERROR ESCAPE ADDRESS
1710	004560	005737	001322			TST	RP.REG+RPER1	:DATA CHECK ERROR ?
1711	004564	100011				BPL	23\$	:NO, CHECK OTHER ERRORS
1712	004566	032777	000200	174344		BIT	#BIT7, @SWR	:INHIBIT SOFT ERROR REPORTS ?
1713	004574	001426				BEQ	7\$	:YES, SKIP ERROR REPORT
1714	004576	012737	000001	001304		MOV	#1, EFLG	:SET ERROR DURING WRITE CHECK FLAG
1715	004604	104020				ERROR	20	:DATA ERROR DURING WRITE CHECK
1716	004606	000421				BR	7\$	:CONTINUE
1717	004610	032737	040000	001316	23\$:	BIT	#WCE, RP.REG+RPCS2	:WRITE CHECK ERROR ?
1718	004616	001411				BEQ	24\$	:NO, MUST BE HARDWARE ERROR
1719	004620	032777	000200	174312		BIT	#BIT7, @SWR	:INHIBIT SOFT ERROR REPORTS ?
1720	004626	001411				BEQ	7\$	:YES, SKIP ERROR REPORT
1721	004630	012737	000001	001304		MOV	#1, EFLG	:SET ERROR DURING WRITE CHECK FLAG
1722	004636	104024				ERROR	24	:WRITE CHECK ERROR
1723	004640	000404				BR	7\$	:CONTINUE
1724	004642	012737	000001	001304	24\$:	MOV	#1, EFLG	:SET ERROR DURING WRITE CHECK FLAG
1725	004650	104025				ERROR	25	:HARDWARE ERROR DURING WRITE CHECK
1726	004652	013701	001252		7\$:	MOV	SAVSEC, R1	:FAILING SECTOR ADDRESS
1727	004656	113737	001252	001366		MOVB	SAVSEC, FMTDPB+10	:SECTOR ADDRESS


```

1728 004664 006301          ASL      R1          ;SETUP INDEX TO ADDRESS WORDS
1729 004666 016137 001500 001364 MOV     ADRTBL(R1),FMTDPB+6 ;BUFFER ADDRESS FOR FAILING SECTOR
1730 004674 013746 001260      MOV     MWC,-(SP) ;GET MAXIMUM WORD COUNT VALUE (2'S COMP)
1731 004700 066116 001554      ADD     WCTBL(R1),(SP) ;ADD WORD COUNT THROUGH FAILING SECTOR
1732 004704 012637 001254      MOV     (SP)+,SAVWC ;STORE WORD COUNT FOR REMAINDER OF TRACK
1733 004710 005737 001220      TST     MODE ;FORMAT OR CHECK MODE ?
1734 004714 001421          BEQ      9$ ;BR IF CHECK
1735 004716 012737 177374 001362 MOV     #-260.,FMTDPB+4 ;WORD COUNT FOR 1 SECTOR
1736 004724 005237 001216      INC     SOFSW ;INDICATE THAT A RETRY IS IN PROGRESS
1737 004730 000137 004236      JMP     WRTRK2 ;REFORMAT ERROR SECTOR
1738 004734 005737 001216      8$: TST     SOFSW ;RETRY IN PROGRESS ?
1739 004740 001435          BEQ      11$ ;BR IF NOT
1740 004742 022737 000002 001216 CMP     #2,SOFSW ;SEE IF LAST RETRY
1741 004750 001403          BEQ      9$ ;BR IF IT IS
1742 004752 005237 001216      INC     SOFSW ;INCREMENT RETRY COUNT
1743 004756 000607          BR       1$ ;READ AGAIN
1744 004760 123737 001266 001366 9$: CMPB   MAXSEC,FMTDPB+10 ;SEE IF LAST SECTOR ON THE TRACK
1745 004766 001422          BEQ      11$ ;BR IF IT IS
1746 004770 004737 006002      JSR     PC,SCAWC ;SETUP TO CHECK REMAINING SECTORS ON THE TRACK
1747 004774 005037 001216      CLR     SOFSW ;CLEAR RETRY COUNTER
1748 005000 005037 001304      CLR     EFLG ;CLEAR ERROR DURING WRITE CHECK FLAG
1749 005004 000137 004376      JMP     1$ ;FINISH CHECKING THE TRACK
1750 005010 004737 005562      10$: JSR     PC,LOP.CK ;CHECK FOR LOOP ON ERROR
1751 005014 022737 000003 001250 CMP     #3,RETRY ;ERROR RETRY REACHED ?
1752 005022 001404          BEQ      11$ ;BR IF IT IS
1753 005024 005237 001250      INC     RETRY ;COUNT THE RETRY
1754 005030 000137 004376      JMP     1$ ;DO THE WRITE CHECK AGAIN
1755 005034 005037 001160      11$: CLR     $ESCAPE ;CLEAR THE ERROR RETURN ESCAPE ADDRESS
1756 005040 005037 001250      CLR     RETRY ;CLEAR THE RETRY COUNTER
1757 005044 005037 001304      CLR     EFLG ;CLEAR ERROR DURING WRITE CHECK FLAG
1758 005050 032777 000002 174062 BIT     #BIT1,@SWR ;LOOP ON CURRENT TRACK ?
1759 005056 001402          BEQ      12$ ;BR IF NOT
1760 005060 000137 004174      JMP     WRTRK1 ;START AGAIN ON THE SAME TRACK
1761 005064 004737 005720      12$: JSR     PC,TRKTST ;GET UPDATED TRACK AND CYLINDER ADDRESSES
1762 005070 000402          BR       HDREAD ;RETURN HERE IF DONE - DO QUICK CHECK OF DISK
1763 005072 000137 004174      JMP     WRTRK1 ;CONTINUE WITH THE FORMAT
1764
1765 ;THIS CODE MAKES SURE EACH CYLINDER FORMATTED CONTAINS THE
1766 ;PROPER CYLINDER ADDRESS. THE PROGRAM IS LOOKING FOR
1767 ;POSSIBLE POSITIONER ERRORS THAT MAY HAVE OCCURRED DURING THE FORMAT.
1768
1769 005076 005037 001262      HDREAD: CLR     HEDERR ;CLEAR HEADER CHECK ERROR INDICATOR
1770 005102 004737 005630      JSR     PC,SETTBL ;GO SET UP DRIVER TABLE
1771 005106 004737 006030      JSR     PC,SETHDR ;GO SET UP HEADERS IN CORE
1772 005112 013737 001224 001246 MOV     BEGCYL,CYLCK ;USE 'BEGCYL' FOR FORMAT VERIFICATION
1773 005120 012737 177774 001362 MOV     #-4,FMTDPB+4 ;SET UP WORD COUNT
1774 005126 012737 025564 001364 MOV     #RBUF,FMTDPB+6 ;SET UP BUFFER ADRS
1775 005134 005037 001366      CLR     FMTDPB+10 ;CLEAR THE SECTOR & TRACK ADDRESS FIELD
1776 005140 013737 001224 001370 MOV     BEGCYL,FMTDPB+12 ;SETUP THE CYLINDER FIELD
1777 005146 112737 000173 001360 MOVB   #RDHD,FMTDPB+2 ;SET UP READ HEADER & DATA COMMAND
1778 005154 012737 005154 001110 MOV     #,$LPERR ;SETUP LOOP ON ERROR ADDRESS
1779 005162 012706 001100      MOV     #STACK,SP ;LOAD STACK POINTER
1780 005166 004037 013312      1$: JSR     R0,RP04 ;GO READ 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
```

\$EOP:

```

3$:  TST      MODE      ;'FORMAT' OR 'CHECK' MODE ?
4$:  BEQ      3$        ;BR IF 'CHECK'
    TYPE     ,MFCMPT    ;'FORMAT COMPLETE'
    BR       4$        ;FINISH THE END MESSAGE
    TYPE     ,MCCMPT    ;'CHECK COMPLETE'
4$:  TYPE     ,NUMERR    ;'TOTAL ERRORS DETECTED: '
    MOV      $ERTTL,-(SP) ;:SAVE $ERTTL FOR TYPEOUT
    TYPDS    ;:GO TYPE--DECIMAL ASCII WITH SIGN
    TYPE     , $CRLF    ;:CR-LF
    INC      $PASS      ;:INCREMENT THE PASS NUMBER
    BIC      #100000,$PASS ;:DON'T ALLOW A NEG. NUMBER
    DEC      (PC)+      ;:LOOP?

```

SEOPCT: .WORD

BGT

MOV

SENDCT: .WORD

\$EOPC

```
$GET42: MOV     @#42,R0
```

; 'FORMAT' OR 'CHECK' MODE ?

```

:BR IF 'CHECK'

```

```

: 'FORMAT COMPLETE'

```

```
;FINISH THE END MESSAGE
```

;'CHECK COMPLETE'

; 'TOTAL ERRORS DETECTED: '

```
;;SAVE $ERTTL FOR TYPEOUT
```

```
;;GO TYPE--DECIMAL ASCII WITH SIGN
```

```
;CR-LF
```

```

:: INCREMENT THE PASS NUMBER

```

;;DON'T ALLOW A NEG. NUMBER

```
;;LOOP?
```

WILLIAMS

:::YES

```

::RESTORE COUNTER

```

FILES

```
;;GET MONITOR ADDRESS
```



```

1837 005416 001405          BEQ      $DOAGN          ;;BRANCH IF NO MONITOR
1838 005420 000005          RESET          ;;CLEAR THE WORLD
1839 005422 004710          $ENDAD: JSR      PC,(R0)    ;;GO TO MONITOR
1840 005424 000240          NOP              ;;SAVE ROOM
1841 005426 000240          NOP              ;;FOR
1842 005430 000240          NOP              ;;ACT11
1843 005432          $DOAGN:          ;;RETURN
1844 005432 000137          JMP      @PC)+
1845 005434 005436          $RTNAD: .WORD  DONE
1846 005436 032777 000001 173474 DONE:  BIT      #SW0,@SWR    ;SEE IF SWR 0 SET
1847 005444 001002          BNE      1$          ;BR IF SET
1848 005446 000137 003232          JMP      MO          ;ASK FOR NEXT DRIVE
1849 005452 012706 001100          1$:  MOV      #STACK,SP    ;RESET STACK POINTER
1850 005456 000137 003642          JMP      M5          ;DO THE FORMAT OR CHECK AGAIN
1851
1852          ;;*****
1853
1854          .SBTTL  SUPPORT SUBROUTINES
1855
1856          ;;*****
1857
1858          ;THIS ROUTINE DETERMINES THE ERROR TYPE
1859
1860 005462 010146          ERINDX: MOV      R1,-(SP)      ;STORE R1
1861 005464 005001          CLR      R1          ;CLEAR R1
1862 005466 033727 001374          BIT      FMTDPB+16,(PC)+ ;CHECK ERROR TYPE
1863 005472 060006          .WORD  BIT14!BIT13!BIT2!BIT1 ;DRIVE OFFLINE
1864 005474 001025          BNE      5$          ;BR IF OFFLINE
1865 005476 033727 001374          BIT      FMTDPB+16,(PC)+ ;CHECK ERROR TYPE
1866 005502 010000          .WORD  BIT12          ;DRIVE PERSISTENTLY UNSAFE
1867 005504 001020          BNE      4$          ;BR IF UNSAFE
1868 005506 033727 001374          BIT      FMTDPB+16,(PC)+ ;CHECK ERROR TYPE
1869 005512 006000          .WORD  BIT11!BIT10        ;UNCORRECTABLE MASSBUS PARITY ERROR ?
1870 005514 001013          BNE      3$          ;BR IF PARITY ERROR
1871 005516 033727 001374          BIT      FMTDPB+16,(PC)+ ;CHECK ERROR TYPE
1872 005522 001400          .WORD  BIT09!BIT08        ;SOFTWARE TIMEOUT ?
1873 005524 001006          BNE      2$          ;BR IF YES
1874 005526 033727 001374          BIT      FMTDPB+16,(PC)+ ;CHECK ERROR TYPE
1875 005532 000020          .WORD  BIT04          ;DRIVE UNSAFE ERROR ?
1876 005534 001001          BNE      1$          ;BR IF YES
1877 005536 005201          INC      R1          ;INCREMENT THE RETURN INDEX
1878 005540 005201          1$:  INC      R1          ;INCREMENT THE RETURN INDEX
1879 005542 005201          2$:  INC      R1          ;INCREMENT THE RETURN INDEX
1880 005544 005201          3$:  INC      R1          ;INCREMENT THE RETURN INDEX
1881 005546 005201          4$:  INC      R1          ;INCREMENT THE RETURN INDEX
1882 005550 006301          5$:  ASL      R1          ;DOUBLE THE INCREMENT
1883 005552 060166 000002          ADD      R1,2(SP)      ;DEVELOP THE RETURN ADDRESS
1884 005556 012601          MOV      (SP)+,R1          ;RESTORE R1
1885 005560 000207          RTS      PC          ;RETURN
1886
1887          ;ROUTINE TO CHECK FOR LOOP ON ERROR
1888
1889 005562 032777 001000 173350 LOP.CK: BIT      #SW09,@SWR    ;LOOP ON ERROR ?
1890 005570 001402          BEQ      1$          ;BR IF NOT
1891 005572 000177 173312          JMP      @SLPERR        ;GO TO THE LOOP ON ERROR ADDRESS
1892 005576 005037 001160          1$:  CLR      $ESCAPE      ;CLEAR THE ERROR ESCAPE ADDRESS

```



```

1893 005602 033727 001374          BIT    FMTDPB+16,(PC)+ ;CHECK FOR 'FATAL' ERROR
1894 005606 072002          .WORD    BIT14!BIT13!BIT12!BIT10!BIT01 ;'FATAL' ERROR BITS
1895 005610 001004          BNE      2$ ;BR IF NOT
1896 005612 032737 004000 001322    BIT    #WLE,RP.REG+RPER1 ;WRITE LOCK ERROR ?
1897 005620 001402          BEQ      3$ ;BR IF NOT
1898 005622 000137 005326          2$:    JMP    $EOP ;TERMINATE THE FORMAT
1899 005626 000207          3$:    RTS    PC ;RETURN
1900
1901          ;THIS ROUTINE SETS UP THE INPUT TABLE FOR THE DRIVER ROUTINE
1902
1903 005630 113737 001214 001356    SETTBL: MOV    DRIVE,FMTDPB ;SET UP DRIVE #
1904 005636 105037 001361          CLRB   FMTDPB+3 ;CLEAR HIGH ORDER ADRS BITS
1905 005642 013737 001260 001362    MOV     MWC,FMTDPB+4 ;LOAD UP WORD COUNT
1906 005650 012737 025574 001364    MOV     #BUFP,FMTDPB+6 ;LOAD UP CURRENT ADRS
1907 005656 105037 001366          CLRB   FMTDPB+10 ;SET SECTOR TO ZERO
1908 005662 113737 001230 001367    MOV    BEGTRK,FMTDPB+11 ;SET UP STARTING TRK ADRS
1909 005670 013737 001224 001370    MOV     BEGCTL,FMTDPB+12 ;SET UP STARTING CYL
1910 005676 005037 001374          CLR     FMTDPB+16 ;CLEAR RP04 STATUS
1911 005702 013737 001230 001236    MOV     BEGTRK,TRKCNT ;SET UP PARTIAL CYL TRACK COUNT
1912 005710 013737 001232 001234    MOV     TTRKS,TTRKSC ;SET UP TOTAL TRACKS COUNTER
1913 005716 000207          RTS      PC ;RETURN FROM SETUP
1914
1915          ;THIS ROUTINE CONTROLS THE DISK ADDRESSING AND TOTAL TRK COUNT
1916          ;IT IS ENTERED AFTER EVERY TRK OPERATION
1917
1918 005720 005337 001234          TRKTST: DEC    TTRKSC ;SEE IF LAST TRACK
1919 005724 001425          BEQ      3$ ;BRANCH IF LAST TRACK
1920 005726 022737 000022 001236    CMP     #18.,TRKCNT ;IS THIS THE LAST TRACK IN THE CYLINDER ?
1921 005734 001407          BEQ      1$ ;BRANCH IF SO
1922 005736 005237 001236          INC     TRKCNT ;COUNT UP TRACK WITHIN CYLINDER
1923 005742 105237 001367          INCB   FMTDPB+11 ;INCREMENT TRACK NUMBER
1924 005746 004737 006140          JSR     PC,UPDATK ;UPDATE TRACK ADDRESS IN BUFFER
1925 005752 000410          BR       2$ ;EXIT
1926 005754 005037 001236          1$:    CLR    TRKCNT ;CLEAR TRACK COUNT FOR NEXT CYLINDER
1927 005760 105037 001367          CLRB   FMTDPB+11 ;RESET TRACK ADDRESS TO 0
1928 005764 005237 001370          INC     FMTDPB+12 ;UPDATE CYLINDER NUMBER
1929 005770 004737 006110          JSR     PC,UPDACY ;UPDATE CYLINDER NUMBER IN BUFFER
1930 005774 062716 000002          2$:    ADD     #2,(SP) ;ADD TWO TO RETURN ADRS
1931 006000 000207          3$:    RTS    PC ;RETURN
1932
1933          ;THIS ROUTINE SETS UP WC & BA FOR REMAINING SECTORS
1934          ;AFTER A WRITE CHECK ERROR
1935
1936 006002 013737 001254 001362    SCAWC: MOV    SAVWC,FMTDPB+4 ;SET UP WC FOR REMAINING SECTORS
1937 006010 062737 001010 001364    ADD     #520.,FMTDPB+6 ;SET UP BA FOR REMAINING SECTORS
1938 006016 005237 001366          INC     FMTDPB+10 ;ADVANCE TO NEXT SECTOR IN TBL
1939 006022 005037 001216          CLR     SOFSW ;RESET RETRY COUNTER
1940 006026 000207          RTS      PC ;RETURN TO COMPLETE TRK FORMAT
1941
1942          ;THIS ROUTINE SETS UP THE CYLINDER ADRS, FORMAT BIT, TRACK AND
1943          ;SECTOR ADRS IN MEMORY WITH THE STARTING CYL - TRK INFORMATION
1944
1945
1946 006030 013701 001266          SETHDR: MOV    MAXSEC,R1 ;SET UP SECTOR COUNT
1947 006034 012700 025574          MOV     #BUFP,R0 ;SET UP HEADER POINTER IN R0
1948 006040 013702 001224          MOV     BEGCTL,R2 ;PUT STARTING CYL # IN R2
  
```


1949	006044	013703	001230		MOV	BEGTRK,R3	:PUT STARTING TRK # IN R3
1950	006050	000303			SWAB	R3	:JUSTIFY TRACK ADRS
1951	006052	005737	001264		TST	SEC20	:SEE IF 20 OR 22 SECTOR MODE
1952	006056	001402			BEQ	1\$	:BR IF 20 SECTOR MODE
1953	006060	052702	010000		BIS	#10000,R2	:SET THE 22 SECTOR FORMAT BIT
1954	006064	010220		1\$:	MOV	R2,(R0)+	:WRITE IN HEADER AREA OF CORE THE CYL ADRS
1955	006066	010320			MOV	R3,(R0)+	:WRITE IN HEADER AREA OF CORE THE TRK ADRS
1956	006070	005020			CLR	(R0)+	:CLR 1ST KEYWORD
1957	006072	005010			CLR	(R0)	:CLR 2ND KEYWORD
1958	006074	062700	001002		ADD	#514.,R0	:SET UP FOR NEXT HEADER
1959	006100	005203			INC	R3	:UPDATE SECTOR ADRS FOR NEXT HEADER
1960	006102	005301			DEC	R1	:MAINTAIN COUNT OF SECTORS
1961	006104	002367			BGE	1\$	:BRANCH IF NOT LAST SECTOR
1962	006106	000207			RTS	PC	:EXIT - HEADERS ARE LOADED INTO CORE
1963							
1964							:THIS ROUTINE UPDATES THE CYLINDER ADRS OF THE HEADER WORDS IN CORE
1965							
1966	006110	013701	001266		UPDACY: MOV	MAXSEC,R1	:SET UP SECTOR COUNT
1967	006114	012700	025574			MOV	#BUFP,R0
1968	006120	005220		1\$:	INC	(R0)+	:SET UP HEADER POINTER IN R0
1969	006122	042710	177400		BIC	#177400,(R0)	:INCREMENT FOR NEXT CYLINDER
1970	006126	062700	001006		ADD	#518.,R0	:RESET TRK ADRS TO 0
1971	006132	005301			DEC	R1	:SET UP FOR NEXT HEADER
1972	006134	002371			BGE	1\$	:COUNT SECTORS
1973	006136	000207			RTS	PC	:BRANCH IF NOT LAST SECTOR
1974							:EXIT
1975							
1976							:THIS ROUTINE UPDATES THE TRACK ADRS OF THE HEADER WORDS IN CORE
1977	006140	013701	001266		UPDATK: MOV	MAXSEC,R1	:SET UP SECTOR COUNT
1978	006144	012700	025574			MOV	#BUFP,R0
1979	006150	005720			TST	(R0)+	:SET UP HEADER POINTER IN R0
1980	006152	062710	000400	1\$:	ADD	#400,(R0)	:POINT HEADER POINTER TO TRK - SEC ADRS
1981	006156	062700	001010		ADD	#520.,R0	:INDEX TRK ADRS
1982	006162	005301			DEC	R1	:SET UP FOR NEXT HEADER
1983	006164	002372			BGE	1\$	:COUNT SECTORS
1984	006166	000207			RTS	PC	:BRANCH IF NOT LAST SECTOR
1985							:EXIT
1986							
1987							:ROUTINE TO CHECK FOR KW11-L OR KW11-P CLOCKS
1988	006170	012737	006246	000004	ST.CLK: MOV	#STCLK1,@#ERRVEC	:SET UP VECTOR FOR P CLK
1989	006176	005037	000006		CLR	@#ERRVEC+2	:NEW PSW
1990	006202	005777	172770		TST	@\$LKCSR	:CHECK FOR KW11-P
1991	006206	013746	001202		MOV	\$LPVEC,-(SP)	:VECTOR ADDRESS
1992	006212	012776	006332	000000	MOV	#CLOCK,@(SP)	:SET UP KW11-P VECTOR
1993	006220	062716	000002		ADD	#2,(SP)	:POINT TO PSW
1994	006224	012736	000300		MOV	#PR6,@(SP)+	:PSW - PRI 6
1995	006230	012777	177777	172742	MOV	#-1,@\$LKCSB	:LOAD COUNTER BUFFER
1996	006236	012777	000135	172732	MOV	#135,@\$LKCSR	:SET CLK - CNT UP
1997	006244	000426			BR	STCLK3	
1998	006246	062706	000004		STCLK1: ADD	#4,SP	:RESTORE THE STACK POINTER
1999	006252	012737	006316	000004	MOV	#STCLK2,@#ERRVEC	:CHANGE ERROR VECTOR
2000	006260	005777	172722		TST	@\$LKS	:LOOK FOR KW11-L
2001	006264	013746	001210		MOV	\$LLVEC,-(SP)	:KW11-L VECTOR ADDRESS
2002	006270	012776	006332	000000	MOV	#CLOCK,@(SP)	:SET UP KW11-L VECTOR
2003	006276	062716	000002		ADD	#2,(SP)	:INCREMENT VECTOR ADDRESS
2004	006302	012736	000300		MOV	#PR6,@(SP)+	:PSW - PRI 6


```

2005 006306 012777 000100 172672      MOV    #100,@$LKS      ;SET KW11-L INTERRUPT ENABLE
2006 006314 000402                BR      STCLK3
2007 006316 062706 000004                STCLK2: ADD    #4,SP      ;RESTORE THE STACK POINTER
2008 006322 012737 000006 000004      STCLK3: MOV    #6,@$ERRVEC ;RESTORE THE ERROR VECTOR
2009 006330 000207                RTS      PC
2010
2011                                ;THIS CODE SERVICES A CLOCK INTERRUPT EVERY 16MS
2012
2013 006332 012746 000020      CLOCK: MOV    #16,-(SP)    ;PUT MILLISECONDS ON THE STACK
2014 006336 004737 016744                JSR      PC,RPTMR      ;GO REPORT TIME
2015 006342 000002                RTI                          ;RETURN AND CONTINUE
2016
2017                                ;ROUTINE TO INTERCEPT 'CONTROL C' TYPED DURING PARAMETER ENTRY TIME
2018
2019 006344 012706 001100      OENTER: MOV    #STACK,SP    ;INITIALIZE THE STACK
2020 006350 005737 012406      1$:   TST      TRNSWT      ;ALL ACTIVITY STOPPED ?
2021 006354 001375                BNE      1$          ;BR IF NOT
2022 006356 000005                RESET      ;CLEAR THE BUS
2023 006360 000137 003176                JMP      M1B        ;START AGAIN WITH DRIVE SELECTION
2024
2025                                ;ROUTINE TO TYPE THE PRESENT DISK ADDRESS. ENTERED BY TYPING 'CONTROL C'
2026
2027 006364 005037 177776      TYPADR: CLR      PSW          ;SET PROCESSOR TO PRIORITY 0
2028 006370 012737 006344 001300      MOV    #OENTER,CNTLC ;CHANGE 'CONTROL C' RETURN ADDRESS
2029 006376 104401 022465                TYPE    ,ADDRIS      ;'PRESENT ADDRESS IS: '
2030 006402 104401 021275                TYPE    ,C          ;'C'
2031 006406 013746 001370      MOV    FMTDPB+12,-(SP)    ;PUT THE CYLINDER ADDRESS ON THE STACK
2032 006412 004737 011716      JSR      PC,$SB2D        ;CONVERT IT TO DECIMAL
2033 006416 004737 011656      JSR      PC,$SUPRS       ;TYPE IT
2034 006422 104401 021305      TYPE    ,LINSF         ;SPACES
2035 006426 104401 021277      TYPE    ,T            ;'T'
2036 006432 005046                CLR      -(SP)         ;CLEAR THE STACK
2037 006434 113716 001367      MOV     FMTDPB+11,(SP)    ;PUT THE TRACK ADDRESS ON THE STACK
2038 006440 004737 011716      JSR      PC,$SB2D        ;CONVERT IT TO DECIMAL
2039 006444 004737 011656      JSR      PC,$SUPRS       ;TYPE IT
2040 006450 104401 001167      TYPE    ,$CRLF         ;CR-LF
2041 006454 012737 006364 001300      MOV    #TYPADR,CNTLC ;RESTORE ENTRANCE TO THIS ROUTINE
2042 006462 000002                RTI                          ;RETURN
2043
2044                                ;PARAMETER ENTRY ROUTINE
2045                                ;CALL
2046                                ;
2047                                ;   MOV    #ADR,R3      ;PARAMETER TABLE ADDRESS
2048                                ;   JSR      PC,PARENT   ;GET THE PARAMETERS
2049 006464 010346                PARENT: MOV    R3,-(SP)    ;SAVE R3
2050 006466 012703 001376      MOV    #TABLE,R3      ;PARAMETER TABLE ADDRESS
2051 006472 012337 006502      1$:   MOV    (R3)+,3$      ;ADDRESS OF PARAMETER NAME
2052 006476 001436                BEQ      9$          ;BR IF AT END OF TABLE
2053 006500 104401                TYPE    ,              ;TYPE THE PARAMETER NAME
2054 006502 000000      3$:   .WORD    0              ;ADDRESS OF PARAMETER NAME TEXT
2055 006504 012302                MOV    (R3)+,R2      ;MAXIMUM PARAMETER VALUE
2056 006506 012305                MOV    (R3)+,R5      ;ADDRESS OF PARAMETER
2057 006510 011546                MOV    (R5),-(SP)    ;CURRENT VALUE OF PARAMETER
2058 006512 104405                TYPDS      ;TYPE THE CURRENT VALUE OF THE PARAMETER
2059 006514 104401 021271      TYPE    ,SLASH        ;' / '
2060 006520 104411                RDLIN      ;READ THE KEYBOARD

```


2061	006522	012601		MOV	(SP)+,R1	:INPUT ASCII STRING ADDRESS
2062	006524	004537	006726	JSR	R5,CK.DIG	:CHECK THE DIGIT(S)
2063	006530	006472		1\$		:CARRIAGE RETURN ONLY ENTERED
2064	006532	006574		9\$		:PERIOD ONLY ENTERED
2065	006534	006550		6\$		:ILLEGAL INPUT
2066	006536	006544		5\$		:TERMINATED WITH A CARRIAGE RETURN
2067	006540	006550		6\$		:TERMINATED WITH A '...'
2068	006542	006562		7\$		:TERMINATED WITH A '...'
2069	006544	010215		5\$: MOV	R2,(R5)	:MOVE NEW VALUE TO PARAMETER LOCATION
2070	006546	000751		BR	1\$	:GET MORE PARAMETERS
2071	006550	104401	022036	6\$: TYPE	,BADENT	: 'BAD ENTRY'
2072	006554	162703	000006	SUB	#6,R3	:DECREMENT THE TABLE POINTER
2073	006560	000744		BR	1\$	:TRY AGAIN
2074	006562	010215		7\$: MOV	R2,(R5)	:NEW VALUE
2075	006564	000403		BR	9\$	:EXIT
2076	006566	012703	001376	8\$: MOV	#TABLE,R3	:RELOAD THE PARAMETER TABLE ADDRESS
2077	006572	000737		BR	1\$	:TRY AGAIN
2078	006574	012603		9\$: MOV	(SP)+,R3	:RESTORE R3
2079	006576	000207		RTS	PC	:RETURN

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

:CALL
 :
 : MOV #ADR,R1 :ADDRESS OF ASCII CHARACTER
 : JSR R5,CK.OCT :CHECK THE CHARACTER
 : RETURN1 :CHARACTER IS NOT BETWEEN 0-7
 : RETURN2 :CHARACTER IS IN R2 AS A
 : :OCTAL DIGIT

2091	006600	121127	000060	CK.OCT: CMPB	(R1),#0	:LESS THAN ZERO?
2092	006604	103407		BLO	1\$	:YES -- BRANCH
2093	006606	121127	000067	CMPB	(R1),#7	:GREATER THAN SEVEN?
2094	006612	101004		BHI	1\$	:YES -- BRANCH
2095	006614	111102		MOVB	(R1),R2	:GET THE CHARACTER
2096	006616	042702	177770	BIC	#^C7,R2	:STRIP AWAY THE ASCII
2097	006622	005725		TST	(R5)+	:ADJUST FOR RETURN
2098	006624	000205		1\$: RTS	R5	:RETURN

:THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER
 :AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.

:CALL
 :
 : MOV #ADR,R1 :ADDRESS OF ASCII CHARACTER
 : JSR R5,CK.DEC :CHECK THE CHARACTER
 : RETURN1 :NOT BETWEEN 0 AND 9
 : RETURN2 :BETWEEN 0 AND 9
 : :R2 = DIGIT

2109	006626	121127	000060	CK.DEC: CMPB	(R1),#0	:LESS THAN ZERO?
2110	006632	103407		BLO	1\$	:YES -- BRANCH
2111	006634	121127	000071	CMPB	(R1),#9	:GREATER THAN NINE?
2112	006640	101004		BHI	1\$	:YES -- BRANCH
2113	006642	111102		MOVB	(R1),R2	:GET THE CHARACTER
2114	006644	042702	000060	BIC	#0,R2	:STRIP AWAY THE ASCII
2115	006650	005725		TST	(R5)+	:ADJUST FOR RETURN
2116	006652	000205		1\$: RTS	R5	:RETURN


```

2117
2118
2119      ;THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO
2120      ;DETERMINE WHAT IT IS.
2121      ;CALL
2122      :      MOV      #ADR,R1      ;ADDRESS OF ASCII CHARACTER
2123      :      JSR      R5,CK.CHR    ;CHECK CHARACTER
2124      :      RETURN   ADR1          ;UNKNOWN CHARACTER
2125      :      RETURN   ADR2          ;CARRIAGE RETURN * (R1)=ADR+1
2126      :      RETURN   ADR3          ;COMMA * (R1)=ADR+1
2127      :      RETURN   ADR4          ;PERIOD * (R1)=ADR+1
2128      :      RETURN   ADR5          ;DIGIT BETWEEN 0 AND 7.
2129      :      RETURN   ADR6          ;DIGIT BETWEEN 8 AND 9.
2130      :      ;R2 = DIGIT * (R1)=ADR+1
2131      006654 105711      CK.CHR: TSTB      (R1)      ;'CARRIAGE RETURN'?
2132      006656 001417      BEQ      3$            ;YES -- BRANCH
2133      006660 121127 000054      CMPB      (R1),#'.    ;'COMMA'?
2134      006664 001413      BEQ      2$            ;YES -- BRANCH
2135      005666 121127 000056      CMPB      (R1),#'.    ;'PERIOD'?
2136      006672 001407      BEQ      1$            ;YES -- BRANCH
2137      006674 004537 006626      JSR      R5,CK.DEC    ;'DIGIT'?
2138      006700 000410      BR      4$            ;NO -- BRANCH
2139      006702 004537 006600      JSR      R5,CK.OCT    ;OCTAL ?
2140      006706 005725      TST      (R5)+        ;DIGIT BETWEEN 8-9
2141      006710 005725      TST      (R5)+        ;DIGIT BETWEEN 0-7
2142      006712 005725      1$: TST      (R5)+        ;PERIOD
2143      006714 005725      2$: TST      (R5)+        ;COMMA
2144      006716 005725      3$: TST      (R5)+        ;CARRIAGE RETURN
2145      006720 005201      INC      R1            ;MOVE POINTER TO NEXT CHARACTER
2146      006722 011505      4$: MOV      (R5),R5      ;UNKNOWN CHARACTER
2147      006724 000205      RTS      R5            ;RETURN
2148
2149      ;THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL
2150      ;CHARACTERS AND FORMS A DECIMAL VALUE BINARY NUMBER IN R2.
2151      ;CALL
2152      :      MOV      #ADR,R1      ;ADDRESS OF ASCII STRING
2153      :      MOV      #NUM,R2      ;MAX. MAGNITUDE OF INPUT NUMBER
2154      :      JSR      R5,CK.DIG    ;CHECK DIGITS
2155      :      RETURN   ADR1          ;'CR' ONLY ENTERED -- R2=0
2156      :      RETURN   ADR2          ;'PERIOD' ONLY ENTERED -- R2=0
2157      :      RETURN   ADR3          ;ILLEGAL CHARACTER OR INPUT TOO LARGE -- R2=?
2158      :      RETURN   ADR4          ;'CR' -- R2 = NUMBER
2159      :      RETURN   ADR5          ;'COMMA' -- R2 = NUMBER
2160      :      RETURN   ADR6          ;'PERIOD' -- R2 = NUMBER
2161
2162      006726 010446      CK.DIG: MOV      R4,-(SP)    ;SAVE R4
2163      006730 010346      MOV      R3,-(SP)    ;SAVE R3
2164      006732 010246      MOV      R2,-(SP)    ;SAVE THE MAX. SIZE ON THE STACK
2165      006734 005002      CLR      R2          ;START WITH 0
2166      006736 005003      CLR      R3
2167      006740 005004      CLR      R4
2168      006742 004537 006654      JSR      R5,CK.CHR    ;CHECK ONE CHARACTER
2169      006746 007042      6$          ;ILLEGAL CHARACTER
2170      006750 007050      9$          ;CARRIAGE RETURN
2171      006752 007042      6$          ;
2172      006754 007044      7$          ;

```



```

2173 006756 006762 1$ :DIGIT 0-7
2174 006760 006762 1$ :DIGIT 8-9
2175 006762 062705 000004 1$: ADD #4,R5 :STEP RETURN POINTER PAST 'CR' & 'PERIOD' RETURNS
2176 006766 006303 2$: ASL R3 :INPUT NUMBER *2
2177 006770 010346 MOV R3,-(SP) :SAVE *2
2178 006772 006303 ASL R3 :*4
2179 006774 006303 ASL R3 :*8
2180 006776 062603 ADD (SP)+,R3 :(*2)+(*8) = *10
2181 007000 060203 ADD R2,R3 :UPDATE THE INPUT NUMBER
2182 007002 004537 006654 JSR R5,CK.CHR :CHECK ONE CHARACTER
2183 007006 007046 8$ :ILLEGAL CHARACTER
2184 007010 007032 5$ :CARRIAGE RETURN
2185 007012 007030 4$ :
2186 007014 007022 3$ :
2187 007016 006766 2$ :DIGIT 0-7
2188 007020 006766 2$ :DIGIT 8-9
2189 007022 105711 3$: TSTB (R1) :DOES A 'CR' FOLLOW THE 'PERIOD'
2190 007024 001010 BNE 8$ :BR IF NOT
2191 007026 005724 TST (R4)+ :INCREMENT THE RETURN
2192 007030 005724 4$: TST (R4)+ :INCREMENT THE RETURN
2193 007032 005724 5$: TST (R4)+ :INCREMENT THE RETURN
2194 007034 020316 CMP R3,(SP) :CHECK THE MAGNITUDE OF THE NUMBER
2195 007036 101004 BHI 9$ :BR IF ENTERED NUMBER TOO LARGE
2196 007040 000402 BR 8$ :BYPASS INCREMENT
2197 007042 005725 6$: TST (R5)+ :INCREMENT RETURN PAST INVALID RETURN
2198 007044 005725 7$: TST (R5)+ :INCREMENT RETURN
2199 007046 060405 8$: ADD R4,R5 :SETUP RETURN POINTER
2200 007050 010302 9$: MOV R3,R2 :ENTERED VALUE
2201 007052 005726 TST (SP)+ :CLEAN MAX. SIZE OFF OF STACK
2202 007054 012603 MOV (SP)+,R3 :RESTORE R3
2203 007056 012604 MOV (SP)+,R4 :RESTORE R4
2204 007060 011505 MOV (R5),R5 :GET RETURN ADDRESS
2205 007062 000205 RTS R5 :RETURN

```

.SBTTL MACRO ROUTINES

.SBTTL ERROR HANDLER ROUTINE

```

2212 *****
2213 *THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
2214 *SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
2215 *AND GO TO TYPERR ON ERROR
2216 *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
2217 *SW15=1 HALT ON ERROR
2218 *SW13=1 INHIBIT ERROR TYPEOUTS
2219 *SW10=1 BELL ON ERROR
2220 *SW09=1 LOOP ON ERROR
2221 *CALL
2222 *
2223 * ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER
2224 $ERROR:
2225 007064 104407 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
2226 007066 010137 001274 MOV R1,DDrive ;;DRIVE ADDRESS IF DRIVE ERROR CALL
2227 007072 010337 001276 MOV R3,ATTN ;;ATTENTION REGISTER CONTENTS
2228 007076 013737 001370 001270 MOV FMTDPB+12,DS.CYL ;;CURRENT CYLINDER ADDRESS

```



```

2229 007104 113737 001367 001272      MOVB   FMTDPB+11,DS,TRK  ;REQUESTED TRACK ADDRESS
2230 007112 005737 001312              TST    RP.REG+RPBA    ;NON-ZERO BUFFER ADDRESS ?
2231 007116 001406              BEQ     7$                ;BR IF NO BUFFER ADDRESS
2232 007120 013746 001312      MOV     RP.REG+RPBA,-(SP)  ;BUFFER ADDRESS
2233 007124 162716 000002      SUB     #2,(SP)        ;DECREMENT THE ADDRESS
2234 007130 013637 001124      MOV     @ (SP)+,$GDDAT  ;GET THE BUFFER WORD WHICH DIDN'T COMPARE
2235 007134 105237 001103      7$:    INCB    $ERFLG        ;SET THE ERROR FLAG
2236 007140 001775              BEQ     7$                ;DON'T LET THE FLAG GO TO ZERO
2237 007142 013777 001102 171772      MOV     $TSTNM,@DISPLAY ;DISPLAY TEST NUMBER AND ERROR FLAG
2238 007150 032777 002000 171762      BIT     #BIT10,@SWR    ;BELL ON ERROR?
2239 007156 001402              BEQ     1$                ;NO - SKIP
2240 007160 104401 001162              TYPE   $BELL        ;RING BELL
2241 007164 005237 001112      1$:    INC     $ERTTL        ;COUNT THE NUMBER OF ERRORS
2242 007170 011637 001116      MOV     (SP),$ERRPC    ;GET ADDRESS OF ERROR INSTRUCTION
2243 007174 162737 000002 001116      SUB     #2,$ERRPC
2244 007202 117737 171710 001114      MOVB    @$ERRPC,$ITEMB ;STRIP AND SAVE THE ERROR ITEM CODE
2245 007210 032777 020000 171722      BIT     #BIT13,@SWR    ;SKIP TYPEOUT IF SET
2246 007216 001004              BNE     20$               ;SKIP TYPEOUTS
2247 007220 004737 007272      JSR     PC,TYPERR    ;GO TO USER ERROR ROUTINE
2248 007224 104401 001167              TYPE   $CRLF
2249 007230              20$:
2250 007230 005777 171704      2$:    TST     @SWR                ;HALT ON ERROR
2251 007234 100002              BPL     3$                ;SKIP IF CONTINUE
2252 007236 000000              HALT                    ;HALT ON ERROR!
2253 007240 104407              CKSWR
2254 007242 032777 001000 171670 3$:    BIT     #BIT09,@SWR    ;TEST FOR CHANGE IN SOFT-SWR
2255 007250 001402              BEQ     4$                ;LOOP ON ERROR SWITCH SET?
2256 007252 013716 001110              MOV     $LPERR,(SP)    ;BR IF NO
2257 007256 005737 001160      4$:    TST     $ESCAPE        ;FUDGE RETURN FOR LOOPING
2258 007262 001402              BEQ     5$                ;CHECK FOR AN ESCAPE ADDRESS
2259 007264 013716 001160              MOV     $ESCAPE,(SP)   ;BR IF NONE
2260 007270              5$:
2261 007270 000002              RTI                ;FUDGE RETURN ADDRESS FOR ESCAPE
2262
2263
2264
2265
2266
2267
2268

```

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

```

2269 007272 104412      TYPERR: SAVREG      ;SAVE R0-R5
2270 007274 005000      CLR     R0                ;CLEAR R0 FOR ERROR NUMBER
2271 007276 113700 001114      MOVB    $ITEMB,R0    ;ERROR NUMBER
2272 007302 005300      DEC     R0                ;FORM INDEX FOR ERROR TABLE
2273 007304 006300      ASL     R0
2274 007306 006300      ASL     R0
2275 007310 006300      ASL     R0
2276 007312 062700 001630 1$:    ADD     #$ERRTB,R0    ;FORM ADDRESS
2277 007316 012037 007332      MOV     (R0)+,2$    ;GET ERROR MESSAGE (EM) POINTER
2278 007322 001404      BEQ     3$                ;BRANCH IF THERE ISN'T ONE
2279 007324 104401 001167      TYPE     $CRLF        ;CARRIAGE RETURN - LINE FEED
2280 007330 104401      TYPE
2281 007332 000000      2$:    .WORD   0                ;"EM" POINTER GOES HERE
2282 007334 012037 007350 3$:    MOV     (R0)+,4$    ;PICK UP DATA HEADER (DH) POINTER
2283 007340 001404      BEQ     5$                ;BRANCH IF NONE
2284 007342 104401 001167      TYPE     $CRLF        ;CARRIAGE RETURN-LINE FEED

```


2285	007346	104401			TYPE		
2286	007350	000000		4\$:	.WORD	0	;'DH' POINTER GOES HERE
2287	007352	012001		5\$:	MOV	(R0)+,R1	;'PICKUP DATA TABLE (DT) POINTER
2288	007354	001460			BEQ	20\$	;'BRANCH IF NONE
2289	007356	005005			CLR	R5	;'SET INDENT SWITCH
2290	007360	012000			MOV	(R0)+,R0	;'DATA FORMAT (DF) POINTER
2291	007362	012002			MOV	(R0)+,R2	;'NUMBER OF DH'S TO TYPE
2292	007364	001451			BEQ	17\$	;'BRANCH IF DH NUMBER IS 0
2293	007366	005105			COM	R5	;'NO INDENT
2294	007370	104401	001167		TYPE	,\$CRLF	;'CARRIAGE RETURN-LINE FEED
2295	007374	112003		10\$:	MOVB	(R0)+,R3	;'NUMBER OF DATA WORDS TO TYPE
2296	007376	112004			MOVB	(R0)+,R4	;'AND HOW TO TYPE THEM
2297	007400	006004		11\$:	ROR	R4	;'OCTAL OR DECIMAL?
2298	007402	103403			BCS	12\$	;'DECIMAL--BRANCH
2299	007404	013146			MOV	@(R1)+,-(SP)	;'SAVE @(R1)+ FOR TYPEOUT
2300	007406	104402			TYPOC		;'GO TYPE--OCTAL ASCII(ALL DIGITS)
2301	007410	000402			BR	13\$	
2302	007412			12\$:			
2303	007412	013146			MOV	@(R1)+,-(SP)	;'SAVE @(R1)+ FOR TYPEOUT
2304	007414	104405			TYPDS		;'GO TYPE--DECIMAL ASCII WITH SIGN
2305	007416	005303		13\$:	DEC	R3	;'MORE NUMBERS TO TYPE?
2306	007420	001403			BEQ	14\$	;'NO--BRANCH
2307	007422	104401	021305		TYPE	,\$LINS	;'YES--TYPE SEPERATORS
2308	007426	000764			BR	11\$	;'LOOP
2309	007430	005302		14\$:	DEC	R2	;'MORE DH'S?
2310	007432	003431			BLE	20\$	;'NO--BRANCH
2311	007434	104401	001167		TYPE	,\$CRLF	;'YES--START A NEW LINE
2312	007440	005760	000002		TST	2(R0)	;'ONLY A 'DH' IN THIS REQUEST ?
2313	007444	001404			BEQ	15\$	;'BR IF YES - BYPASS THE INDENT
2314	007446	005105			COM	R5	;'INDENT?
2315	007450	001002			BNE	15\$	;'NO--BRANCH
2316	007452	104401	021305		TYPE	,\$LINS	;'YES--TYPE SPACES
2317	007456	012037	007464	15\$:	MOV	(R0)+,16\$	;'GET NEXT DH
2318	007462	104401			TYPE		;'AND TYPE IT
2319	007464	000000		16\$:	.WORD	0	;'DH POINTER GOES HERE
2320	007466	005710			TST	(R0)	;'TYPE A 'DT' ?
2321	007470	001003			BNE	21\$	;'BR IF A 'DT'
2322	007472	062700	000004		ADD	#4,R0	;'INCREMENT THE 'DF' POINTER
2323	007476	000754			BR	14\$	;'SEE IF END OF 'DF' BLOCK
2324	007500	104401	001167	21\$:	TYPE	,\$CRLF	;'CARRIAGE RETURN-LINE FEED
2325	007504	005705			TST	R5	;'INDENT?
2326	007506	001332			BNE	10\$	;'NO--BRANCH
2327	007510	104401	021305	17\$:	TYPE	,\$LINS	;'YES--TYPE SPACES
2328	007514	000727			BR	10\$	;'LOOP
2329	007516	104413		20\$:	RESREG		;'RESTORE R0-R5
2330	007520	000207			RTS	PC	;'RETURN

.SBTTL TYPE ROUTINE

2331
 2332
 2333
 2334
 2335
 2336
 2337
 2338
 2339
 2340

```

*****
*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.
*
```



```

2341      ;*CALL:
2342      ;*1) USING A TRAP INSTRUCTION
2343      ;*      TYPE      ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
2344      ;*OR
2345      ;*      TYPE
2346      ;*      MESADR
2347      ;*
2348
2349 007522 105737 001157 $TYPE: TSTB $TPFLG      ;;IS THERE A TERMINAL?
2350 007526 100002      BPL 1$      ;;BR IF YES
2351 007530 000000      HALT      ;;HALT HERE IF NO TERMINAL
2352 007532 000407      BR 3$      ;;LEAVE
2353 007534 010046 1$: MOV R0,-(SP)      ;;SAVE R0
2354 007536 017600 000002      MOV @2(SP),R0      ;;GET ADDRESS OF ASCIZ STRING
2355 007542 112046 2$: MOVB (R0)+,-(SP)      ;;PUSH CHARACTER TO BE TYPED ONTO STACK
2356 007544 001005      BNE 4$      ;;BR IF IT ISN'T THE TERMINATOR
2357 007546 005726      TST (SP)+      ;;IF TERMINATOR POP IT OFF THE STACK
2358 007550 012600 60$: MOV (SP)+,R0      ;;RESTORE R0
2359 007552 062716 000002 3$: ADD #2,(SP)      ;;ADJUST RETURN PC
2360 007556 000002      RTI      ;;RETURN
2361 007560 122716 000011 4$: CMPB #HT,(SP)      ;;BRANCH IF <HT>
2362 007564 001430      BEQ 8$
2363 007566 122716 000200      CMPB #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
2364 007572 001006      BNE 5$
2365 007574 005726      TST (SP)+      ;;POP <CR><LF> EQUIV
2366 007576 104401      TYPE      ;;TYPE A CR AND LF
2367 007600 001167 $CRLF
2368 007602 105037 007736      CLRB $CHARCNT      ;;CLEAR CHARACTER COUNT
2369 007606 000755      BR 2$      ;;GET NEXT CHARACTER
2370 007610 004737 007672 5$: JSR PC,$TYPEPC      ;;GO TYPE THIS CHARACTER
2371 007614 123726 001156 6$: CMPB $FILLC,(SP)+      ;;IS IT TIME FOR FILLER CHARS.?
2372 007620 001350      BNE 2$      ;;IF NO GO GET NEXT CHAR.
2373 007622 013746 001154      MOV $NULL,-(SP)      ;;GET # OF FILLER CHARS. NEEDED
2374      ;;AND THE NULL CHAR.
2375 007626 105366 000001 7$: DECB 1(SP)      ;;DOES A NULL NEED TO BE TYPED?
2376 007632 002770      BLT 6$      ;;BR IF NO--GO POP THE NULL OFF OF STACK
2377 007634 004737 007672      JSR PC,$TYPEPC      ;;GO TYPE A NULL
2378 007640 105337 007736      DECB $CHARCNT      ;;DO NOT COUNT AS A COUNT
2379 007644 000770      BR 7$      ;;LOOP
2380
2381      ;HORIZONTAL TAB PROCESSOR
2382
2383 007646 112716 000040 8$: MOVB #' ,(SP)      ;;REPLACE TAB WITH SPACE
2384 007652 004737 007672 9$: JSR PC,$TYPEPC      ;;TYPE A SPACE
2385 007656 132737 000007 007736      BITB #7,$CHARCNT      ;;BRANCH IF NOT AT
2386 007664 001372      BNE 9$      ;;TAB STOP
2387 007666 005726      TST (SP)+      ;;POP SPACE OFF STACK
2388 007670 000724      BR 2$      ;;GET NEXT CHARACTER
2389 007672 105777 171252 $TYPEPC: TSTB @2$TPS      ;;WAIT UNTIL PRINTER IS READY
2390 007676 100375      BPL $TYPEPC
2391 007700 116677 000002 171244      MOVB 2(SP),@2$TPB      ;;LOAD CHAR TO BE TYPED INTO DATA REG.
2392 007706 122766 000015 000002      CMPB #CR,2(SP)      ;;IS CHARACTER A CARRIAGE RETURN?
2393 007714 001003      BNE 1$      ;;BRANCH IF NO
2394 007716 105037 007736      CLRB $CHARCNT      ;;YES--CLEAR CHARACTER COUNT
2395 007722 000406      BR $TYPEPC      ;;EXIT
2396 007724 122766 000012 000002 1$: CMPB #LF,2(SP)      ;;IS CHARACTER A LINE FEED?

```



```

2397 007732 001402          BEQ      $TYPEX      ;;BRANCH IF YES
2398 007734 105227          INCB      (PC)+      ;;COUNT THE CHARACTER
2399 007736 000000          $CHARCNT: .WORD 0      ;;CHARACTER COUNT STORAGE
2400 007740 000207          $TYPEX: RTS      PC
2401
2402
2403          .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
2404
2405          ;*****
2406          ;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
2407          ;OCTAL (ASCII) NUMBER AND TYPE IT.
2408          ;$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
2409          ;CALL:
2410          ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
2411          ;*      TYPOS      ;;CALL FOR TYPEOUT
2412          ;*      .BYTE  N      ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
2413          ;*      .BYTE  M      ;;M=1 OR 0
2414          ;*                      ;;1=TYPE LEADING ZEROS
2415          ;*                      ;;0=SUPPRESS LEADING ZEROS
2416          ;*
2417          ;$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
2418          ;$TYPOS OR $TYPOC
2419          ;CALL:
2420          ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
2421          ;*      TYPON      ;;CALL FOR TYPEOUT
2422          ;*
2423          ;$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
2424          ;CALL:
2425          ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
2426          ;*      TYPOC      ;;CALL FOR TYPEOUT
2427          ;*
2428 007742 017646 000000          $TYPOS: MOV      @ (SP),-(SP)      ;;PICKUP THE MODE
2429 007746 116637 000001 010165  MOV      1(SP), $OFILL      ;;LOAD ZERO FILL SWITCH
2430 007754 112637 010167          MOV      (SP)+, $OMODE+1      ;;NUMBER OF DIGITS TO TYPE
2431 007760 062716 000002          ADD      #2, (SP)      ;;ADJUST RETURN ADDRESS
2432 007764 000406          BR      $TYPON
2433 007766 112737 000001 010165  $TYPOC: MOV      #1, $OFILL      ;;SET THE ZERO FILL SWITCH
2434 007774 112737 000006 010167  MOV      #6, $OMODE+1      ;;SET FOR SIX(6) DIGITS
2435 010002 112737 000005 010164  $TYPON: MOV      #5, $OCNT      ;;SET THE ITERATION COUNT
2436 010010 010346          MOV      R3, -(SP)      ;;SAVE R3
2437 010012 010446          MOV      R4, -(SP)      ;;SAVE R4
2438 010014 010546          MOV      R5, -(SP)      ;;SAVE R5
2439 010016 113704 010167          MOV      $OMODE+1, R4      ;;GET THE NUMBER OF DIGITS TO TYPE
2440 010022 005404          NEG      R4
2441 010024 062704 000006          ADD      #6, R4      ;;SUBTRACT IT FOR MAX. ALLOWED
2442 010030 110437 010166          MOV      R4, $OMODE      ;;SAVE IT FOR USE
2443 010034 113704 010165          MOV      $OFILL, R4      ;;GET THE ZERO FILL SWITCH
2444 010040 016605 000012          MOV      12(SP), R5      ;;PICKUP THE INPUT NUMBER
2445 010044 005003          CLR      R3      ;;CLEAR THE OUTPUT WORD
2446 010046 006105          1$:  ROL      R5      ;;ROTATE MSB INTO 'C'
2447 010050 000404          BR      3$      ;;GO DO MSB
2448 010052 006105          2$:  ROL      R5      ;;FORM THIS DIGIT
2449 010054 006105          ROL      R5
2450 010056 006105          ROL      R5
2451 010060 010503          MOV      R5, R3
2452 010062 006103          3$:  ROL      R3      ;;GET LSB OF THIS DIGIT

```


2453	010064	105337	010166		DECB	\$OMODE	::TYPE THIS DIGIT?
2454	010070	100016			BPL	7\$	::BR IF NO
2455	010072	042703	177770		BIC	#177770,R3	::GET RID OF JUNK
2456	010076	001002			BNE	4\$	::TEST FOR 0
2457	010100	005704			TST	R4	::SUPPRESS THIS 0?
2458	010102	001403			BEQ	5\$	::BR IF YES
2459	010104	005204		4\$:	INC	R4	::DON'T SUPPRESS ANYMORE 0'S
2460	010106	052703	000060		BIS	#'0,R3	::MAKE THIS DIGIT ASCII
2461	010112	052703	000040	5\$:	BIS	#',R3	::MAKE ASCII IF NOT ALREADY
2462	010116	110337	010162		MOVB	R3,8\$	::SAVE FOR TYPING
2463	010122	104401	010162		TYPE	8\$	::GO TYPE THIS DIGIT
2464	010126	105337	010164	7\$:	DECB	\$OCNT	::COUNT BY 1
2465	010132	003347			BGT	2\$	::BR IF MORE TO DO
2466	010134	002402			BLT	6\$	::BR IF DONE
2467	010136	005204			INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
2468	010140	000744			BR	2\$	::GO DO THE LAST DIGIT
2469	010142	012605		6\$:	MOV	(SP)+,R5	::RESTORE R5
2470	010144	012604			MOV	(SP)+,R4	::RESTORE R4
2471	010146	012603			MOV	(SP)+,R3	::RESTORE R3
2472	010150	016666	000002 000004		MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
2473	010156	012616			MOV	(SP)+,(SP)	
2474	010160	000002			RTI		::RETURN
2475	010162	000		8\$:	.BYTE	0	::STORAGE FOR ASCII DIGIT
2476	010163	000			.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
2477	010164	000		\$OCNT:	.BYTE	0	::OCTAL DIGIT COUNTER
2478	010165	000		\$OFILL:	.BYTE	0	::ZERO FILL SWITCH
2479	010166	000000		\$OMODE:	.WORD	0	::NUMBER OF DIGITS TO TYPE
2480							
2481							
2482							
2483							
2484							
2485							
2486							
2487							
2488							
2489							
2490							
2491							
2492							
2493	010170						
2494	010170	010046					
2495	010172	010146					
2496	010174	010246					
2497	010176	010346					
2498	010200	010546					
2499	010202	012746	020200				
2500	010206	016605	000020				
2501	010212	100004					
2502	010214	005405					
2503	010216	112766	000055 000001				
2504	010224	005000					
2505	010226	012703	010404	1\$:	CLR	R0	::ZERO THE CONSTANTS INDEX
2506	010232	112723	000040		MOV	#DBLK,R3	::SETUP THE OUTPUT POINTER
2507	010236	005002		2\$:	MOVB	#',(R3)+	::SET THE FIRST CHARACTER TO A BLANK
2508	010240	016001	010374		CLR	R2	::CLEAR THE BCD NUMBER
					MOV	\$DTBL(R0),R1	::GET THE CONSTANT

.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 R1 ::MAKE THE BINARY NUMBER POS.

MOVB #'-,1(SP) ::MAKE THE ASCII NUMBER NEG.

CLR R0 ::ZERO THE CONSTANTS INDEX

MOV #DBLK,R3 ::SETUP THE OUTPUT POINTER

MOVB #'',(R3)+ ::SET THE FIRST CHARACTER TO A BLANK

CLR R2 ::CLEAR THE BCD NUMBER

MOV \$DTBL(R0),R1 ::GET THE CONSTANT


```

2509 010244 160105      3$: SUB R1,R5      ;;FORM THIS BCD DIGIT
2510 010246 002402      BLT 4$      ;;BR IF DONE
2511 010250 005202      INC R2      ;;INCREASE THE BCD DIGIT BY 1
2512 010252 000774      BR 3$
2513 010254 060105      4$: ADD R1,R5      ;;ADD BACK THE CONSTANT
2514 010256 005702      TST R2      ;;CHECK IF BCD DIGIT=0
2515 010260 001002      BNE 5$      ;;FALL THROUGH IF 0
2516 010262 105716      TSTB (SP)      ;;STILL DOING LEADING 0'S?
2517 010264 100407      BMI 7$      ;;BR IF YES
2518 010266 106316      5$: ASLB (SP)      ;;MSD?
2519 010270 103003      BCC 6$      ;;BR IF NO
2520 010272 116663 000001 177777      MOVB 1(SP),-1(R3)      ;;YES--SET THE SIGN
2521 010300 052702 000060      6$: BIS #'0,R2      ;;MAKE THE BCD DIGIT ASCII
2522 010304 052702 000040      7$: BIS #' ,R2      ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
2523 010310 110223      MOVB R2,(R3)+      ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
2524 010312 005720      TST (R0)+      ;;JUST INCREMENTING
2525 010314 020027 000010      CMP R0,#10      ;;CHECK THE TABLE INDEX
2526 010320 002746      BLT 2$      ;;GO DO THE NEXT DIGIT
2527 010322 003002      BGT 8$      ;;GO TO EXIT
2528 010324 010502      MOV R5,R2      ;;GET THE LSD
2529 010326 000764      BR 6$      ;;GO CHANGE TO ASCII
2530 010330 105726      8$: TSTB (SP)+      ;;WAS THE LSD THE FIRST NON-ZERO?
2531 010332 100003      BPL 9$      ;;BR IF NO
2532 010334 116663 177777 177776      MOVB -1(SP),-2(R3)      ;;YES--SET THE SIGN FOR TYPING
2533 010342 105013      9$: CLRB (R3)      ;;SET THE TERMINATOR
2534 010344 012605      MOV (SP)+,R5      ;;POP STACK INTO R5
2535 010346 012603      MOV (SP)+,R3      ;;POP STACK INTO R3
2536 010350 012602      MOV (SP)+,R2      ;;POP STACK INTO R2
2537 010352 012601      MOV (SP)+,R1      ;;POP STACK INTO R1
2538 010354 012600      MOV (SP)+,R0      ;;POP STACK INTO R0
2539 010356 104401 010404      TYPE $DBLK      ;;NOW TYPE THE NUMBER
2540 010362 016666 000002 000004      MOV 2(SP),4(SP)      ;;ADJUST THE STACK
2541 010370 012616      MOV (SP)+,(SP)
2542 010372 000002      RTI      ;;RETURN TO USER
2543 010374 023420      $DTBL: 10000.
2544 010376 001750      1000.
2545 010400 000144      100.
2546 010402 000012      10.
2547 010404 000004      $DBLK: .BLKW 4
2548
2549      .SBTTL TTY INPUT ROUTINE
2550
2551      ;*****
2552      .ENABL LSB
2553 010414 000000      $TKCNT: .WORD 0      ;;NUMBER OF ITEMS IN QUEUE
2554 010416 000000      $TKQIN: .WORD 0      ;;INPUT POINTER
2555 010420 000000      $TKQOUT: .WORD 0      ;;OUTPUT POINTER
2556 010422 000007      $TKQSRT: .BLKB 7      ;;TTY KEYBOARD QUEUE
2557      $TKQEND=.
2558      .EVEN
2559
2560      ;*TK INITIALIZE ROUTINE
2561      ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
2562      ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
2563      ;
2564      ;*CALL:
  
```



```

2565      JSR      PC,$TKINT
2566      RETURN
2567
2568 010432 005037 010414      $TKINT: CLR      $TKCNT      ;;CLEAR COUNT OF ITEMS IN QUEUE
2569 010436 012737 010422 010416 MOV      #$TKQSR,$TKQIN ;;MOVE THE STARTING ADDRESS OF THE
2570 010444 013737 010416 010420 MOV      $TKQIN,$TKQOUT ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
2571 010452 012737 010502 000060 MOV      #$TKSRV,$TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
2572 010460 012737 000200 000062 MOV      #200,$TKVEC+2 ;;'BR' LEVEL 4
2573 010466 005777 170454      TST      @TKB      ;;CLEAR DONE FLAG
2574 010472 012777 000100 170444 MOV      #100,$TKS      ;;ENABLE TTY KEYBOARD INTERRUPT
2575 010500 000207      RTS      PC      ;;RETURN TO CALLER
2576
2577      ;;TK SERVICE ROUTINE
2578      ;;THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
2579      ;;BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
2580      ;;IT IN THE QUEUE.
2581      ;;IF THE CHARACTER IS A 'CONTROL-C' (^C) $TKINT IS CALLED AND
2582      ;;UPON RETURN EXIT IS MADE TO THE 'CONTROL-C' RESTART ADDRESS (@CNTLC)
2583
2584 010502 117746 170440      $TKSRV: MOVB     @TKB,-(SP)      ;;PICKUP THE CHARACTER
2585 010506 042716 177600      BIC      #^C177,(SP)      ;;STRIP THE JUNK
2586 010512 021627 000003      CMP      (SP),#3      ;;IS IT A CONTROL C?
2587 010516 001007      BNE      1$      ;;BRANCH IF NO
2588 010520 104401 011615      TYPE     , $CNTLC      ;;TYPE A CONTROL-C (^C)
2589 010524 004737 010432      JSR      PC,$TKINT      ;;INIT THE KEYBOARD
2590 010530 005726      TST      (SP)+      ;;CLEAN UP STACK
2591 010532 000177 170542      JMP      @CNTLC      ;;CONTROL C RESTART
2592 010536 021627 000007      1$:      CMP      (SP),#7      ;;IS IT A CONTROL G?
2593 010542 001004      BNE      2$      ;;BRANCH IF NO
2594 010544 022737 000176 001140      CMP      #SWREG,SWR      ;;IS SOFT-SWR SELECTED?
2595 010552 001500      BEQ      6$      ;;GO TO SWR CHANGE
2596
2597 010554      2$:      CMP      #7,$TKCNT      ;;IS THE QUEUE FULL?
2598 010554 022737 000007 010414      BNE      3$      ;;BRANCH IF NO
2599 010562 001004      TYPE     , $BELL      ;;RING THE TTY BELL
2600 010564 104401 001162      TST      (SP)+      ;;CLEAN CHARACTER OFF OF STACK
2601 010570 005726      BR      5$      ;;EXIT
2602 010572 000451      3$:      CMP      (SP),#23      ;;IS IT A CONTROL-S?
2603 010574 021627 000023      BNE      32$      ;;BRANCH IF NO
2604 010600 001021      CLR      @TKS      ;;DISABLE TTY KEYBOARD INTERRUPTS
2605 010602 005077 170336      TST      (SP)+      ;;CLEAN CHAR OFF STACK
2606 010606 005726      31$:      TSTB     @TKS      ;;WAIT FOR A CHAR
2607 010610 105777 170330      BPL      31$      ;;LOOP UNTIL ITS THERE
2608 010614 100375      MOVB     @TKB,-(SP)      ;;GET THE CHARACTER
2609 010616 117746 170324      BIC      #^C177,(SP)      ;;MAKE IT 7-BIT ASCII
2610 010622 042716 177600      CMP      (SP)+,#21      ;;IS IT A CONTROL-Q?
2611 010626 022627 000021      BNE      31$      ;;BRANCH IF NO
2612 010632 001366      MOV      #100,$TKS      ;;REENABLE TTY KEYBOARD INTERRUPTS
2613 010634 012777 000100 170302      RTI      ;;RETURN
2614 010642 000002      32$:      INC      $TKCNT      ;;COUNT THIS CHARACTER
2615 010644 005237 010414      CMP      (SP),#140      ;;IS IT UPPER CASE?
2616 010650 021627 000140      BLT      4$      ;;BRANCH IF YES
2617 010654 002405      CMP      (SP),#175      ;;IS IT A SPECIAL CHAR?
2618 010656 021627 000175      BGT      4$      ;;BRANCH IF YES
2619 010662 003002      BIC      #40,(SP)      ;;MAKE IT UPPER CASE
2620 010664 042716 000040

```



```

2621 010670 112677 177522      4$:  MOV  (SP)+,@$TKQIN  ;;AND PUT IT IN QUEUE
2622 010674 005237 010416      INC  $TKQIN      ;;UPDATE THE POINTER
2623 010700 023727 010416 010431  CMP  $TKQIN,$$TKQEND ;;GO OFF THE END?
2624 010706 001003          BNE  5$      ;;BRANCH IF NO
2625 010710 012737 010422 010416  MOV  $$TKQSRT,$TKQIN ;;RESET THE POINTER
2626 010716 000002      5$:  RTI          ;;RETURN
2627
2628 *****
2629 ;;SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
2630 ;;ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
2631 ;;SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
2632 ;;CALL WHEN OPERATING IN TTY INTERRUPT MODE.
2633 010720 022737 000176 001140  $CKSWR: CMP  #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED
2634 010726 001124          BNE  15$      ;;EXIT IF NOT
2635 010730 105777 170210          TSTB  @$TKS      ;;IS A CHAR WAITING?
2636 010734 100121          BPL  15$      ;;IF NOT, EXIT
2637 010736 117746 170204          MOV  @$TKB,-(SP) ;;YES
2638 010742 042716 177600          BIC  #^C177,(SP) ;;MAKE IT 7-BIT ASCII
2639 010746 021627 000007          CMP  (SP),#7      ;;IS IT A CONTROL-G?
2640 010752 001300          BNE  2$      ;;IF NOT, PUT IT IN THE TTY QUEUE
2641                                     ;;AND EXIT
2642
2643 *****
2644 ;;CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
2645 ;;ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
2646 ;;CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
2647 010754 123727 001134 000001  6$:  CMPB  $AUTOB,#1      ;;ARE WE RUNNING IN AUTO-MODE?
2648 010762 001674          BEQ  2$      ;;BRANCH IF YES
2649 010764 005726          TST  (SP)+      ;;CLEAR CONTROL-G OFF STACK
2650 010766 004737 010432          JSR  PC,$TKINT      ;;FLUSH THE TTY INPUT QUEUE
2651 010772 005077 170146          CLR  @$TKS      ;;DISABLE TTY KEYBOARD INTERRUPTS
2652 010776 112737 000001 001135  MOV  #1,$INTAG      ;;SET INTERRUPT MODE INDICATOR
2653
2654 011004 104401 011627          TYPE  , $CNTLG      ;;ECHO THE CONTROL-G (^G)
2655 011010 104401 011634          $GTSWR: TYPE  , $MSWR      ;;TYPE CURRENT CONTENTS
2656 011014 013746 000176          MOV  SWREG,-(SP)      ;;SAVE SWREG FOR TYPEOUT
2657 011020 104402          TYPOC      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2658 011022 104401 011645          TYPE  , $MNEW      ;;PROMPT FOR NEW SWR
2659 011026 005046          19$:  CLR  -(SP)      ;;CLEAR COUNTER
2660 011030 005046          CLR  -(SP)      ;;THE NEW SWR
2661 011032 105777 170106          7$:  TSTB  @$TKS      ;;CHAR THERE?
2662 011036 100375          BPL  7$      ;;IF NOT TRY AGAIN
2663
2664 011040 117746 170102          MOV  @$TKB,-(SP)      ;;PICK UP CHAR
2665 011044 042716 177600          BIC  #^C177,(SP)      ;;MAKE IT 7-BIT ASCII
2666
2667 011050 021627 000003          CMP  (SP),#3      ;;IS IT A CONTROL-C?
2668 011054 001015          BNE  9$      ;;BRANCH IF NOT
2669 011056 104401 011615          TYPE  , $CNTLC      ;;YES, ECHO CONTROL-C (^C)
2670 011062 062706 000006          ADD  #6,SP      ;;CLEAN UP STACK
2671 011066 123727 001135 000001  CMPB  $INTAG,#1      ;;REENABLE TTY KEYBOARD INTERRUPTS?
2672 011074 001003          BNE  8$      ;;BRANCH IF NO
2673 011076 012777 000100 170040  MOV  #100,$TKS      ;;ALLOW TTY KEYBOARD INTERRUPTS
2674 011104 000177 170170          8$:  JMP  @CNTLC      ;;CONTROL-C RESTART
2675
2676

```



```

2677 011110 021627 000025      9$:    CMP      (SP),#25      ;;IS IT A CONTROL-U?
2678 011114 001005              BNE      10$      ;;BRANCH IF NOT
2679 011116 104401 011622      TYPE    ,SCN$LU      ;;YES, ECHO CONTROL-U (^U)
2680 011122 062706 000006      20$:    ADD      #6,SP      ;;IGNORE PREVIOUS INPUT
2681 011126 000737              BR       19$      ;;LET'S TRY IT AGAIN
2682
2683
2684 011130 021627 000015      10$:    CMP      (SP),#15      ;;IS IT A <CR>?
2685 011134 001022              BNE      16$      ;;BRANCH IF NO
2686 011136 005766 000004      TST      4(SP)      ;;YES, IS IT THE FIRST CHAR?
2687 011142 001403              BEQ      11$      ;;BRANCH IF YES
2688 011144 016677 000002 167766      MOV      2(SP),@SWR      ;;SAVE NEW SWR
2689 011152 062706 000006      11$:    ADD      #6,SP      ;;CLEAR UP STACK
2690 011156 104401 001167      14$:    TYPE    ,SCRLF      ;;ECHO <CR> AND <LF>
2691 011162 123727 001135 000001      CMPB    $INTAG,#1      ;;RE-ENABLE TTY KBD INTERRUPTS?
2692 011170 001003              BNE      15$      ;;BRANCH IF NOT
2693 011172 012777 000100 167744      MOV      #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
2694 011200 000002              15$:    RTI              ;;RETURN
2695 011202 004737 007672      16$:    JSR      PC,$TYPEC      ;;ECHO CHAR
2696 011206 021627 000060      CMP      (SP),#60      ;;CHAR < 0?
2697 011212 002420              BLT      18$      ;;BRANCH IF YES
2698 011214 021627 000067      CMP      (SP),#67      ;;CHAR > 7?
2699 011220 003015              BGT      18$      ;;BRANCH IF YES
2700 011222 042726 000060      BIC      #60,(SP)+      ;;STRIP-OFF ASCII
2701 011226 005766 000002      TST      2(SP)      ;;IS THIS THE FIRST CHAR
2702 011232 001403              BEQ      17$      ;;BRANCH IF YES
2703 011234 006316              ASL      (SP)      ;;NO, SHIFT PRESENT
2704 011236 006316              ASL      (SP)      ;;CHAR OVER TO MAKE
2705 011240 006316              ASL      (SP)      ;;ROOM FOR NEW ONE.
2706 011242 005266 000002      17$:    INC      2(SP)      ;;KEEP COUNT OF CHAR
2707 011246 056616 177776      BIS      -2(SP),(SP)      ;;SET IN NEW CHAR
2708 011252 000667              BR       7$      ;;GET THE NEXT ONE
2709 011254 104401 001166      18$:    TYPE    ,SQUES      ;;TYPE ?<CR><LF>
2710 011260 000720              BR       20$      ;;SIMULATE CONTROL-U
2711      .DSABL    LSB
2712
2713
2714
2715      ;;*****
2716      ;;THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
2717      ;;CALL:
2718      ;;*      RDCHR      ;;GET A CHARACTER FROM THE QUEUE
2719      ;;*      RETURN HERE  ;;CHARACTER IS ON THE STACK
2720      ;;*      ;;WITH PARITY BIT STRIPPED OFF
2721      ;;
2722 011262 011646 000004 000002 $RDCHR: MOV      (SP),-(SP)      ;;PUSH DOWN THE PC AND
2723 011264 016666 000004      MOV      4(SP),2(SP)      ;;THE PS
2724 011272 005066 000004      CLR      4(SP)      ;;GET READY FOR A CHARACTER
2725 011276 005046 011306      CLR      -(SP)      ;;PUT NEW PS ON STACK
2726 011300 012746 000002      MOV      #64$,-(SP)      ;;PUT NEW PC ON STACK
2727 011304 000002      RTI              ;;POP NEW PC AND PS
2728 011306
2729 011306 005737 010414      64$:    TST      $TKCNT      ;;WAIT ON A CHARACTER
2730 011312 001775              1$:    BEQ      1$      ;;
2731 011314 005337 010414      DEC      $TKCNT      ;;DECREMENT THE COUNTER
2732 011320 117766 177074 000004      MOVB    @$TKQOUT,4(SP)      ;;GET ONE CHARACTER
  
```



```

2733 011326 005237 010420      INC      $TKQOUT      ;;UPDATE THE POINTER
2734 011332 023727 010420 010431  CMP      $TKQOUT,$$TKQEND ;;DID IT GO OFF OF THE END?
2735 011340 001003      BNE      2$          ;;BRANCH IF NO
2736 011342 012737 010422 010420  MOV      $$TKQSRT,$TKQOUT ;;RESET THE POINTER
2737 011350 000002      RTI          ;;RETURN
2738                                     ;*****
2739                                     ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
2740                                     ;*CALL:
2741                                     ;*      RDLIN          ;;INPUT A STRING FROM THE TTY
2742                                     ;*      RETURN HERE      ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
2743                                     ;*                                     ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
2744
2745 011352 010346      $RDLIN: MOV      R3,-(SP)      ;;SAVE R3
2746 011354 005046      CLR      -(SP)          ;;CLEAR THE RUBOUT KEY
2747 011356 012703 011606 1$:      MOV      $$TTYIN,R3      ;;GET ADDRESS
2748 011362 022703 011615 2$:      CMP      $$TTYIN+7,R3      ;;BUFFER FULL?
2749 011366 101456      BLOS      4$          ;;BR IF YES
2750 011370 104410      RDCHR          ;;GO READ ONE CHARACTER FROM THE TTY
2751 011372 112613      MOV      (SP)+,(R3)      ;;GET CHARACTER
2752 011374 122713 000177 10$:      CMP      #177,(R3)      ;;IS IT A RUBOUT
2753 011400 001022      BNE      5$          ;;BR IF NO
2754 011402 005716      TST      (SP)          ;;IS THIS THE FIRST RUBOUT?
2755 011404 001007      BNE      6$          ;;BR IF NO
2756 011406 112737 000134 011604  MOV      #'\\,9$      ;;TYPE A BACK SLASH
2757 011414 104401 011604      TYPE      ,9$
2758 011420 012716 177777      MOV      #-1,(SP)      ;;SET THE RUBOUT KEY
2759 011424 005303 6$:      DEC      R3          ;;BACKUP BY ONE
2760 011426 020327 011606      CMP      R3,$$TTYIN      ;;STACK EMPTY?
2761 011432 103434      BLO      4$          ;;BR IF YES
2762 011434 111337 011604      MOV      (R3),9$      ;;SETUP TO TYPEOUT THE DELETED CHAR.
2763 011440 104401 011604      TYPE      ,9$
2764 011444 000746      BR      2$          ;;GO TYPE
2765 011446 005716 5$:      TST      (SP)          ;;GO READ ANOTHER CHAR.
2766 011450 001406      BEQ      7$          ;;RUBOUT KEY SET?
2767 011452 112737 000134 011604  MOV      #'\\,9$      ;;TYPE A BACK SLASH
2768 011460 104401 011604      TYPE      ,9$
2769 011464 005016      CLR      (SP)          ;;CLEAR THE RUBOUT KEY
2770 011466 122713 000025 7$:      CMP      #25,(R3)      ;;IS CHARACTER A CTRL U?
2771 011472 001003      BNE      8$          ;;BR IF NO
2772 011474 104401 011622      TYPE      ,SCNTLU      ;;TYPE A CONTROL 'U'
2773 011500 000726      BR      1$          ;;GO START OVER
2774 011502 122713 000022 8$:      CMP      #22,(R3)      ;;IS CHARACTER A '^R'?
2775 011506 001011      BNE      3$          ;;BRANCH IF NO
2776 011510 105013      CLRB      (R3)          ;;CLEAR THE CHARACTER
2777 011512 104401 001167      TYPE      ,SCRLF      ;;TYPE A 'CR' & 'LF'
2778 011516 104401 011606      TYPE      ,$$TTYIN      ;;TYPE THE INPUT STRING
2779 011522 000717      BR      2$          ;;GO PICKUP ANOTHER CHACTER
2780 011524 104401 001166 4$:      TYPE      ,SQUES      ;;TYPE A '?'
2781 011530 000712      BR      1$          ;;CLEAR THE BUFFER AND LOOP
2782 011532 111337 011604 3$:      MOV      (R3),9$      ;;ECHO THE CHARACTER
2783 011536 104401 011604      TYPE      ,9$
2784 011542 122723 000015      CMP      #15,(R3)+      ;;CHECK FOR RETURN
2785 011546 001305      BNE      2$          ;;LOOP IF NOT RETURN
2786 011550 105063 177777      CLRB      -1(R3)      ;;CLEAR RETURN (THE 15)
2787 011554 104401 001170      TYPE      ,SLF      ;;TYPE A LINE FEED
2788 011560 005726      TST      (SP)+      ;;CLEAN RUBOUT KEY FROM THE STACK

```



```

2789 011562 012603          MOV      (SP)+,R3          ;;RESTORE R3
2790 011564 011646          MOV      (SP),-(SP)        ;;ADJUST THE STACK AND PUT ADDRESS OF THE
2791 011566 016666 000004 000002 MOV      4(SP),2(SP)        ;; FIRST ASCII CHARACTER ON IT
2792 011574 012766 011606 000004 MOV      #STTYIN,4(SP)
2793 011602 000002          RTI                          ;;RETURN
2794 011604 000          9$: .BYTE 0          ;;STORAGE FOR ASCII CHAR. TO TYPE
2795 011605 000          .BYTE 0          ;;TERMINATOR
2796 011606 000007 $TTYIN: .BLKB 7          ;;RESERVE 7 BYTES FOR TTY INPUT
2797 011615 136 006503 000012 $CNTLC: .ASCIZ /^C/<15><12>          ;;CONTROL 'C'
2798 011622 052536 005015 000 $CNTLU: .ASCIZ /^U/<15><12>          ;;CONTROL 'U'
2799 011627 136 006507 000012 $CNTLG: .ASCIZ /^G/<15><12>          ;;CONTROL 'G'
2800 011634 005015 053523 020122 $MSWR: .ASCIZ <15><12>/SWR = /
2801 011642 020075 000
2802 011645 040 047040 053505 $MNEW: .ASCIZ / NEW = /
2803 011652 036440 000040
2804
2805 .SBTTL TYPE NUMERICAL ASCIZ STRING SUPPRESS LEADING ZEROS
2806
2807 ;*****
2808 ;*THIS ROUTINE IS USED TO TYPE AN ASCIZ NUMBER SUPPRESSING THE
2809 ;*LEADING NUMBERS.
2810 ;*CALL
2811 ;*      MOV      #NUMADR, -(SP)          ;;FIRST ADDRESS OF ASCIZ STRING
2812 ;*      JSR      PC,@#$SUPRS
2813
2814 $SUPRS: MOV      R0, -(SP)          ;;SAVE R0
2815 011656 010046          MOV      4(SP),R0          ;;PICKUP THE POINTER
2816 011660 016600 000004 1$: TSTB      (R0)          ;;TERMINATOR?
2817 011664 105710          BEQ      2$          ;;BR IF YES
2818 011666 001403          CMPB      #'0',(R0)+          ;;IS THIS AN ASCII '0' ?
2819 011670 122720 000060          BEQ      1$          ;;BR IF YES
2820 011674 001773          DEC      R0          ;;BACKUP BY '1'
2821 011676 005300          MOV      R0,3$          ;;SAVE FOR TYPING
2822 011700 010037 011706 2$: TYPE          ;;GO TYPE
2823 011704 104401          .WORD 0          ;;ASCIZ POINTER GOES HERE
2824 011706 000000          MOV      (SP)+,R0          ;;RESTORE R0
2825 011710 012600          MOV      (SP)+,(SP)          ;;RESTORE THE STACK
2826 011712 012616          RTS      PC          ;;RETURN
2827 011714 000207
2828
2829 .SBTTL SINGLE LENGTH BINARY TO DECIMAL ASCIZ ROUTINE
2830
2831 ;*****
2832 ;*THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
2833 ;*UNSIGNED DECIMAL ASCIZ NUMBER.
2834 ;*CALL
2835 ;*      MOV      NUMBER, -(SP)          ;;PUT BINARY NUMBER ON THE STACK
2836 ;*      JSR      PC,@#$SB2D          ;;CALL
2837 ;*      RETURN          ;;ADDRESS OF THE 1ST ASCIZ CHAR.IS ON THE STACK
2838
2839 $SB2D: MOV      2(SP),1$          ;;SAVE BINARY NUMBER
2840 011716 016637 000002 011746 MOV      #1$, -(SP)          ;;SET POINTER
2841 011724 012746 011746          JSR      PC,@#$DB2D          ;;CALL DOUBLE LENGTH CONVERT
2842 011730 004737 011752          ADD      #5,(SP)          ;;ONLY ALLOW FIVE CHARACTERS
2843 011734 062716 000005          MOV      (SP)+,2(SP)          ;;PICKUP POINTER
2844 011740 012666 000002

```



```

2845 011744 000207          RTS      PC          ;;RETURN
2846 011746 000000 000000 1$:      .WORD    0,0
2847
2848      .SBTTL  DOUBLE LENGTH BINARY TO DECIMAL ASCIIZ CONVERT ROUTINE
2849
2850      ;*****
2851      ;*THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
2852      ;*DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
2853      ;*POSITIVE.
2854      ;*CALL
2855      ;*      MOV      #PNTR, -(SP)      ;;POINTER TO LOW WORD OF BINARY NUMBER
2856      ;*      JSR      PC, @#$DB2D
2857      ;*      RETURN
2858      ;*      ;;THE FIRST ADDRESS OF ASCIIZ
2859      ;*      ;;IS ON THE STACK
2860
2861 011752 104412          $DB2D:  SAVREG      ;;SAVE REGISTERS
2862 011754 016602 000002      MOV      2(SP), R2      ;;PICKUP THE DATA POINTER
2863 011760 012700 012132      MOV      #$DECVL, R0      ;;GET ADDRESS OF '$DECVL' STRING
2864 011764 010066 000002      MOV      R0, 2(SP)      ;;PUT ADDRESS OF ASCIIZ STRING ON STACK
2865 011770 012201          MOV      (R2)+, R1      ;;PICKUP THE BINARY NUMBER
2866 011772 012202          MOV      (R2)+, R2
2867 011774 012737 000012 012050      MOV      #10, 4$      ;;SET UP TO DO 10 CONVERSIONS
2868 012002 012704 012062      MOV      $TNPWR, R4      ;;ADDRESS OF TEN POWER
2869 012006 012705 012064      MOV      $TNPWR+2, R5
2870 012012 005003          1$:      CLR      R3      ;;CLEAR PARTIAL
2871 012014 161401          2$:      SUB      (R4), R1      ;;SUBTRACT TEN POWER
2872 012016 005602          SBC      R2
2873 012020 161502          SUB      (R5), R2
2874 012022 002402          BLT      3$      ;;BR IF TEN POWER TO LARGE
2875 012024 005203          INC      R3      ;;ADD 1 TO PARTIAL
2876 012026 000772          BR      2$      ;;LOOP
2877 012030 062401          3$:      ADD      (R4)+, R1      ;;RESTORE SUBTRACTED VALUE
2878 012032 005502          ADC      R2
2879 012034 062402          ADD      (R4)+, R2
2880 012036 022525          CMP      (R5)+, (R5)+      ;;MOVE TO NEXT TEN POWER
2881 012040 052703 000060      BIS      #'0, R3      ;;CHANGE PARTIAL TO ASCII
2882 012044 110320          MOVB     R3, (R0)+      ;;SAVE IT
2883 012046 005327          DEC      (PC)+      ;;DONE?
2884 012050 000000          4$:      .WORD    0
2885 012052 001357          BNE      1$      ;;BR IF NO
2886 012054 105020          CLRB     (R0)+      ;;TERMINATOR
2887 012056 104413          RESREG
2888 012060 000207          RTS      PC      ;;RESTORE REGISTERS
2889 012062 145000          $TNPWR: 145000      ;;RETURN
2890 012064 035632          35632      ;;1.0E09
2891 012066 160400          160400      ;;1.0E08
2892 012070 002765          2765      ;;1.0E07
2893 012072 113200          113200      ;;1.0E06
2894 012074 000230          230      ;;1.0E05
2895 012076 041100          041100      ;;1.0E04
2896 012100 000017          17
2897 012102 103240          103240
2898 012104 000001          1
2899 012106 023420          23420
2900 012110 000000          0

```



```

2901 012112 001750          1750          ;;1.0E03
2902 012114 000000          0              ;;
2903 012116 000144          144           ;;1.0E02
2904 012120 000000          0              ;;
2905 012122 000012          12            ;;1.0E01
2906 012124 000000          0              ;;
2907 012126 000001          1             ;;1.0E00
2908 012130 000000          0              ;;
2909 012132 000014          $DECVL: .BLKB 12.      ;;RESERVE STORAGE FOR ASCII STRING
2910
2911          .SBTTL SAVE AND RESTORE R0-R5 ROUTINES
2912
2913          ;*****
2914          ;*SAVE R0-R5
2915          ;*CALL:
2916          ;* SAVREG
2917          ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
2918          ;*
2919          ;*TOP---(+16)
2920          ;* +2---(+18)
2921          ;* +4---R5
2922          ;* +6---R4
2923          ;* +8---R3
2924          ;*+10---R2
2925          ;*+12---R1
2926          ;*+14---R0
2927
2928          $SAVREG:
2929          MOV R0,-(SP)          ;;PUSH R0 ON STACK
2930          MOV R1,-(SP)          ;;PUSH R1 ON STACK
2931          MOV R2,-(SP)          ;;PUSH R2 ON STACK
2932          MOV R3,-(SP)          ;;PUSH R3 ON STACK
2933          MOV R4,-(SP)          ;;PUSH R4 ON STACK
2934          MOV R5,-(SP)          ;;PUSH R5 ON STACK
2935          MOV 22(SP),-(SP)      ;;SAVE PS OF MAIN FLOW
2936          MOV 22(SP),-(SP)      ;;SAVE PC OF MAIN FLOW
2937          MOV 22(SP),-(SP)      ;;SAVE PS OF CALL
2938          MOV 22(SP),-(SP)      ;;SAVE PC OF CALL
2939          RTI
2940
2941          ;*RESTORE R0-R5
2942          ;*CALL:
2943          ;* RESREG
2944          $RESREG:
2945          MOV (SP)+,22(SP)      ;;RESTORE PC OF CALL
2946          MOV (SP)+,22(SP)      ;;RESTORE PS OF CALL
2947          MOV (SP)+,22(SP)      ;;RESTORE PC OF MAIN FLOW
2948          MOV (SP)+,22(SP)      ;;RESTORE PS OF MAIN FLOW
2949          MOV (SP)+,R5          ;;POP STACK INTO R5
2950          MOV (SP)+,R4          ;;POP STACK INTO R4
2951          MOV (SP)+,R3          ;;POP STACK INTO R3
2952          MOV (SP)+,R2          ;;POP STACK INTO R2
2953          MOV (SP)+,R1          ;;POP STACK INTO R1
2954          MOV (SP)+,R0          ;;POP STACK INTO R0
2955          RTI
2956
2928 012146
2929 012146 010046
2930 012150 010146
2931 012152 010246
2932 012154 010346
2933 012156 010446
2934 012160 010546
2935 012162 016646 000022
2936 012166 016646 000022
2937 012172 016646 000022
2938 012176 016646 000022
2939 012202 000002
2944 012204
2945 012204 012666 000022
2946 012210 012666 000022
2947 012214 012666 000022
2948 012220 012666 000022
2949 012224 012605
2950 012226 012604
2951 012230 012603
2952 012232 012602
2953 012234 012601
2954 012236 012600
2955 012240 000002
  
```


2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012

.SBTTL TRAP DECODER

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

\$TRAP:	MOV	R0,-(SP)	::SAVE R0
	MOV	2(SP),R0	::GET TRAP ADDRESS
	TST	-(R0)	::BACKUP BY 2
	MOVB	(R0),R0	::GET RIGHT BYTE OF TRAP
	ASL	R0	::POSITION FOR INDEXING
	MOV	\$TRPAD(R0),R0	::INDEX TO TABLE
	RTS	R0	::GO TO ROUTINE

::THIS IS USE TO HANDLE THE 'GETPRI' MACRO

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

.SBTTL TRAP TABLE

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

	ROUTINE	
\$TRPAD:	.WORD \$TRAP2	
	\$TYPE ::CALL=TYPE	TRAP+1(104401) TTY TYPEOUT ROUTINE
	\$TYPOC ::CALL=TYPOC	TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
	\$TYPOS ::CALL=TYPOS	TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
	\$TYPON ::CALL=TYPON	TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
	\$TYPDS ::CALL=TYPDS	TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
	\$GTSWR ::CALL=GTSWR	TRAP+6(104406) GET SOFT-SWR SETTING
	\$CKSWR ::CALL=CKSWR	TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
	\$RDCHR ::CALL=RDCHR	TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
	\$RDLIN ::CALL=RDLIN	TRAP+11(104411) TTY TYPEIN STRING ROUTINE
	\$SAVREG ::CALL=SAVREG	TRAP+12(104412) SAVE R0-R5 ROUTINE
	\$RESREG ::CALL=RESREG	TRAP+13(104413) RESTORE R0-R5 ROUTINE

.SBTTL SINGLE/DUAL PORT RH11/RP04/5/6 DRIVER (REV 1.0)

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

```

3013 ;STORAGE FOR RPDS1, RPER1, RPER2, AND RPER3 ON AN ERROR '2'
3014 ;RPERRS = RPDS1
3015 ;RPERRS+2 = RPER1
3016 ;RPERRS+4 = RPER2
3017 ;RPERRS+6 = RPER3
3018
3019 012326 000000 000000 000000 RPERRS: .WORD 0,0,0,0
3020 012334 000000
3021
3022 ;TABLE OF DRIVE ACTIVE INDICATORS (DRVACT=8 BYTES)
3023 ;DRVACT=0 IF DRIVE IS IDLE
3024 ;DRVACT>0 IF DRIVE IS ACTIVE WITH A COMMAND
3025 ;DRVACT<0 IF DRIVE IS ACTIVE WITH AN ERROR RECOVERY OPERATION
3026
3027 012336 000 DRVACT: .BYTE 0 ;DRIVE 0
3028 012337 000 .BYTE 0 ;DRIVE 1
3029 012340 000 .BYTE 0 ;DRIVE 2
3030 012341 000 .BYTE 0 ;DRIVE 3
3031 012342 000 .BYTE 0 ;DRIVE 4
3032 012343 000 .BYTE 0 ;DRIVE 5
3033 012344 000 .BYTE 0 ;DRIVE 6
3034 012345 000 .BYTE 0 ;DRIVE 7
3035
3036 ;TABLE OF DRIVE STATUS INDICATORS (DRVSTA=8 BYTES)
3037 ;DRVSTA=0 IF DRIVE IS OFFLINE OR NONEXISTENT
3038 ;DRVSTA>0 IF DRIVE IS ONLINE
3039 ;DRVSTA<0 IF DRIVE IS UNSAFE
3040
3041 012346 000 DRVSTA: .BYTE 0 ;DRIVE 0
3042 012347 000 .BYTE 0 ;DRIVE 1
3043 012350 000 .BYTE 0 ;DRIVE 2
3044 012351 000 .BYTE 0 ;DRIVE 3
3045 012352 000 .BYTE 0 ;DRIVE 4
3046 012353 000 .BYTE 0 ;DRIVE 5
3047 012354 000 .BYTE 0 ;DRIVE 6
3048 012355 000 .BYTE 0 ;DRIVE 7
3049
3050 ;TABLE OF DRIVE TYPES (DRVTYPE=8 BYTES)
3051 ;DRVTYPE=0 IF DRIVE IS NONEXISTENT (DRVSTA=0, ALSO)
3052 ;DRVTYPE=1 IF DRIVE IS RP04
3053 ;DRVTYPE=2 IF DRIVE IS RP05
3054 ;DRVTYPE=4 IF DRIVE IS RP06
3055 ;DRVTYPE=-1 IF NOT RP04/5/6
3056
3057 012356 000 DRVTYPE: .BYTE 0 ;DRIVE 0
3058 012357 000 .BYTE 0 ;DRIVE 1
3059 012360 000 .BYTE 0 ;DRIVE 2
3060 012361 000 .BYTE 0 ;DRIVE 3
3061 012362 000 .BYTE 0 ;DRIVE 4
3062 012363 000 .BYTE 0 ;DRIVE 5
3063 012364 000 .BYTE 0 ;DRIVE 6
3064 012365 000 .BYTE 0 ;DRIVE 7
3065
3066 ;TABLE OF DUAL PORT INITIALIZATION INDICATORS
3067 ;DPINT=0 IF INITIALIZATION IS NOT ACTIVE ON THE DRIVE
3068 ;DPINT<0 IF INITIALIZATION IS IN PROGRESS

```



```

3069
3070 012366 000 DPINT: .BYTE 0 ;DRIVE 0
3071 012367 000 .BYTE 0 ;DRIVE 1
3072 012370 000 .BYTE 0 ;DRIVE 2
3073 012371 000 .BYTE 0 ;DRIVE 3
3074 012372 000 .BYTE 0 ;DRIVE 4
3075 012373 000 .BYTE 0 ;DRIVE 5
3076 012374 000 .BYTE 0 ;DRIVE 6
3077 012375 000 .BYTE 0 ;DRIVE 7
3078
3079 ;TABLE OF PENDING DUAL PORT REQUESTS
3080 ;DPRQS=0 IF THAT A DUAL PORT REQUEST IS NOT PENDING FOR THAT DRIVE
3081 ;DPRQS<0 IF THAT A DUAL PORT REQUEST IS PENDING FOR THAT DRIVE
3082
3083 012376 000 DPRQS: .BYTE 0 ;DRIVE 0
3084 012377 000 .BYTE 0 ;DRIVE 1
3085 012400 000 .BYTE 0 ;DRIVE 2
3086 012401 000 .BYTE 0 ;DRIVE 3
3087 012402 000 .BYTE 0 ;DRIVE 4
3088 012403 000 .BYTE 0 ;DRIVE 5
3089 012404 000 .BYTE 0 ;DRIVE 6
3090 012405 000 .BYTE 0 ;DRIVE 7
3091
3092 ;TRANSFER WAIT FLAG (TRNSWT=1 WORD)
3093 ;THIS IS A ONE WORD QUEUE. IT WILL CONTAIN THE ADDRESS OF
3094 ;'DPB' OF THE I/O OPERATION.
3095
3096 012406 000000 TRNSWT: .WORD 0
3097
3098 ;SEARCH WAIT KEYS (SRCHWT=1 WORD)
3099 ;THIS IS A ONE WORD QUEUE THAT WILL CONTAIN A KEY FOR EACH OF
3100 ;THE DRIVES THAT ARE PERFORMING A SEARCH COMMAND FOR THE I/O
3101 ;REQUEST THAT IS AT THE TOP OF THEIR REQUEST QUEUE.
3102 ;EACH DRIVE IS ASSIGNED ONE BIT, STARTING AT BIT00 FOR DRIVE 0.
3103
3104 012410 000000 SRCHWT: .WORD 0
3105
3106 ;RP04/5/6 DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
3107 ;ACTDRV=0 IF DRIVER IS INACTIVE
3108 ;ACTDRV>0 IF DRIVER IS ACTIVE
3109
3110 012412 000 ACTDRV: .BYTE 0
3111
3112 ;SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)
3113 ;ACTSTR=0 IF SOFTWARE TIMER ROUTINE IS INACTIVE
3114 ;ACTSTR>0 IF SOFTWARE TIMER ROUTINE IS ACTIVE
3115
3116 012413 000 ACTSTR: .BYTE 0
3117
3118 ;UNLOAD FLAG (ULDFLG=8 BYTES)
3119 ;ULDFLG=0 IF NO UNLOAD COMMAND
3120 ;ULDFLG>0 IF UNLOAD COMMAND IN PROGRESS
3121 ;ULDFLG<0 IF UNLOAD COMMAND IN WAIT QUEUE
3122
3123 012414 000 ULDFLG: .BYTE 0 ;DRIVE 0
3124

```



```

3125 012415 000 .BYTE 0 ;DRIVE 1
3126 012416 000 .BYTE 0 ;DRIVE 2
3127 012417 000 .BYTE 0 ;DRIVE 3
3128 012420 000 .BYTE 0 ;DRIVE 4
3129 012421 000 .BYTE 0 ;DRIVE 5
3130 012422 000 .BYTE 0 ;DRIVE 6
3131 012423 000 .BYTE 0 ;DRIVE 7
3132
3133 ;LOOK AHEAD COUNT (LACNT=8 BYTES)
3134 ;LACNT WILL INDICATE THE NUMBER OF LOOK AHEADS PERFORMED
3135
3136 012424 000 LACNT: .BYTE 0 ;DRIVE 0
3137 012425 000 .BYTE 0 ;DRIVE 1
3138 012426 000 .BYTE 0 ;DRIVE 2
3139 012427 000 .BYTE 0 ;DRIVE 3
3140 012430 000 .BYTE 0 ;DRIVE 4
3141 012431 000 .BYTE 0 ;DRIVE 5
3142 012432 000 .BYTE 0 ;DRIVE 6
3143 012433 000 .BYTE 0 ;DRIVE 7
3144
3145 ;SAVE REGISTERS FLAG (SAVEFG =1 WORD)
3146 ;SAVEFG <0 IF SAVE THE RH11/RP04/5/6 REGISTERS WHEN THE
3147 ;OPERATION IS COMPLETED AS PER (DPB+14).
3148 ;SAVEFG=0 IF SAVE THE RH11/RP04/5/6 REGISTERS, AS PER
3149 ;(DPB+14), AFTER AN ERROR.
3150
3151 012434 000000 SAVEFG: .WORD 0
3152
3153 ;SEEK FLAG (SEEKFG=1 WORD)
3154 ;SEEKFG=0 IF WHEN THE DISK ADDRESS ISN'T IN THE WINDOW
3155 ;FOR A DATA TRANSFER START A SEARCH COMMAND
3156 ;SEEKFG<0 IF DATA TRANSFER WILL DO IMPLIED SEEKS,
3157 ;DISREGARD THE WINDOW
3158
3159 012436 000000 SEEKFG: .WORD 0
3160
3161 ;TIMEOUT TABLE (TIMER=8 WORDS)
3162 ;THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION
3163
3164 012440 177777 TIMER: .WORD -1 ;DRIVE 0
3165 012442 177777 .WORD -1 ;DRIVE 1
3166 012444 177777 .WORD -1 ;DRIVE 2
3167 012446 177777 .WORD -1 ;DRIVE 3
3168 012450 177777 .WORD -1 ;DRIVE 4
3169 012452 177777 .WORD -1 ;DRIVE 5
3170 012454 177777 .WORD -1 ;DRIVE 6
3171 012456 177777 .WORD -1 ;DRIVE 7
3172
3173 ;DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)
3174 ;DTUW<0 IF NO DATA TRANSFER UNDERWAY
3175 ;DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE N
3176
3177 012460 177777 DTUW: .WORD -1
3178
3179 ;ATTENTION BITS TABLE (ATABIT=8 BYTES)
3180 ;THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES

```



```

3181                                     ;ATTENTION BIT
3182
3183 012462      001      ATABIT: .BYTE 1      ;DRIVE 0
3184 012463      002      .BYTE 2      ;DRIVE 1
3185 012464      004      .BYTE 4      ;DRIVE 2
3186 012465      010      .BYTE 10     ;DRIVE 3
3187 012466      020      .BYTE 20     ;DRIVE 4
3188 012467      040      .BYTE 40     ;DRIVE 5
3189 012470      100      .BYTE 100    ;DRIVE 6
3190 012471      200      .BYTE 200    ;DRIVE 7
3191
3192                                     ;RP04/5/6 TO RH11 'MASSBUS CONTROL BUS PARITY ERRORS' (MCPE) ALLOWED BEFORE
3193                                     ;CALLING IT FATAL (MCPEMX=1 WORD)
3194
3195 012472      000003      MCPEMX: .WORD 3
3196
3197                                     ;STORAGE FOR RPADR (THE FIRST ADDRESS (776700) OF THE RH11/RP04/5/6),
3198                                     ;RPVEC (THE VECTOR ADDRESS (254)), AND RPVEC+2 (THE BR LEVEL (5)).
3199
3200 012474      176700      RPADR: .WORD 176700
3201 012476      000254      RPVEC: .WORD 254,5*32.
3202
3203                                     ;MAXIMUM NUMBER OF LOOK AHEADS ALLOWED IS 4 (MXLACT=1 WORD)
3204
3205 012502      000004      MXLACT: .WORD 4
3206                                     ;MAXIMUM DELTA DELAY IS 8 SECTORS (MXDLTA=1 WORD)
3207
3208 012504      001000      MXDLTA: .WORD 8.*64.
3209                                     ;MINIMUM DELTA DELAY IS 2 SECTORS (MNDLTA=1 WORD)
3210
3211 012506      000200      MNDLTA: .WORD 2*64.
3212                                     ;MAXIMUM SEARCH FOR I/O WINDOW IS 5 SECTORS (MXWNDW=1 WORD)
3213
3214 012510      000005      MXWNDW: .WORD 5
3215
3216                                     ;DEFINITIONS OF THE RH11/RP04/5/6 ADDRESS INDEXES
3217
3218      000000      RPCS1=0      ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00)
3219      000002      RPWC=2      ;WORD COUNT REGISTER (NOT A DRIVE REG)
3220      000004      RPBA=4      ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
3221      000006      RPDA=6      ;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 05)
3222      000010      RPCS2=10     ;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
3223      000012      RPDS1=12     ;DRIVE STATUS REGISTER (DRIVE REG 01)
3224      000014      RPER1=14     ;ERROR REGISTER #1 (DRIVE REG. 02)
3225      000016      RPAS=16     ;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 04)
3226      000020      RPLA=20     ;LOOK AHEAD REGISTER (DRIVE REG. 07)
3227      000022      RPDB=22     ;DATA BUFFER REGISTER (NOT A DRIVE REG.)
3228      000024      RPMR=24     ;MAINTAINABILITY REGISTER (DRIVE REG. 03)
3229      000026      RPD1=26     ;DRIVE TYPE REGISTER (DRIVE REG. 06)
3230      000030      RPSN=30     ;SERIAL NUMBER REGISTER (DRIVE REG. 10)
3231      000032      RPOF=32     ;OFFSET REGISTER (DRIVE REG. 11)
3232      000034      RPCA=34     ;DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
3233      000036      RPCC=36     ;CURRENT CYLINDER ADDRESS REGISTER (DRIVE REG. 13)
3234      000040      RPER2=40     ;ERROR REGISTER #2 (DRIVE REG. 14)
3235      000042      RPER3=42     ;ERROR REGISTER #3 (DRIVE REG. 15)
3236      000044      RPEC1=44     ;ECC POSITION REGISTER (DRIVE REG. 16)

```



```

3237          000046          RPEC2=46          ;ECC PATTERN REGISTER (DRIVE REG. 17)
3238
3239          ;RH11/RP04/5/6 DRIVER INITIALIZATION CODE
3240          ;THIS ROUTINE WILL DETERMINE WHICH RP04/5/6 DRIVES ARE
3241          ;AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
3242          ;TO THE PROPER STATE FOR EACH DRIVE.
3243          ;NOTE: THIS ROUTINE CALLS DRVINT
3244
3245          :CALL
3246
3247          JSR      PC,RPINIT
3248          RETURN
3249
3250          ;NOTE: THE 'P' OR 'L' CLOCK MUST BE STARTED
3251
3252          RPINIT: SAVREG          ;SAVE R0 - R5
3253          MOV      @#PS,-(SP)      ;SAVE THE PRESENT PROCESSOR STATUS
3254          MOV      #<5*32.>,@#PS    ;CHANGE THE PRIORITY TO 5
3255          JSR      PC,CLRQUE        ;CLEAR ALL REQUEST QUEUES
3256          MOV      #RPERRS,R1      ;FIRST ADDRESS TO BE CLEARED
3257          MOV      #SEEKFG,R2      ;LAST ADDRESS TO BE CLEARED
3258          1$:      CLR      (R1)+    ;CLEAR
3259          CMP      R1,R2            ;ARE WE DONE?
3260          BLOS     1$              ;BRANCH IF NO
3261          MOV      #DTUW,R2        ;LAST ADDRESS
3262          2$:      MOV      #-1,(R1)+ ;INITIALIZE
3263          CMP      R1,R2            ;DONE?
3264          BLOS     2$              ;LOOP IF NO
3265          CLR      DRVSTA          ;SET ALL DRIVES TO OFFLINE
3266          CLR      DRVSTA+2
3267          CLR      DRVSTA+4
3268          CLR      DRVSTA+6
3269          MOV      RPVEC,R3        ;SETUP THE RH11/RP04/5/6 VECTOR
3270          MOV      #ISR,(R3)+
3271          MOV      RPVEC+2,(R3)
3272          MOV      RPADR,R4        ;FIRST ADDRESS OF RH11/RP04
3273          MOV      #BIT05,RPCS2(R4) ;MASSBUS INIT
3274          CLR      R1              ;START WITH DRIVE 0
3275          3$:      JSR      R0,DRVINT ;INIT THE DRIVE
3276          BR      4$              ;'DVA' NOT SET OR PARITY ERROR
3277          BR      5$              ;NORMAL RETURN
3278          4$:      CLRB     DRVSTA(R1) ;SET DRIVE STATUS TO OFFLINE
3279          5$:      INC      R1        ;GO TO NEXT DRIVE
3280          BIC      #^C7,R1        ;MASK OUT UNUSED BITS
3281          BNE      3$              ;BR IF MORE DRIVES TO GO
3282          MOV      #7,R1          ;START WITH DRIVE 7
3283          CLR      @#PS          ;CLEAR THE PROCESSOR STATUS
3284          6$:      TSTB     DPINT(R1) ;WAITING FOR DRIVE TO SWITCH PORTS ?
3285          BEQ      8$              ;BR NOT WAITING
3286          JSR      PC,SET.IE      ;SET INTERRUPT
3287          7$:      TSTB     DPINT(R1) ;DRIVE SWITCHED PORTS ?
3288          BNE      7$              ;BR IF NOT
3289          8$:      DEC      R1        ;GO TO THE NEXT DRIVE
3290          BPL      6$              ;CHECK NEXT DRIVE
3291          MOV      (SP)+,@#PS      ;RESTORE THE PROCESSOR STATUS
3292          RESREG          ;RESTORE R0 - R5
  
```



```

3293 012722 000207          RTS      PC          ;BYE-BYE
3294
3295          ;DRIVE INITIALIZATION ROUTINE
3296          ;THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
3297          ;AN RP04/5/6. IF IT IS, A 'READ-IN PRESET' IS ISSUED AND FMT22
3298          ;IS SET TO A '1'. THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
3299          ;INSURE THEY ARE ALL ON A '1'. AND DEPENDING ON THEIR STATE,
3300          ;DRVSTA IS SET TO THE PROPER CONDITION.
3301          ;CALL
3302          :
3303          :      MOV      #DRVNUM,R1          ;DRIVE NUMBER TO R1
3304          :      MOV      RPADR,R4          ;UNIBUS ADDRESS OF RH11/RP04/5/6 (RPCS1)
3305          :      JSR      R0,DRVINT          ;CALLED BY A JSR
3306          :      RETURN1          ;ERROR OCCURRED (PARITY)
3307          :      RETURN2          ;NORMAL RETURN
3308          :
3309 012724 010546          DRVINT: MOV      R5,-(SP)          ;SAVE R5
3310 012726 105061 012346          CLR      DRVSTA(R1)          ;START DRIVE STATUS AS OFFLINE
3311 012732 105061 012356          CLR      DRVSTYP(R1)          ;CLEAR THE DRIVE TYPE INDICATOR
3312 012736 105061 012414          CLR      ULDFLG(R1)          ;CLEAR THE UNLOAD FLAG
3313 012742 010164 000010          MOV      R1,RPCS2(R4)          ;SELECT A DRIVE
3314 012746 112764 000111 000000          MOV      #111,RPCS1(R4)          ;DO A DRIVE CLEAR COMMAND (& SEIZE DRIVE)
3315 012754 032764 010000 000010          BIT      #BIT12,RPCS2(R4)          ;NONEXISTENT DRIVE?
3316 012762 001403          BEQ      1$          ;NO---BRANCH
3317 012764 004737 020162          JSR      PC,SET.IE          ;GO SET 'IE' WITHOUT A 'TRE'
3318 012770 000533          BR      6$          ;LEAVE THIS ROUTINE
3319 012772 105061 012346          1$: CLR      DRVSTA(R1)          ;SET DRIVE STATUS TO OFFLINE
3320 012776 032764 004000 000000          BIT      #BIT11,RPCS1(R4)          ;SEE IF DRIVE AVAILABLE
3321 013004 001527          BEQ      7$          ;BR IF DRIVE NOT AVAILABLE
3322 013006 004037 017472          JSR      R0,RD.RP          ;READ THE DRIVE TYPE REG.
3323 013012 000026          RPD      8$          ;ERROR RETURN ADDRESS
3324 013014 013304          MOV      (SP)+,R5          ;PUT DRIVE TYPE IN R5
3325 013016 012605          MOV      #1,DRVSTYP(R1)          ;SET RP04 INDICATOR
3326 013020 112761 000001 012356          CMP      #20020,R5          ;IS IT A SINGLE PORT RP04?
3327 013026 022705 020020          BEQ      2$          ;BRANCH IF YES
3328 013032 001431          CMP      #24020,R5          ;IS IT A DUAL PORT RP04?
3329 013034 022705 024020          BEQ      2$          ;BR IF YES
3330 013040 001426          MOV      #2,DRVSTYP(R1)          ;SET RP05 INDICATOR
3331 013042 112761 000002 012356          CMP      #20021,R5          ;SINGLE PORT RP05 ?
3332 013050 022705 020021          BEQ      2$          ;BR IF YES
3333 013054 001420          CMP      #24021,R5          ;DUAL PORT RP05 ?
3334 013056 022705 024021          BEQ      2$          ;BR IF YES
3335 013062 001415          MOV      #4,DRVSTYP(R1)          ;SET RP06 INDICATOR
3336 013064 112761 000004 012356          CMP      #20022,R5          ;SINGLE PORT RP06 ?
3337 013072 022705 020022          BEQ      2$          ;BR IF YES
3338 013076 001407          CMP      #24022,R5          ;DUAL PORT RP06 ?
3339 013100 022705 024022          BEQ      2$          ;BR IF YES
3340 013104 001404          MOV      #-1,DRVSTYP(R1)          ;SET INDICATOR TO 'OTHER'
3341 013106 112761 177777 012356          BR      6$          ;EXIT
3342 013114 000461          TST      TSTPGM          ;INHIBIT PROGRAMMABLE DRIVE?
3343 013116 005737 013310          BNE      9$          ;BRANCH IF NO
3344 013122 001010          BIT      #BIT09,RPDS1(R4)          ;IS DRIVE PROGRAMMABLE?
3345 013124 032764 001000 000012          BEQ      9$          ;BRANCH IF NO
3346 013132 001404          BIS      #BIT03,DRVSTYP(R1)          ;SET INDICATOR
3347 013134 152761 000010 012356          BR      6$          ;EXIT
3348 013142 000446

```



```

3349 013144 012746 000121      9$:  MOV    #121,-(SP)      ;DO A 'READ-IN PRESET'
3350 013150 004037 017652      JSR    R0,WRT.RP
3351 013154 000000      RPCS1
3352 013156 013304      8$
3353 013160 012746 010000      MOV    #BIT12,-(SP)      ;SET FMT22=1
3354 013164 004037 017652      JSR    R0,WRT.RP
3355 013170 000032      RPOF
3356 013172 013304      8$
3357 013174 004037 017472      JSR    R0,RD.RP      ;READ RPDS1
3358 013200 000012      RPDS1
3359 013202 013304      8$
3360 013204 012605      MOV    (SP)+,R5      ;AND SAVE IT IN R5
3361 013206 100015      BPL    4$      ;BRANCH IF ATA=0
3362 013210 116164 012462 000016  MOVB   ATABIT(R1),RPAS(R4) ;CLEAR ATTENTION BIT
3363 013216 004037 017472      JSR    R0,RD.RP      ;FIND OUT WHY ATA=1
3364 013222 000014      RPER1
3365 013224 013304      8$
3366 013226 006126      ROL    (SP)+      ;IS IT UNSAFE?
3367 013230 100004      BPL    4$      ;BR IF NOT
3368 013232 112761 177777 012346  MOVB   #-1,DRVSTA(R1) ;SET UNSAFE INDICATOR
3369 013240 000407      BR     6$      ;EXIT
3370 013242 005105      4$:  COM    R5      ;CHECK MOL, DPR, DRY, AND VV
3371 013244 042705 167077      BIC    #^C<BIT12!BIT08!BIT07!BIT06>,R5
3372 013250 001003      BNE    6$      ;BRANCH IF MOL, DPR, DRY, OR VV IS CLEAR
3373 013252 112761 000001 012346  MOVB   #1,DRVSTA(R1) ;SET DRIVE STATUS TO ONLINE
3374 013260 005720      6$:  TST    (R0)+      ;STEP OVER THE ERROR RETURN
3375 013262 000410      BR     8$      ;EXIT
3376 013264 006301      7$:  ASL    R1      ;CHANGE INDEX TO ADDRESS WORDS
3377 013266 012761 003720 012440  MOV    #2000.,TIMER(R1) ;START 2 SEC TIMER
3378 013274 006201      ASR    R1      ;RESTORE R1
3379 013276 105161 012366      COMB   DPINT(R1)      ;SET PORT INITIALIZE INIDICATOR
3380 013302 005720      TST    (R0)+
3381 013304 012605      8$:  MOV    (SP)+,R5      ;RESTORE R5
3382 013306 000200      RTS    R0      ;EXIT
3383
3384      ;TEST PROGRAMMABLE DRIVE FLAG (TSTPGM=1 WORD)
3385      ;THE FLAG WILL BE SET BY THE PROGRAM UNDER
3386      ;MANUFACTURING CONDITIONS (ACT,APT) AND
3387      ;CLEARED UNDER FIELD CONDITIONS(XXDP CHAIN,
3388      ;STANDALONE) WITH STARTING ADDRESS 200.
3389      ;THE FLAG WILL BE SET UNDER ALL CONDITIONS
3390      ;WITH STARTING ADDRESS 220.
3391
3392 013310 000000      TSTPGM: .WORD 0      ;=1 TEST PROGRAMMABLE DRIVE
3393
3394      ;REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
3395      ;CALL
3396      ;
3397      ;
3398      JSR    R0,@#RP04      ;CALL THE RP04/5/6 DRIVER
3399      PNTADR      ;ADDRESS OF POINTER OF DRIVES PARAMETER BLOCK
3400      RETURN1      ;RETURN HERE IF QUEUE IS FULL
3401      RETURN2      ;RETURN HERE IF REQUEST IS IN QUEUE OR THERE
3402      ;IS AN ERROR CONDITION
3403
3404 013312 013746 177776      RP04:  MOV    @#PS,-(SP)      ;SAVE THE CALLING STATUS

```


Address	Hex	Hex	Hex	Hex	Assembly	Comment
3405	013316	013737	012500	177776	MOV	RPVEC+2,@#PS ;DON'T ALLOW ANY RP04/5/6 INTERRUPTS
3406	013324	112737	000001	012412	MOVB	#1,ACTDRV ;SET 'ACTIVE DRIVER' FLAG
3407	013332	104412			SAVREG	;SAVE R0 - R5
3408	013334	011002			MOV	(R0),R2 ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
3409	013336	005062	000016		CLR	16(R2) ;CLEAR THE STATUS/ERROR INDICATOR
3410	013342	111201			MOVB	(R2),R1 ;PICKUP THE DRIVE NUMBER
3411	013344	013704	012474		MOV	RPADR,R4 ;UNIBUS ADDRESS OF RPCS1
3412	013350	105761	012346		TSTB	DRVSTA(R1) ;CHECK DRIVES STATUS
3413	013354	003014			BGT	1\$;BRANCH IF ONLINE
3414	013356	105761	012414		TSTB	ULDFLG(R1) ;UNLOAD COMMAND IN QUEUE?
3415	013362	001036			BNE	3\$;BRANCH IF YES
3416	013364	105761	012366		TSTB	DPINT(R1) ;TRYING TO INIT THE DRIVE
3417	013370	001042			BNE	5\$;BR IF YES
3418	013372	004037	012724		JSR	R0,DRVINT ;GO INIT. THE DRIVE
3419	013376	000434			BR	4\$;ERROR RETURN
3420	013400	105761	012346		TSTB	DRVSTA(R1) ;IS DRIVE STATUS ONLINE?
3421	013404	003445			BLE	6\$;BR IF NOT
3422	013406	105761	012376	1\$:	TSTB	DPRQS(R1) ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
3423	013412	001031			BNE	5\$;BR IF YES
3424	013414	010164	000010		MOV	R1,RPCS2(R4) ;SELECT THE DRIVE
3425	013420	004037	020624		JSR	R0,DRVQUE ;PUT THIS REQUEST IN QUEUE
3426	013424	000460			BR	9\$;QUEUE IS FULL
3427	013426	122762	000103	000002	CMPB	#103,2(R2) ;IS THIS REQ. FOR AN UNLOAD?
3428	013434	001003			BNE	2\$;BR IF NO
3429	013436	112761	177777	012414	MOVB	#-1,ULDFLG(R1) ;SET THE 'UNLOAD IN QUEUE' FLAG
3430	013444	105761	012336	2\$:	TSTB	DRVACT(R1) ;IS THIS DRIVE ACTIVE?
3431	013450	001043			BNE	8\$;BR IF YES
3432	013452	004737	013604		JSR	PC,OPT ;CALL THE OPTIMIZER
3433	013456	000440			BR	8\$
3434	013460	012762	120000	000016	3\$:	MOV #BIT15!BIT13,16(R2) ;SET THE 'UNLOAD IN QUEUE' ERROR FLAG
3435	013466	000434			BR	8\$;EXIT
3436	013470	004737	014714	4\$:	JSR	PC,C17 ;GO HANDLE THE PARITY ERROR
3437	013474	000431			BR	8\$
3438	013476	004037	020624	5\$:	JSR	R0,DRVQUE ;PUT REQUEST IN QUEUE
3439	013502	000431			BR	9\$;QUEUE IS FULL
3440	013504	032714	000100		BIT	#BIT06,(R4) ;IS 'IE' SET ALREADY ?
3441	013510	001023			BNE	8\$;BR IF IT IS
3442	013512	004737	020162		JSR	PC,SET.IE ;SET INTERRUPT
3443	013516	000420			BR	8\$;RETURN, REQUEST IN QUEUE
3444	013520	105761	012346	6\$:	TSTB	DRVSTA(R1) ;SEE IF DRIVE OFFLINE OR UNSAFE
3445	013524	002412			BLT	7\$;BR IF UNSAFE
3446	013526	012762	140000	000016	MOV	#BIT15!BIT14,16(R2) ;SET OFFLINE ERROR INDICATOR
3447	013534	105761	012356		TSTB	DRVTYP(R1) ;SEE IF OFFLINE OR NONEXISTENT
3448	013540	001007			BNE	8\$;BR IF OFFLINE
3449	013542	012762	100002	000016	MOV	#BIT15!BIT01,16(R2) ;REPORT DRIVE NONEXISTENT
3450	013550	000403			BR	8\$;GO TO EXIT
3451	013552	012762	110000	000016	7\$:	MOV #BIT15!BIT12,16(R2) ;DRIVE IS UNSAFE
3452	013560	104413		8\$:	RESREG	;RESTORE R0 - R5
3453	013562	005720			TST	(R0)+ ;SETUP FOR NORMAL RETURN
3454	013564	000401			BR	10\$;FINISH UP, THEN EXIT
3455	0					


```

3461 ;OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
3462
3463 :CALL
3464 :
3465 :   MOV    #DRVNUM,R1    ;DRIVE NUMBER TO R1
3466 :   JSR    PC,OPT        ;SETUP A COMMAND
3467
3467 013604 104412 OPT: SAVREG ;SAVE R0 - R5
3468 013606 013746 MOV @#PS,-(SP) ;SAVE PROC. STATUS
3469 013612 146137 012462 012410 BICB ATABIT(R1),SRCHWT ;CLEAR 'SEARCH WAIT' KEY
3470 013620 004737 020700 JSR PC,GETREQ ;GET 'DPB' POINTER OF REQUEST
3471 013624 005702 TST R2 ;IS THERE A REQUEST IN QUEUE?
3472 013626 001505 BEQ 7$ ;NO--BRANCH TO EXIT
3473 013630 032764 004000 000000 BIT #BIT11,RPCS1(R4) ;IS DVA SET?
3474 013636 001407 BEQ 10$ ;BRANCH IF NOT
3475 013640 032764 000100 000012 BIT #BIT6,RPDS1(R4) ;IS VV SET?
3476 013646 001003 BNE 10$ ;BR IF IT IS
3477 013650 004037 012724 9$: JSR R0,DRVINT ;SEE IF DRIVE STILL ONLINE?
3478 013654 000470 BR 6$ ;PARITY OR 'DVA' NOT SET
3479 013656 105761 012346 10$: TSTB DRVSTA(R1) ;IS DRIVE ONLINE?
3480 013662 003014 BGT 1$ ;YES--BRANCH
3481 013664 004737 020722 JSR PC,POPQUE ;NO--REMOVE REQUEST FROM QUEUE
3482 013670 012762 140000 000016 MOV #BIT15!BIT14,16(R2) ;SET OFFLINE STATUS/ERROR INDICATOR
3483 013676 105761 012346 TSTB DRVSTA(R1) ;IS DRIVE UNSAFE?
3484 013702 100064 BPL 8$ ;BR TO EXIT IF NOT
3485 013704 012762 110000 000016 MOV #BIT15!BIT12,16(R2) ;SET UNSAFE STATUS/ERROR INDICATOR
3486 013712 000460 BR 8$ ;BRANCH TO EXIT
3487 013714 012746 000111 1$: MOV #111,-(SP) ;LOAD COMMAND ONTO THE STACK
3488 013720 004037 017652 JSR R0,WRT.RP ;LOAD THE REGISTER
3489 013724 000000 RPCS1 ;REGISTER INCREMENT
3490 013726 014036 6$ ;ERROR RETURN ADDRESS
3491 013730 032714 004000 BIT #BIT11,(R4) ;DRIVE AVAILABLE?
3492 013734 001427 BEQ 5$ ;BR IF NOT
3493 013736 122762 000150 000002 CMPB #150,2(R2) ;IS THE REQUEST FOR I/O?
3494 013744 002403 BLT 2$ ;YES--BRANCH
3495 013746 004737 014300 JSR PC,C14 ;CALL THE COMMAND INITIATOR
3496 013752 000440 BR 8$ ;BRANCH TO EXIT
3497 013754 005737 012460 2$: TST DTUW ;DATA TRANSFER UNDERWAY?
3498 013760 002012 BGE 4$ ;YES--GO START A SEARCH
3499 013762 005737 012436 TST SEEKFG ;DO IMPLIED SEEKS?
3500 013766 100404 BMI 3$ ;YES---BRANCH
3501 013770 004037 015250 JSR R0,LA ;NO--DO LOOK AHEAD
3502 013774 000427 BR 8$ ;RETURN HERE ON A PARITY ERROR
3503 013776 000403 BR 4$ ;GO START A SEARCH
3504 014000 004737 014064 3$: JSR PC,C11 ;START A DATA TRANSFER
3505 014004 000423 BR 8$
3506 014006 004737 014172 4$: JSR PC,C13 ;START A SEARCH
3507 014012 000420 BR 8$ ;GO TO THE EXIT
3508 014014 112761 177777 012376 5$: MOVB #-1,DPRQS(R1) ;SET PORT REQUEST INDICATOR
3509 014022 010103 MOV R1,R3 ;SET UP TO ADDRESS WORDS
3510 014024 006303 ASL R3 ;CONVERT TO WORD INDEX
3511 014026 012763 023420 012440 MOV #10000.,TIMER(R3) ;START 10 SEC TIMER
3512 014034 000402 BR 7$ ;EXIT
3513 014036 004737 014714 6$: JSR PC,C17 ;PROCESS THE PARITY ERROR
3514 014042 032714 000100 7$: BIT #BIT06,(R4) ;SEE IF 'IE' ALREADY SET
3515 014046 001002 BNE 8$ ;BR IF SET
3516 014050 004737 020162 JSR PC,SET.IE ;SET 'IE' WITHOUT A 'TRE'

```



```

3517 014054 012637 177776      8$:  MOV      (SP)+, @#PS      ;RESTORE PROC. STATUS
3518 014060 104413              RESREG      ;RESTORE R0 - R5
3519 014062 000207              RTS        PC
3520
3521      ;COMMAND INITIATOR
3522
3523      ;CALL
3524      MOV      #DRVNUM, R1      ;DRIVE NUMBER
3525      MOV      #DPB, R2        ;ADDRESS OF DPB
3526      JSR      PC, CI?         ;CI? = CI1, CI3, OR CI4
3527      ;WHERE:
3528      ;CI1=DATA TRANSFER
3529      ;CI2=SEARCH REQUESTED BY DATA XFER
3530      ;CI4=NOT DATA TRANSFER
3531
3532 014064 004737 020722      CI1:  JSR      PC, POPQUE      ;REMOVE REQUEST FROM 'DRIVES WAIT' QUEUE
3533 014070 010237 012406      MOV      R2, TRNSWT      ;PUT REQ. IN TRANSFER WAIT QUEUE
3534 014074 010203              MOV      R2, R3        ;DPB ADDRESS TO R3
3535 014076 013704 012474      MOV      RPADR, R4        ;RPCS1 ADDRESS
3536 014102 010164 000010      MOV      R1, RPCS2(R4)    ;SELECT DRIVE
3537 014106 062703 000004      ADD      #4, R3          ;DESIRED WORD COUNT
3538 014112 062704 000002      ADD      #2, R4          ;RPWC ADDRESS
3539 014116 012324              MOV      (R3)+, (R4)+      ;LOAD WORD COUNT
3540 014120 012324              MOV      (R3)+, (R4)+      ;LOAD BUFFER ADDRESS
3541 014122 012346              MOV      (R3)+, -(SP)      ;LOAD SECTOR AND TRACK
3542 014124 004037 017652      JSR      R0, WRT.RP        ;CALL THE LOAD(WRITE) ROUTINE
3543 014130 000006              RPDA
3544 014132 014714              CI7
3545 014134 012346              MOV      (R3)+, -(SP)      ;LOAD CYLINDER ADDRESS
3546 014136 004037 017652      JSR      R0, WRT.RP
3547 014142 000034              RPCA
3548 014144 014714              CI7
3549 014146 016246 000002      MOV      2(R2), -(SP)      ;LOAD 'COMMAND+GO', 'A17&A16', AND 'PSEL'
3550 014152 004037 017652      JSR      R0, WRT.RP
3551 014156 000000              RPCS1
3552 014160 014714              CI7
3553 014162 010137 012460      MOV      R1, DTUW          ;SET 'DATA TRANSFER UNDERWAY'
3554 014166 000137 014656      JMP      CI5
3555 014172 013704 012474      CI3:  MOV      RPADR, R4        ;RPCS1 ADDRESS
3556 014176 010164 000010      MOV      R1, RPCS2(R4)    ;SELECT DRIVE
3557 014202 016246 000012      MOV      12(R2), -(SP)     ;DESIRED CYLINDER ADDRESS
3558 014206 004037 017652      JSR      R0, WRT.RP
3559 014212 000034              RPCA
3560 014214 014714              CI7
3561 014216 116203 000010      MOV      10(R2), R3        ;PICKUP SECTOR ADDRESS
3562 014222 163703 012510      SUB      MXWINDW, R3      ;BACKUP BY MAX. SEARCH FOR I/O WINDOW
3563 014226 002002              BGE      1$
3564 014230 062703 000026      ADD      #22, R3
3565 014234 010346              MOV      R3, -(SP)
3566 014236 116266 000011      MOV      11(R2), 1(SP)     ;COMBINE THE ADJUSTED SECTOR WITH
3567 014244 004037 017652      MOV      R0, WRT.RP        ;THE DESIRED TRACK
3568 014250 000006              JSR      R0, WRT.RP        ;LOAD DESIRED TRACK & SECTOR
3569 014252 014714              RPDA
3570 014254 012746 000131      CI7
3571 014260 004037 017652      MOV      #131, -(SP)      ;START A SEARCH
3572 014264 000000              JSR      R0, WRT.RP
3573                                RPCS1
  
```


3573	014266	014714			CI7		
3574	014270	156137	012462	012410	BISB	ATABIT(R1),SRCHWT	;SET 'SEARCH WAIT' KEY
3575	014276	000567			BR	CI5	
3576	014300	013704	012474		MOV	RPADR,R4	;RPCS1 ADDRESS
3577	014304	010164	000010		MOV	R1,RPCS2(R4)	;SELECT DRIVE
3578	014310	116203	000002		MOVB	2(R2),R3	;PICKUP THE REQUESTED COMMAND
3579	014314	122703	000131		CMPB	#131,R3	;IS IT A SEARCH COMMAND?
3580	014320	001007			BNE	1\$	;BRANCH IF NO
3581	014322	016246	000010		MOV	10(R2),-(SP)	;LOAD DESIRED TRACK & SECTOR
3582	014326	004037	017652		JSR	R0,WRT.RP	
3583	014332	000006			RPDA		
3584	014334	014714			CI7		
3585	014336	000403			BR	2\$	;GO LOAD CYLINDER
3586	014340	122703	000105		CMPB	#105,R3	;IS IT A SEEK COMMAND
3587	014344	001007			BNE	3\$	;BRANCH IF NO
3588	014346	016246	000012		MOV	12(R2),-(SP)	;LOAD DESIRED CYLINDER
3589	014352	004037	017652		JSR	R0,WRT.RP	
3590	014356	000034			RPCA		
3591	014360	014714			CI7		
3592	014362	000546			BR	CI6	
3593	014364	122703	000115		CMPB	#115,R3	;IS IT AN 'OFFSET' COMMAND?
3594	014370	001013			BNE	4\$	;BR IF NO
3595	014372	004037	017472		JSR	R0,RD.RP	;MERGE THE OFFSET VALUE INTO RPOF
3596	014376	000032			RPOF		;BUT DON'T CHANGE THE UPPER
3597	014400	014714			CI7		
3598	014402	116216	000001		MOVB	1(R2),-(SP)	;BYTE WHEN LOADING THE
3599	014406	004037	017652		JSR	R0,WRT.RP	;REGISTER (RPOF)
3600	014412	000032			RPOF		
3601	014414	014714			CI7		
3602	014416	000530			BR	CI6	;GO START THE COMMAND
3603	014420	122703	000107		CMPB	#107,R3	;IS IT A 'RECALIBRATE' COMMAND?
3604	014424	001525			BEQ	CI6	;BRANCH IF YES
3605	014426	122703	000117		CMPB	#117,R3	;IS IT A RETURN TO CENTER?
3606	014432	001522			BEQ	CI6	;BRANCH IF YES
3607	014434	122703	000103		CMPB	#103,R3	;IS IT AN 'UNLOAD' COMMAND?
3608	014440	001016			BNE	5\$	;BRANCH IF NO
3609	014442	112761	000001	012336	MOVB	#1,DRVACT(R1)	;SET THE DRIVE ACTIVE INDICATOR
3610	014450	105061	012346		CLRB	DRVSTA(R1)	;PUT DRIVE STATUS TO OFFLINE
3611	014454	112761	000001	012414	MOVB	#1,ULDFLG(R1)	;SET 'UNLOAD IN PROGRESS' FLAG
3612	014462	010346			MOV	R3,-(SP)	;START THE 'UNLOAD' COMMAND
3613	014464	004037	017652		JSR	R0,WRT.RP	
3614	014470	000000			RPCS1		
3615	014472	014714			CI7		
3616	014474	000207			RTS	PC	;RETURN TO USER
3617	014476	122703	000143		CMPB	#143,R3	;IS IT A 'SET FORMAT' COMMAND?
3618	014502	001014			BNE	6\$	;BRANCH IF NO
3619	014504	004037	017472		JSR	R0,RD.RP	;READ THE OFFSET REGISTER
3620	014510	000032			RPOF		
3621	014512	014714			CI7		
3622	014514	116266	000001	000001	MOVB	1(R2),1(SP)	;COMBINE 'FMT22','ECI', AND 'HCI'
3623	014522	004037	017652		JSR	R0,WRT.RP	;LOAD 'FMT22','ECI', AND/OR 'HCI'.
3624	014526	000032			RPOF		
3625	014530	014714			CI7		
3626	014532	000436			BR	12\$	
3627	014534	122703	000141		CMPB	#141,R3	;IS IT A 'GET REGISTER' COMMAND?
3628	014540	001023			BNE	10\$	;BRANCH IF NO

3629	014542	016203	000006		7\$:	MOV	6(R2),R3	:POINTS TO 1ST ADDRESS OF WHERE
3630								:TO PUT THE REGISTER(S)
3631	014546	116237	000010	014564		MOVB	10(R2),9\$	:INIT. THE INDEX FOR THE FIRST REG.
3632	014554	116205	000011			MOVB	11(R2),R5	:INDEX OF LAST REG. TO MOVE
3633	014560	004037	017472		8\$:	JSR	R0,RD.RP	:READ RP04/5/6 REGISTER
3634	014564	000000			9\$:	RPCS1		:INDEX OF REG. TO READ
3635	014566	014714				CI7		
3636	014570	012623				MOV	(SP)+,(R3)+	:GET THE CONTENTS OF RH11/RP04/5/6 REG.
3637	014572	023705	014564			CMP	9\$,R5	:LAST REG. BEEN READ?
3638	014576	001414				BEQ	12\$	:GET OUT IF YES
3639	014600	062737	000002	014564		ADD	#2,9\$	:INCREASE THE INDEX BY 2
3640	014606	000764				BR	8\$	:LOOP--MORE TO READ
3641	014610	122703	000145		10\$:	CMPB	#145,R3	:IS IT A 'SELECT DRIVE' COMMAND?
3642	014614	001405				BEQ	12\$	:BRANCH IF YES
3643	014616	010346			11\$:	MOV	R3,-(SP)	:LOAD THE COMMAND
3644	014620	004037	017652			JSR	R0,WRT.RP	
3645	014624	000000				RPCS1		
3646	014626	014714				CI7		
3647	014630	004737	020722		12\$:	JSR	PC,POPQUE	:REMOVE REQ. FROM QUEUE
3648	014634	052762	000200	000016		BIS	#BIT07,16(R2)	:SET THE 'DONE' BIT
3649	014642	005737	012434			TST	SAVEFG	:SAVE THE RH11/RP04/5/6 REGISTERS?
3650	014646	100002				BPL	13\$	:BRANCH IF NO
3651	014650	004737	020044			JSR	PC,SVRH11	:YES--GO SAVE THE REGISTERS
3652	014654	000207			13\$:	RTS	PC	:RETURN TO USER
3653	014656	006301			CI5:	ASL	R1	
3654	014660	012761	001750	012440		MOV	#1000.,TIMER(R1)	:SET A ONE SECOND TIMER
3655	014666	006201				ASR	R1	
3656	014670	112761	000001	012336		MOVB	#1,DRVACT(R1)	:SET THE DRIVE ACTIVE
3657	014676	000207				RTS	PC	:RETURN TO THE USER
3658	014700	010346			CI6:	MOV	R3,-(SP)	:LOAD THE COMMAND
3659	014702	004037	017652			JSR	R0,WRT.RP	
3660	014706	000000				RPCS1		
3661	014710	014714				CI7		
3662	014712	000761				BR	CI5	
3663	014714	032764	010000	000010	CI7:	BIT	#BIT12,RPCS2(R4)	:DRIVE NON-EXISTENT ?
3664	014722	001034				BNE	CI8	:BR IF YES
3665	014724	005702			1\$:	TST	R2	:ANYTHING IN QUEUE ?
3666	014726	001405				BEQ	CI7B	:BR IF NOT
3667	014730	012762	104000	000016		MOV	#BIT15!BIT11,16(R2)	:SET 'PARITY' ERROR INDICATOR
3668	014736	004737	020044			JSR	PC,SVRH11	:GO SAVE THE RH11/RP04/5/6 REGISTERS
3669	014742	012746	000111		CI7B:	MOV	#111,-(SP)	:DO A 'DRIVE CLEAR'
3670	014746	004037	017652			JSR	R0,WRT.RP	
3671	014752	000000				RPCS1		
3672	014754	015014				CI8		
3673	014756	004737	020604			JSR	PC,EMPTYQ	:EMPTY THE QUEUE
3674	014762	105061	012414			CLRB	ULDFLG(R1)	:CLEAR THE UNLOAD IN QUEUE FLAG
3675	014766	105061	012336			CLRB	DRVACT(R1)	:DRIVE IS IDLE
3676	014772	020137	012460			CMP	R1,DTUW	:IF THIS DRIVE HAD AN I/O REQUEST
3677	014776	001005				BNE	1\$	:IN PROGRESS CLEAR ALL OF THE FLAGS
3678	015000	005037	012406			CLR	TRNSWT	
3679	015004	012737	177777	012460		MOV	#-1,DTUW	
3680	015012	000207			1\$:	RTS	PC	
3681	015014	104412			CI8:	SAVREG		:SAVE R0 - R5
3682	015016	032764	010000	000010		BIT	#BIT12,RPCS2(R4)	:IS 'NED' SET ?
3683	015024	001002				BNE	1\$	:BR IF YES
3684	015026	005001				CLR	R1	

Address	Offset	Hex	Label	Instruction	Comment
3685	015030	005003		CLR R3	
3686	015032	105761	012336	1\$: TSTB DRVACT(R1)	:DRIVE ACTIVE?
3687	015036	001443		BEQ 5\$	:BRANCH IF NO
3688	015040	013702	012406	MOV TRNSWT,R2	:GET THE 'TRANSFER WAIT' QUEUE
3689	015044	020137	012460	CMP R1,DTUW	:DID THIS DRIVE HAVE AN I/O IN PROGRESS?
3690	015050	001402		BEQ 2\$	:BRANCH IF YES
3691	015052	004737	020700	JSR PC,GETREQ	:GET THE DPB POINTER
3692	015056	005702		2\$: TST R2	:QUEUE ENTRY FOR DRIVE ?
3693	015060	001415		BEQ 4\$	:BR IF NOT
3694	015062	032764	010000 000010	BIT #BIT12,RPCS2(R4)	: 'NED' SET ?
3695	015070	001404		BEQ 3\$	:BR IF NOT
3696	015072	012762	100002 000016	MOV #BIT15!BIT01,16(R2)	:SET 'DRIVE NON-EXISTENT' INDICATOR
3697	015100	000405		BR 4\$	:CONTINUE
3698	015102	012762	102000 000016	3\$: MOV #BIT15!BIT10,16(R2)	:SET 'NON-CLEARABLE PARITY' ERROR INDICATOR
3699	015110	004737	020044	JSR PC,SVRH11	:SAVE RH11/RP04/5/6 REGISTERS
3700	015114	012763	177777 012440	4\$: MOV #-1,TIMER(R3)	:STOP THE TIMER
3701	015122	105061	012336	CLRB DRVACT(R1)	:SET 'DRIVE ACTIVE' TO IDLE
3702	015126	020137	012460	CMP R1,DTUW	:IS THIS DRIVE SETUP FOR A TRANSFER
3703	015132	001005		BNE 5\$	:BR IF NOT
3704	015134	012737	177777 012460	MOV #-1,DTUW	:RESET THE INDICATOR
3705	015142	005037	012406	CLR TRNSWT	:CLEAR THE TRANSFER QUEUE
3706	015146	105061	012414	5\$: CLRB ULDFLG(R1)	:CLEAR UNLOAD FLAG
3707	015152	032764	010000 000010	BIT #BIT12,RPCS2(R4)	: 'NED' SET ?
3708	015160	001021		BNE 6\$	:BR IF YES
3709	015162	005201		INC R1	:MOVE TO THE NEXT DRIVE
3710	015164	062703	000002	ADD #2,R3	
3711	015170	042701	177770	BIC #^C7,R1	
3712	015174	001316		BNE 1\$	:BRANCH IF MORE DRIVES
3713	015176	012737	177777 012460	MOV #-1,DTUW	:NO DATA TRANSFERS UNDERWAY
3714	015204	005037	012406	CLR TRNSWT	:CLEAR THE 'TRANSFER WAIT' QUEUE
3715	015210	004737	020526	JSR PC,CLRQUE	:CLEAR ALL OF THE REQUEST QUEUES
3716	015214	012764	000040 000010	MOV #BIT05,RPCS2(R4)	:DO A MASSBUS INIT.
3717	015222	000406		BR 7\$	:CONTINUE
3718	015224	004737	020604	6\$: JSR PC,EMPTYQ	:CLEAR THE DRIVE'S QUEUE
3719	015230	105061	012346	CLRB DRVSTA(R1)	:SET DRIVE TO OFFLINE
3720	015234	105061	012356	CLRB DRVTP(R1)	:CLEAR THE DRIVE TYPE INDICATOR
3721	015240	004737	020162	7\$: JSR PC,SET.IE	:SET 'IE' WITHOUT 'TRE'
3722	015244	104413		RESREG	:RESTORE R0 - R5
3723	015246	000207		RTS PC	:RETURN
3724					
3725				:LOOK AHEAD ROUTINE	
3726				:CALL	
3727					
3728				MOV #DRVNUM,R1	:DRIVE NUMBER
3729				MOV #DPB,R2	:POINT TO DPB
3730				JSR R0,LA	:GO CHECK THE WINDOW
3731				RETURN1	:ERROR RETURN
3732				RETURN2	:START A SEARCH
3733				RETURN3	:START A DATA TRANSFER
3734					
3735	015250	013704	012474	LA: MOV RPADR,R4	:GET RPCS1'S ADDRESS
3736	015254	010164	000010	MOV R1,RPCS2(R4)	:SELECT DRIVE
3737	015260	004037	017472	JSR R0,RD.RP	:READ CURRENT CYLINDER
3738	015264	000036		RPCC	
3739	015266	015400		4\$	:ERROR RETURN ADDRESS
3740	015270	022662	000012	CMP (SP)+,12(R2)	:IS CURRENT CYLINDER=DESIRED


```

3741
3742 015274 001037          BNE      3$          ;CYLINDER?
3743 015276 105261 012424  INCB     LACNT(R1)      ;EXIT IF NO
3744 015302 126137 012424 012502 CMPB     LACNT(R1),MXLACT ;INCREMENT THE LOOK AHEAD COUNT
3745 015310 003026          BGT      2$          ;EXCEED MAX?
3746 015312 116203 000010  MOV     10(R2),R3      ;BRANCH IF YES
3747 015316 000303          SWAB     R3            ;GET DESIRED SECTOR ADDRESS AND
3748 015320 006203          ASR      R3            ;MULT. BY 64--ALIGN WITH
3749 015322 006203          ASR      R3            ;LOOK AHEAD REGISTER
3750 015324 012737 000340 177776 MOV     #340,@#PS      ;PRIORITY LEVEL '7'
3751 015332 004037 017472  JSR      R0,RD.RP      ;READ LOOK AHEAD REGISTER
3752 015336 000020          RPLA     4$
3753 015340 015400          4$
3754 015342 162603          SUB      (SP)+,R3      ;CALCULATE THE DELTA
3755 015344 002002          BGE      1$
3756 015346 062703 002600  ADD      #<22.*64.>,R3 ;MAKE THE DELTA POSITIVE
3757 015352 023703 012504 1$:    CMP     MXDLTA,R3 ;CHECK THE DELTA TO SEE
3758 015356 002406          BLT      3$          ;IF IT IS WITHIN THE
3759 015360 023703 012506  CMP     MNDLTA,R3 ;WINDOW---IF YES, ZERO
3760 015364 002003          BGE      3$          ;THE LOOK AHEAD COUNT
3761 015366 105061 012424 2$:    CLRB     LACNT(R1) ;AND TAKE THE I/O EXIT
3762 015372 005720          TST      (R0)+
3763 015374 005720          3$:    TST      (R0)+ ;ADJUST THE RETURN ADDRESS
3764 015376 000402          BR       5$          ;EXIT
3765 015400 004737 014714 4$:    JSR      PC,C17 ;PROCESS THE ERROR
3766 015404 000200          5$:    RTS      R0      ;RETURN
3767
3768          ;INTERRUPT SERVICE ROUTINE
3769
3770 015406 112737 000001 012412 ISR:  MOV     #1,ACTDRV ;SET 'ACTIVE DRIVER' FLAG
3771 015414 104412          SAVREG ;SAVE R0 - R5
3772 015416 013704 012474          MOV     RPADR,R4 ;ADDRESS OF RHSCS1
3773 015422 013701 012460          MOV     DTUW,R1 ;GET 'DATA TRANSFER UNDERWAY' INDICATOR
3774 015426 002403          BLT      1$          ;BRANCH IF NO DATA TRANSFER UNDERWAY
3775 015430 004737 015452          JSR      PC,TD ;CALL TRANSFER DONE
3776 015434 000402          BR       2$          ;EXIT
3777 015436 004737 015612 1$:    JSR      PC,SC ;CALL SPECIAL CONDITIONS
3778 015442 104413          2$:    RESREG ;RESTORE R0 - R5
3779 015444 105037 012412          CLRB     ACTDRV ;CLEAR 'ACTIVE DRIVER' FLAG
3780 015450 000002          RTI      ;RETURN
3781
3782          ;TRANSFER DONE ROUTINE
3783
3784 015452 105061 012336  TD:    CLRB     DRVACT(R1) ;SET DRIVE ACTIVE INDICATOR TO IDLE
3785 015456 012737 177777 012460  MOV     #-1,DTUW ;NO DATA TRANSFERS UNDERWAY
3786 015464 006301          ASL      R1
3787 015466 012761 177777 012440  MOV     #-1,TIMER(R1) ;CANCEL TIMEOUT
3788 015474 006201          ASR      R1
3789 015476 03702 012406          MOV     TRNSWT,R2 ;GET 'DPB' ADDRESS FROM THE
3790 015502 005037 012406          CLR     TRNSWT ;TRANSFER WAIT QUEUE--CLEAR QUEUE
3791 015506 052762 000200 000016  BIS     #BIT07,16(R2) ;SET DONE
3792 015514 010164 000010          MOV     R1,RPCS2(R4) ;SELECT THE DRIVE
3793 015520 004037 017472          JSR      R0,RD.RP ;TRANSFER ERROR(TRE=1)?
3794 015524 000000          RPCS1
3795 015526 014714          C17
3796 015530 006126          ROL      (SP)+

```



```

3797 015532 100413 BMI 3$ ;BR IF YES
3798 015534 005737 012434 TST SAVEFG ;SAVE THE RH11/RP04/5/6 REGISTERS?
3799 015540 100002 BPL 1$ ;BRANCH IF NO
3800 015542 004737 020044 JSR PC,SVRH11 ;YES--SAVE THE REGISTERS
3801 015546 004737 013604 1$: JSR PC,OPT ;CALL OPTIMIZER
3802 015552 000417 BR SC ;CHECK OTHER DRIVES
3803 015554 012714 000113 2$: MOV #113,(R4) ;RELEASE THE DRIVE
3804 015560 000414 BR SC ;CHECK FOR OTHER DRIVES
3805 015562 052762 100100 000016 3$: BIS #BIT15:BIT06,16(R2) ;SET DATA ERROR FLAG
3806 015570 004737 020604 JSR PC,EMPTYQ ;EMPTY THE 'DRIVE'S WAIT' QUEUE
3807 015574 004737 020044 JSR PC,SVRH11 ;SAVE THE RH11/RP04/5/6 REGISTERS
3808 015600 012714 040111 MOV #40111,(R4) ;ISSUE A 'DRIVE CLEAR'
3809 015604 012714 000113 MOV #113,(R4) ;ISSUE A RELEASE TO THE DRIVE
3810 015610 000400 BR SC ;CHECK FOR OTHER DRIVES
3811
3812 ;SPECIAL CONDITION ROUTINE
3813
3814 015612 116403 000016 SC: MOVB RPAS(R4),R3 ;READ 'RPAS'
3815 015616 001014 BNE 2$ ;BRANCH IF ANY 'ATA' BITS SET
3816 015620 004037 017472 JSR R0,RD.RP ;READ CONTROL AND STATUS REGISTER
3817 015624 000000 RPCS1
3818 015626 015014 C18
3819 015630 106126 ROLB (SP)+ ;IS 'IE'=1?
3820 015632 100405 BMI 1$ ;YES, NO DRIVES TO CHECK
3821 015634 004037 020764 JSR R0,ES.SAV ;SAVE THE ADDRESS IN '$ESCAPE'
3822 015640 104001 ERROR 1 ;REPORT AN ILLEGAL INTERRUPT
3823 015642 004737 020162 JSR PC,SET.IE ;SET INTERRUPT ENABLE
3824 015646 000207 1$: RTS PC ;RETURN
3825 015650 005046 2$: CLR -(SP) ;PROCESS ALL DRIVES THAT HAVE
3826 015652 110316 MOVB R3,(SP) ;AN 'ATA'=1
3827 015654 012703 000001 MOV #1,R3
3828 015660 005001 CLR R1
3829 015662 030316 SC3: BIT R3,(SP) ;ATA=1?
3830 015664 001005 BNE SC5 ;YES--BRANCH
3831 015666 005201 SC4: INC R1 ;MOVE TO THE NEXT DRIVE
3832 015670 106303 ASLB R3
3833 015672 001373 BNE SC3 ;BRANCH IF MORE TO CHECK?
3834 015674 005726 TST (SP)+ ;CLEAN OFF THE STACK
3835 015676 000207 RTS PC ;RETURN TO USER
3836 015700 105761 012366 SC5: TSTB DPINT(R1) ;INITIALIZING THE DRIVE ?
3837 015704 001402 BEQ 1$ ;BR IF NOT
3838 015706 000137 016604 JMP SC13 ;PROCESS THE DRIVE
3839 015712 105761 012376 1$: TSTB DPRQS(R1) ;PORT REQUEST OUTSTANDING ?
3840 015716 001402 BEQ 2$ ;BR IF NOT
3841 015720 000137 016604 JMP SC13 ;START THE OUTSTANDING COMMAND
3842 015724 105761 012346 2$: TSTB DRVSTA(R1) ;CHECK THE DRIVE STATUS
3843 015730 003025 BGT 5$ ;BRANCH IF ONLINE
3844 015732 105761 012414 TSTB ULDFLG(R1) ;UNLOAD IN PROGRESS?
3845 015736 003422 BLE 5$ ;BRANCH IF NOT
3846 015740 004737 020700 JSR PC,GETREQ ;GET DPB POINTER
3847 015744 004737 020044 JSR PC,SVRH11 ;SAVE THE RH11/RP04/5/6 REGISTERS
3848 015750 004737 016534 JSR PC,SC12 ;SAVE RPDS1, RPER1, RPER2, AND RPER3
3849 ;ALSO DO A DRIVE INIT (DRVINT)
3850 015754 105761 012346 TSTB DRVSTA(R1) ;DID DRIVE COME ONLINE?
3851 015760 003416 BLE 6$ ;NO---BRANCH
3852 015762 032737 040000 012326 BIT #BIT14,RPERRS ;WAS THERE AN ERROR?

```



```

3853 015770 001002      BNE      3$      ;BR IF ERROR
3854 015772 000137 016444 JMP      SC11      ;NO ERROR
3855 015776 013705 012330 3$:      MOV      RPERRS+2,R5 ;YES -- PICKUP RPER1 AND
3856 016002 000502      BR       SC6A      ;GO PROCESS THE ERROR
3857 016004 105761 012336 5$:      TSTB     DRVACT(R1) ;DRIVE ACTIVE WITH COMMAND OR ERROR RECOVERY ?
3858 016010 001033      BNE      SC6      ;BR IF EITHER
3859 016012 004737 016534 JSR      PC,SC12    ;SAVE RPDS1, RPER1, RPER2, AND RPER3
3860                                ;ALSO DO A DRVINT
3861 016016 105761 012366 6$:      TSTB     DPINT(R1) ;TRYING TO INIT THE DRIVE ?
3862 016022 001321      BNE      SC4      ;BR IF YES, CHECK ON MORE DRIVES
3863 016024 105761 012346      TSTB     DRVSTA(R1) ;CHECK ON DRIVE'S STATUS
3864 016030 100412      BMI      7$      ;BR IF UNSAFE
3865 016032 032737 020000 012334 BIT      #BIT13,RPERRS+6 ;ADDRESS PLUG CHANGED ?
3866 016040 001013      BNE      8$      ;BR IF YES
3867 016042 012746 000113 MOV      #113,-(SP) ;RELEASE COMMAND
3868 016046 004037 017652 JSR      R0,WRT.RP  ;WRITE THE COMMAND INTO RPCS1
3869 016052 000000      RPCS1      ;REGISTER INDEX
3870 016054 016414      SC8      ;PARITY EXIT ADDRESS
3871 016056 011605      MOV      (SP),R5 ;PICKUP (RPAS) BEFORE THE ERROR CALL
3872 016060 004037 020764 JSR      R0,ES.SAV ;SAVE THE ADDRESS IN '$ESCAPE'
3873 016064 104002      ERROR     2      ;REPORT THE UNEXPECTED ATTENTION
3874 016066 000677      BR       SC4      ;GO CHECK FOR MORE ATA'S
3875 016070
3876 016070 004037 020764 8$:      JSR      R0,ES.SAV ;SAVE THE ADDRESS IN '$ESCAPE'
3877 016074 104005      ERROR     5      ;REPORT THE ADDRESS PLUG CHANGE
3878 016076 000673      BR       SC4      ;CHECK FOR MORE DRIVES
3879 016100 006301      SC6:      ASL      R1      ;SETUP TO ADDRESS WORDS
3880 016102 012761 177777 012440 MOV      #-1,TIMER(R1) ;STOP THE TIMER
3881 016110 006201      ASR      R1      ;RESTORE THE DRIVE ADDRESS
3882 016112 004737 020700 JSR      PC,GETREQ ;GET THE DPB POINTER FROM THE QUEUE
3883 016116 010164 000010 MOV      R1,RPCS2(R4) ;SELECT DRIVE
3884 016122 004037 017472 JSR      R0,RD.RP  ;READ THE RP04'S STATUS REG.
3885 016126 000012      RPDS1      ;
3886 016130 016414      SC8      ;
3887 016132 011605      MOV      (SP),R5 ;AND PUT IT IN R5
3888 016134 006126      ROL      (SP)+ ;WAS THERE AN ERROR?
3889 016136 100407      BMI      1$      ;BR IF ERROR
3890 016140 105761 012336      TSTB     DRVACT(R1) ;CHECK DRIVE'S STATE
3891 016144 003137      BGT      SC11      ;BR IF DRIVE ACTIVE WITH ORDER
3892 016146 052762 100210 000016 BIS      #BIT15:BIT07:BIT03,16(R2) ;INFORM USER OF ERROR RECOVER COMPLETION
3893 016154 000470      BR       SC7      ;
3894 016156 004037 017472 1$:      JSR      R0,RD.RP  ;READ ERROR REGISTER #1
3895 016162 000014      RPER1      ;
3896 016164 016414      SC8      ;
3897 016166 012605      MOV      (SP)+,R5 ;AND SAVE IT IN R5
3898 016170 004737 020044 JSR      PC,SVRH11 ;SAVE RH11/RP04/5/6 REGISTERS
3899 016174 012746 000111 MOV      #111,-(SP) ;ISSUE A DRIVE CLEAR
3900 016200 004037 017652 JSR      R0,WRT.RP  ;
3901 016204 000000      RPCS1      ;
3902 016206 016414      SC8      ;
3903 016210 006105      SC6A:      ROL      R5      ;WAS 'UNSAFE' CONDITION =1?
3904 016212 100406      BMI      1$      ;BRANCH IF YES
3905 016214 005702      TST      R2      ;ANYTHING IN QUEUE ?
3906 016216 001447      BEQ      SC7      ;BR IF NOT
3907 016220 052762 100240 000016 BIS      #BIT15:BIT07:BIT05,16(R2) ;INFORM USER OF ERROR
3908 016226 000443      BR       SC7

```



```

3909 016230 004037 017472      1$: JSR      R0,RD.RP      ;READ DRIVE STATUS REG. #1
3910 016234 000012              RPDS1
3911 016236 016414              SC8
3912 016240 011605              MOV      (SP),R5      ;SAVE RPDS1 IN R5
3913 016242 006126              ROL      (SP)+      ;'ERR'=1?
3914 016244 100011              BPL      2$      ;BR IF NO--UNSAFE CLEARED
3915 016246 112761 177777 012346 MOVB     #-1,DRVSTA(R1) ;DRIVE IS UNSAFE
3916 016254 004737 020044      JSR      PC,SVRH11    ;SAVE RH11/RP04/5/6 REGISTERS
3917 016260 052762 110000 000016 BIS      #BIT15!BIT12,16(R2) ;INFORM USER OF UNSAFE ERROR
3918 016266 000423              BR      SC7
3919 016270 032705 010000      2$: BIT      #BIT12,R5      ;'MOL' = 1 ?
3920 016274 001015              BNE      3$      ;BR IF YES
3921 016276 112761 177777 012336 MOVB     #-1,DRVACT(R1) ;ACTIVE ERROR RECOVER
3922 016304 112761 000001 012346 MOVB     #1,DRVSTA(R1) ;ONLINE
3923 016312 006301              ASL      R1
3924 016314 012761 072460 012440 MOV      #30000.,TIMER(R1) ;START 30 SECOND TIMER
3925 016322 006201              ASR      R1
3926 016324 000137 015666      JMP      SC4
3927 016330 052762 100220 000016 3$: BIS      #BIT15!BIT07!BIT04,16(R2) ;INFORM USER OF ERROR
3928 016336 105061 012336      SC7: CLRB     DRVACT(R1) ;DRIVE IS IDLE
3929 016342 004737 020604      JSR      PC,EMPTYQ    ;DUMP THE QUEUE
3930 016346 105761 012414      TSTB     ULDFLG(R1) ;UNLOAD IN PROGRESS OR QUEUE?
3931 016352 003002              BGT      1$      ;BR IF NOT
3932 016354 105061 012414      CLRB     ULDFLG(R1) ;CLEAR UNLOAD FLAG
3933 016360 116164 012462 000016 1$: MOVB     ATABIT(R1),RPAS(R4) ;CLEAR ATTENTION BIT
3934 016366 105761 012346      TSTB     DRVSTA(R1) ;IS THE DRIVE UNSAFE ?
3935 016372 100406              BMI      2$      ;BR IF IT IS
3936 016374 012746 000113      MOV      #113,-(SP) ;RELEASE COMMAND
3937 016400 004037 017652      JSR      R0,WRT.RP    ;WRITE THE COMMAND INTO RPCS1
3938 016404 000000              RPCS1
3939 016406 016414              SC8
3940 016410 000137 015666      2$: JMP      SC4
3941 016414 105761 012336      SC8: TSTB     DRVACT(R1) ;IS DRIVE IDLE?
3942 016420 001405              BEQ      1$      ;YES--BRANCH
3943 016422 004737 020700      JSR      PC,GETREQ    ;GET DPB POINTER
3944 016426 004737 014714      JSR      PC,C17      ;PROCESS THE PARITY ERROR
3945 016432 000402              BR      2$      ;CONTINUE
3946 016434 004737 014742      1$: JSR      PC,C17B     ;PROCESS THE UNCORRECTABLE PARITY ERROR
3947 016440 000137 015666      2$: JMP      SC4
3948 016444 105761 012414      SC11: TSTB     ULDFLG(R1) ;'UNLOAD IN PROGRESS'?
3949 016450 003402              BLE      1$      ;BRANCH IF NO
3950 016452 105061 012414      CLRB     ULDFLG(R1) ;CLEAR UNLOAD FLAG
3951 016456 105061 012336      1$: CLRB     DRVACT(R1) ;SET DRIVE IDLE
3952 016462 136137 012462 012410 BITB     ATABIT(R1),SRCHWT ;DOING A SEARCH OPERATION FOR
3953                                ;AN I/O COMMAND?
3954 016470 001012              BNE      2$      ;BRANCH IF YES
3955 016472 004737 020722      JSR      PC,POPQUE    ;REMOVE REQUEST FROM QUEUE
3956 016476 052762 000200 000016 BIS      #BIT07,16(R2) ;SET 'DONE' BIT
3957 016504 005737 012434      TST      SAVEFG      ;SAVE THE REGISTERS?
3958 016510 100002              BPL      2$      ;BRANCH IF NO
3959 016512 004737 020044      JSR      PC,SVRH11    ;YES--SAVE ALL OF THE RH11/RP04/5/6 REG'S
3960 016516 116164 012462 000016 2$: MOVB     ATABIT(R1),RPAS(R4) ;CLEAR ATTENTION BIT
3961 016524 004737 013604      JSR      PC,OPT      ;START A REQUEST
3962 016530 000137 015666      JMP      SC4
3963 016534 010164 000010      SC12: MOV      R1,RPCS2(R4) ;SELECT DRIVE
3964 016540 016437 000012 012326 MOV      RPDS1(R4),RPERRS ;SAVE THE FOUR REGISTERS THAT

```



```

3965 016546 016437 000014 012330      MOV      RPER1(R4),RPERRS+2      ;WILL TELL US SOMETHING
3966 016554 016437 000040 012332      MOV      RPER2(R4),RPERRS+4
3967
3968 016562 016437 000042 012334      MOV      RPER3(R4),RPERRS+6
3969 016570 004037 012724              JSR      R0,DRVINT      ;INIT. THE STATE OF THE DRIVE
3970 016574 000401              BR       1$            ;TAKE ERROR EXIT
3971 016576 000207              RTS      PC              ;RETURN
3972 016600 005726              1$:    TST      (SP)+      ;POP PC OFF OF THE STACK
3973 016602 000704              BR       SC8          ;PROCESS THE PARITY ERROR
3974 016604 006301              SC13:  ASL      R1        ;SETUP TO ADDRESS WORDS
3975 016606 012761 177777 012440      MOV      #-1,TIMER(R1) ;STOP THE TIMER
3976 016614 006201              ASR      R1
3977 016616 010164 000010      MOV      R1,RPCS2(R4)  ;SELECT THE DRIVE
3978 016622 116164 012462 000016      MOV      ATABIT(R1),RPAS(R4) ;CLEAR THE ATTENTION BIT
3979 016630 032714 004000      BIT      #BIT11,(R4)  ;DRIVE AVAILABLE ?
3980 016634 001006              BNE     1$            ;BR IF AVAILABLE
3981 016636 006301              ASL      R1
3982
3983 016640 012761 023420 012440      MOV      #10000.,TIMER(R1) ;START 10 SEC TIMER AGAIN
3984
3985 016646 006201              ASR      R1
3986 016650 000433              BR       3$
3987 016652 105761 012366              1$:    TSTB     DPINT(R1) ;INITIALIZING THE DRIVE ?
3988 016656 001424              BEQ     2$            ;BR IF NOT
3989 016660 105061 012366              CLRB     DPINT(R1)    ;CLEAR THE INIT INDICATOR
3990 016664 004037 012724              JSR      R0,DRVINT    ;GO INIT THE DRIVE
3991 016670 000240              NOP
3992 016672 105761 012346              TSTB     DRVSTA(R1)   ;DRIVE ONLINE ?
3993 016676 003014              BGT     2$            ;BR IF YES -- START ORDER
3994 016700 005702              TST      R2          ;QUEUE ENTRY FOR THE DRIVE
3995 016702 001416              BEQ     3$            ;BR IF NOT
3996 016704 004737 020700              JSR      PC,GETREQ    ;GET DPB ADDRESS
3997 016710 052762 140000 000016      BIS      #BIT15:BIT14,16(R2) ;INFORM USER THAT DRIVE OFFLINE
3998 016716 004737 020044              JSR      PC,SVRH11    ;SAVE THE REGISTERS
3999 016722 004737 020604              JSR      PC,EMPTYQ    ;EMPTY THE REQUEST QUEUE
4000 016726 000404              BR       3$
4001 016730 105061 012376              2$:    CLRB     DPRQS(R1) ;CLEAR THE PORT REQUEST INDICATOR
4002 016734 004737 013604              JSR      PC,OPT      ;START THE PENDING REQUEST
4003 016740 000137 015666              3$:    JMP      SC4    ;PROCESS OTHER DRIVES
4004
4005      ;RP04/5/6 TIMER ROUTINE
4006      ;CALL
4007      ;
4008      ;      MOV      #TIME, -(SP) ;ELAPSED TIME IN MILLISECONDS ON THE STACK
4009      ;      JSR      PC,RPTMR    ;CALL RP04/5/6 TIME ROUTINE
4010 016744 005737 012412              RPTMR: TST      ACTDRV ;CHECK 'ACTDRV & ACTSTR'
4011 016750 001030              BNE     4$            ;IF NON ZERO EXIT
4012 016752 112737 000001 012413      MOV      #1,ACTSTR    ;SET 'ACTSTR'
4013 016760 104412              SAVREG ;SAVE R0 - R5
4014 016762 005001              CLR      R1          ;START WITH DRIVE 0
4015 016764 005003              CLR      R3
4016 016766 005763 012440              1$:    TST      TIMER(R3) ;IS THE TIMER RUNNING?
4017 016772 002407              BLT     2$            ;BRANCH IF NO
4018 016774 166663 000002 012440      SUB      2(SP),TIMER(R3) ;COUNT THE INTERVAL
4019 017002 003003              BGT     2$            ;BR IF NO SOFTWARE TIMEOUT
4020 017004 004737 017036              JSR      PC,STO      ;CALL SOFTWARE TIMEOUT ROUTINE

```



```

4021 017010 000405
4022 017012 005201
4023 017014 005723
4024 017016 022701 000010
4025 017022 003361
4026 017024 104413
4027 017026 105037 012413
4028 017032 012616
4029 017034 000207
4030
4031
4032
4033
4034
4035
4036
4037
4038
4039
4040
4041 017036 010146
4042 017040 010346
4043 017042 013704 012474
4044 017046 010164 000010
4045 017052 004037 017472
4046 017056 000012
4047 017060 017360
4048 017062 105726
4049 017064 100477
4050 017066 105761 012366
4051 017072 001074
4052 017074 105761 012376
4053 017100 001071
4054 017102 013702 012406
4055 017106 020137 012460
4056 017112 001402
4057 017114 004737 020700
4058 017120 052762 101000 000016
4059 017126 004737 020044
4060 017132 012764 000040 000010
4061 017140 105061 012336
4062 017144 105061 012414
4063 017150 005001
4064 017152 005003
4065 017154 004037 012724
4066 017160 000477
4067 017162 105761 012336
4068 017166 001414
4069 017170 013702 012406
4070 017174 023701 012460
4071 017200 001402
4072 017202 004737 020700
4073 017206 052762 100400 000016
4074 017214 105061 012336
4075 017220 105061 012414
4076 017224 012763 177777 012440

2$: BR 3$ ;GO TO THE EXIT
INC R1 ;MOVE TO NEXT DRIVE
TST (R3)+
CMP #8.,R1 ;OUT OF DRIVES?
BGT 1$ ;BRANCH IF NO
3$: RESREG ;RESTORE R0 - R5
CLRB ACTSTR ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
4$: MOV (SP)+,(SP) ;ADJUST THE STACK
RTS PC ;RETURN

;SOFTWARE TIMEOUT ROUTINE
;NOTE: THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
;OR GREATER
;CALL: STO
;MOV #DRVNUM,R1 ;DRIVE NUMBER
;JSR PC,STO ;CALL
;RETURN
STO: MOV R1,-(SP) ;SAVE R1
MOV R3,-(SP) ;SAVE R3
MOV RPADR,R4 ;GET ADDRESS OF 'RPCS1'
MOV R1,RPCS2(R4) ;SELECT THE DRIVE
JSR R0,RD.RP ;READ 'DRIVE STATUS REG'
RPDS1
STO5
TSTB (SP)+ ;IS 'DRY'=1?
BMI STO2 ;BR IF YES
STO1: TSTB DPINT(R1) ;TRYING TO INITIALIZE THE DRIVE ?
BNE STO2 ;BR IF YES
TSTB DPRQS(R1) ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
BNE STO2 ;BR IF YES
MOV TRNSWT,R2 ;PICKUP TRANSFER WAIT QUEUE
CMP R1,DTUW ;TRANSFER UNDERWAY ON THIS DRIVE?
BEQ 1$ ;BRANCH IF YES
JSR PC,GETREQ ;GET DPB ADDRESS
1$: BIS #BIT15!BIT09,16(R2) ;SET THE ERROR FLAGS
JSR PC,SVRH11 ;SAVE RH11/RP04/5/6 REGISTERS
MOV #BIT05,RPCS2(R4) ;'INIT' THE MASS BUS
CLRB DRVACT(R1) ;DRIVE IS IDLE
CLRB ULDFLG(R1) ;CLEAR THE UNLOAD FLAG
CLR R1 ;START WITH DRIVE 0
CLR R3
2$: JSR R0,DRVINT ;INIT. THIS DRIVE
BR STO5 ;PARITY ERROR RETURN
TSTB DRVACT(R1) ;DRIVE IDLE BEFORE THE INIT.?
BEQ 4$ ;YES--BRANCH
MOV TRNSWT,R2 ;GET TRANSFER WAIT QUEUE
CMP DTUW,R1 ;WAS THERE I/O ON THIS DRIVE?
BEQ 3$ ;YES--BRANCH
JSR PC,GETREQ ;GET THE DPB POINTER FROM QUEUE
3$: BIS #BIT15!BIT08,16(R2) ;INFORM USER OF INIT.
CLRB DRVACT(R1) ;SET DRIVE ACTIVE TO IDLE
CLRB ULDFLG(R1) ;NO UNLOAD
MOV #-1,TIMER(R3) ;STOP THE TIMER

```



```

4077 017232 005723          TST      (R3)+      ;UPDATE THE INDEX
4078 017234 005201          INC      R1          ;INCREMENT THE DRIVE NUMBER
4079 017236 022701 000010    CMP      #8.,R1      ;LAST DRIVE BEEN CHECKED?
4080 017242 003344          BGT      2$          ;NO--LOOP
4081 017244 012737 177777 012460 MOV     #-1,DTUW      ;NO DATA TRANSFERS UNDERWAY
4082 017252 005037 012406    CLR      TRNSWT      ;CLEAR TRANSFER WAIT QUEUE
4083 017256 004737 020526    JSR      PC,CLRQUE    ;CLEAR ALL REQUEST QUEUES
4084 017262 000500          BR       ST09         ;EXIT
4085 017264 116405 000016    ST02:  MOVB     RPAS(R4),R5    ;READ ATTENTION REG
4086 017270 136105 012462    BITB     ATABIT(R1),R5    ;IS ATTENTION FOR THIS DRIVE UP ?
4087 017274 001017          BNE      ST03         ;YES--BRANCH
4088 017276 105761 012366    TSTB     DPINT(R1)      ;TRYING TO INITIALIZE THE DRIVE ?
4089 017302 001031          BNE      ST06         ;BR IF YES - DRIVE NOT ONLINE
4090 017304 105761 012376    TSTB     DPRQS(R1)      ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
4091 017310 001045          BNE      ST07         ;BR IF YES - NO RESPONSE TO REQUEST
4092 017312 020137 012460    CMP      R1,DTUW      ;DATA TRANSFER UNDERWAY FOR THIS DRIVE
4093 017316 001263          BNE      ST01         ;BR IF NO
4094 017320 004037 017472    JSR      R0,RD.RP      ;YES--CHECK 'RDY'
4095 017324 000000          RPCS1
4096 017326 017360          ST05
4097 017330 105726          TSTB     (SP)+
4098 017332 100255          BPL      ST01         ;BR IF 'RDY'=0
4099 017334 105761 012366    ST03:  TSTB     DPINT(R1)      ;INITIALIZING THE DRIVE ?
4100 017340 001003          BNE      1$          ;BR IF INIT PENDING
4101 017342 105761 012376    TSTB     DPRQS(R1)      ;PORT REQUEST PENDING ?
4102 017346 001446          BEQ      ST09         ;BR IF NOT
4103 017350 012763 177777 012440 1$:  MOV     #-1,TIMER(R3) ;STOP THE TIMER
4104 017356 000442          BR       ST09         ;EXIT
4105 017360 004737 015014    ST05:  JSR      PC,C18      ;GO HANDLE THE PARITY ERROR
4106 017364 000437          BR       ST09
4107 017366 105061 012366    ST06:  CLRB     DPINT(R1)      ;CLEAR THE INITIALIZE INDICATOR
4108 017372 105061 012346    CLRB     DRVSTA(R1)      ;SET UNIT OFFLINE
4109 017376 012763 177777 012440 MOV     #-1,TIMER(R3) ;STOP THE TIMER
4110 017404 004737 020700    JSR      PC,GETREQ    ;GET THE DPB ADDRESS
4111 017410 005702          TST      R2          ;REQUEST IN QUEUE ?
4112 017412 001424          BEQ      ST09         ;BR IF NOT
4113 017414 052762 140000 000016 BIS     #BIT15!BIT14,16(R2) ;INFORM THE USER DRIVE NOT AVAILABLE
4114 017422 000414          BR       ST08         ;FINISH
4115 017424 012763 177777 012440 ST07: MOV     #-1,TIMER(R3) ;STOP THE TIMER
4116 017432 105061 012376    CLRB     DPRQS(R1)      ;CLEAR PORT REQUEST INDICATOR
4117 017436 004737 020700    JSR      PC,GETREQ    ;GET DPB ADDRESS
4118 017442 005702          TST      R2          ;QUEUE ENTRY FOR DRIVE ?
4119 017444 001407          BEQ      ST09         ;BR IF NONE
4120 017446 012762 100004 000016 MOV     #BIT15!BIT2,16(R2) ;INFORM USER OF PORT REQUEST ERROR
4121 017454 004737 020604    ST08:  JSR      PC,EMPTYQ    ;CLEAR THE QUEUE FOR THE DRIVE
4122 017460 004737 020044    JSR      PC,SVRH11    ;SAVE THE REGISTERS
4123 017464 012603          ST09:  MOV     (SP)+,R3      ;RESTORE R3
4124 017466 012601          MOV     (SP)+,R1      ;RESTORE R1
4125 017470 000207          RTS      PC          ;RETURN
4126
4127 ;ROUTINE TO READ A RH11/RP04/5/6 REGISTER
4128
4129 ;CALL
4130 ;JSR      R0,RD.RP      ;GO READ A REGISTER
4131 ;INDEX
4132 ;ERRADR      ;REG. INDEX FROM BASE
;ERROR ADDRESS--PROCESS ERROR STARTING

```



```

4133      :
4134      : RETURN
4135      :
4136 017472 013737 012472 017640 RD.RP: MOV MCPEMX, RD.RP2 ;MAX. RETRYS ALLOWED
4137 017500 011646          MOV (SP), -(SP) ;SAVE R0 FOR RETURN
4138 017502 013737 012474 017516          MOV RPADR, RD.ADR ;FORM THE DESIRED ADDRESS
4139 017510 062037 017516          ADD (R0)+, RD.ADR ;USING THE BASE AND THE INDEX
4140 017514 013727          RD.RP1: MOV @ (PC)+, (PC)+ ;READ THE DESIRED REGISTER OF THE RP04
4141 017516 000000          RD.ADR: .WORD 0 ;ADDRESS IS FORMED HERE
4142 017520 000000          RD.WRD: .WORD 0 ;REG. CONTENTS PUT HERE
4143 017522 013766 017520 000002          MOV RD.WRD, 2(SP) ;RETURN IT TO THE USER
4144 017530 013746 012474          MOV RPADR, -(SP) ;PUT THE ADDRESS ON THE STACK
4145 017534 062716 000010          ADD #RPCS2, (SP) ;FORM THE ADDRESS OF RPCS2
4146 017540 032736 010000          BIT #BIT12, @ (SP)+ ;CHECK THE 'NED' BIT
4147 017544 001037          BNE RD.RP3 ;BR IF DRIVE NON-EXISTENT
4148 017546 017746 172722          MOV @RPADR, -(SP) ;READ RPCS1
4149 017552 032716 020000          BIT #BIT13, (SP) ;DID MCPE SET?
4150 017556 001002          BNE 1$ ;BRANCH IF YES
4151 017560 022620          CMP (SP)+, (R0)+ ;ADJUST FOR RETURN
4152 017562 000432          BR RD.RP4 ;EXIT
4153 017564          1$:
4154 017564 004037 020764          JSR R0, ES.SAV ;SAVE THE ADDRESS IN '$ESCAPE'
4155 017570 104003          ERROR 3 ;REPORT 'MCPE' ERROR
4156 017572 005737 012460          TST DTUW ;DATA TRANSFER UNDERWAY?
4157 017576 100405          BMI 2$ ;NO--BRANCH
4158 017600 032716 040000          BIT #BIT14, (SP) ;NO--'TRE'=1?
4159 017604 001402          BEQ 2$ ;NO--BRANCH
4160 017606 005726          TST (SP)+ ;YES--CLEAN OFF THE STACK AND
4161 017610 000415          BR RD.RP3 ;TAKE THE FATAL ERROR EXIT
4162 017612 052716 040000          2$: BIS #BIT14, (SP) ;CLEAR 'MCPE' BY SENDING A '1' TO 'TRE'
4163 017616 000316          SWAB (SP) ;POSITION BEFORE WRITING
4164 017620 013737 012474 017634          MOV RPADR, 3$ ;FORM ADDRESS OF HIGH BYTE
4165 017626 005237 017634          INC 3$
4166 017632 112637          MOVAB (SP)+, @ (PC)+ ;WRITE THE HIGH BYTE OF RPCS1
4167 017634 000000          3$: .WORD 0 ;ADDRESS STORAGE
4168 017636 005327          DEC (PC)+ ;EXCEEDED MAX. RETRYS
4169 017640 000003          RD.RP2: .WORD 3
4170 017642 002324          BGE RD.RP1 ;BRANCH IF NO
4171 017644 011000          RD.RP3: MOV (R0), R0 ;FATAL ERROR EXIT
4172 017646 012616          MOV (SP)+, (SP)
4173 017650 000200          RD.RP4: RTS R0
4174
4175      ;ROUTINE TO WRITE A REGISTER
4176      :
4177      : CALL
4178      :
4179      : MOV DATA, -(SP) ;DATA TO BE LOADED ON THE STACK
4180      : JSR R0, WRT.RP ;CALL THE ROUTINE TO LOAD(WRITE) THE REG.
4181      : INDEX ;INDEX OF THE REGISTER TO BE LOADED
4182      : ERRADR ;ADDRESS TO RETURN TO ON AN ERROR
4183      : RETURN ;ERROR FREE RETURN
4184 017652 013737 012472 020026 WRT.RP: MOV MCPEMX, WRT.R2 ;MAX RETRYS ALLOWED
4185 017660 016637 000002 017740          MOV 2(SP), WRT.WD ;SAVE THE WORD TO WRITE
4186 017666 012616          MOV (SP)+, (SP) ;ADJUST THE STACK
4187 017670 012037 017742          MOV (R0)+, WRT.AD ;GET INDEX OF REGISTER TO BE WRITTEN
4188 017674 001015          BNE 1$ ;BRANCH IF NOT RPCS1

```


Address	Offset	Hex	Label	Register	Operation	Comment	
4189	017676	122737	000150	017740	CMPB	#150,WRT.WD	:IS THE COMMAND FOR DATA TRANSFERS?
4190	017704	002411			BLT	1\$	:YES--DON'T GET THE OLD A16 & A17, & PSEL
4191	017706	004037	017472		JSR	R0,RD.RP	:NO---COMBINE A16&A17, & PSEL WITH
4192	017712	000000			RPCS1		:THE COMMAND BEFORE SENDING IT TO
4193	017714	020034			WRT.R3		:THE RH11/RP04
4194	017716	000316			SWAB	(SP)	
4195	017720	042716	177770		BIC	#^C7,(SP)	
4196	017724	112637	017741		MOVB	(SP)+,WRT.WD+1	
4197	017730	063737	012474	017742	1\$: ADD	RPADR,WRT.AD	:FORM THE ADDRESS OF THE DISK REG.
4198	017736	012737			WRT.R1: MOV	(PC)+,@(PC)+	:LOAD THE DESIRED REG.
4199	017740	000000			WRT.WD: .WORD	0	:WORD TO WRITE GOES HERE
4200	017742	000000			WRT.AD: .WORD	0	:ADDRESS IS FORMED HERE
4201	017744	013746	012474		MOV	RPADR,-(SP)	:PUT THE ADDRESS ON THE STACK
4202	017750	062716	000010		ADD	#RPCS2,(SP)	:FORM THE ADDRESS OF RPCS2
4203	017754	032736	010000		BIT	#BIT12,@(SP)+	:CHECK THE 'NED' BIT
4204	017760	001025			BNE	WRT.R3	:BR IF DRIVE NON-EXISTENT
4205	017762	004037	017472		JSR	R0,RD.RP	:CHECK FOR PARITY ERROR ON WRITE
4206	017766	000014			RPER1		
4207	017770	020034			WRT.R3		
4208	017772	032726	000010		BIT	#BIT03,(SP)+	
4209	017776	001420			BEQ	WRT.R4	:BRANCH IF 'PAR=0'
4210	020000	016037	177776	020012	MOV	-2(R0),1\$	:PICKUP THE INDEX
4211	020006	004037	017472		JSR	R0,RD.RP	:READ THE REG.
4212	020012	000000			1\$: .WORD	0	:REG. INDEX
4213	020014	020034			WRT.R3		:RETURN TO THIS ADDRESS ON ERROR
4214	020016	004037	020764		JSR	R0,ES.SAV	:SAVE THE ADDRESS IN '\$ESCAPE'
4215	020022	104004			ERROR	4	:REPORT THE PARITY ON WRITE ERROR
4216	020024	005327			DEC	(PC)+	:DECREMENT THE ERROR COUNT
4217	020026	000003			WRT.R2: .WORD	3	:RETRY COUNTER
4218	020030	002342			BGE	WRT.R1	:TRY AGAIN IF NOT FINISHED
4219	020032	005726			TST	(SP)+	:CLEAN OFF THE STACK
4220	020034	011000			WRT.R3: MOV	(R0),R0	:TAKE THE 'PARITY ON WRITE' ERROR EXIT
4221	020036	000401			BR	WRT.R5	:EXIT
4222	020040	005720			WRT.R4: TST	(R0)+	:ADJUST FOR ERROR FREE EXIT
4223	020042	000200			WRT.R5: RTS	R0	
4224							
4225							:ROUTINE TO SAVE THE RH11/RP04/5/6 REGISTERS AS PER DPB+14
4226							
4227							:CALL
4228							
4229					MOV	#DPBNUM,R2	:DPB POINTER TO R2
4230					JSR	PC,SVRH11	:SAVE THE DRIVES REG'S
4231	020044	104412			SVRH11: SAVREG		:SAVE R0 - R5
4232	020046	005702			TST	R2	:QUEUE ENTRY FOR THE DRIVE ?
4233	020050	001430			BEQ	4\$	:BR IF NONE
4234	020052	013704	012474		MOV	RPADR,R4	
4235	020056	111264	000010		MOVB	(R2),RPCS2(R4)	:SELECT DRIVE
4236	020062	016203	000014		MOV	14(R2),R3	:GET THE ERROR TABLE POINTER
4237	020066	001433			BEQ	6\$	:EXIT IF NO ADDRESS
4238	020070	005037	020124		CLR	3\$	:COUNTER & POINTER
4239							


```

4245 020120 004037 017472      2$: JSR    R0,RD.RP      ;READ THE SELECTED REGISTER
4246 020124 000000      3$:      .WORD    0      ;REGISTER INDEX
4247 020126 020152      5$:      ;ERROR RETURN ADDRESS
4248 020130 012623      MOV    (SP)+,(R3)+      ;STORE THE REGISTER CONTENTS
4249 020132 023727 020124 000046 4$: CMP    3$,#RPEC2      ;REACHED THE END ?
4250 020140 001406      BEQ    6$      ;BR IF YES
4251 020142 062737 000002 020124 ADD    #2,3$      ;INCREMENT THE REGISTER INDEX
4252 020150 000751      BR     1$      ;CONTINUE READING THE REGISTERS
4253 020152 004737 014714      5$: JSR    PC,C17      ;PROCESS THE UNCORRECTABLE PARITY ERROR
4254 020156 104413      6$: RESREG      ;RESTORE R0 - R5
4255 020160 000207      RTS     PC      ;RETURN
4256
4257      ;ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A 'TRE'
4258      ;CALL
4259      ;      MOV    #DRVNUM,R1      ;DRIVE NUMBER TO R1
4260      ;      JSR    PC,SET.IE      ;SET 'IE'
4261      ;
4262      ;
4263 020162 010446      SET.IE: MOV    R4,-(SP)      ;SAVE R4
4264 020164 013704 012474      MOV    RPADR,R4      ;PICKUP ADDRESS OF RPCS1
4265 020170 010164 000010      MOV    R1,RPCS2(R4)      ;SELECT DRIVE
4266 020174 011446      MOV    (R4),-(SP)      ;READ RPCS1
4267 020176 052716 040000      BIS    #BIT14,(SP)      ;SET THE 'TRE' BIT OF THE WORD READ
4268 020202 000316      SWAB    (SP)      ;ADJUST FOR DATO
4269 020204 112714 000100      MOVB    #BIT06,(R4)      ;SET 'IE'
4270 020210 032764 010000 000010 BIT    #BIT12,RPCS2(R4) ;IS 'NED'=1?
4271 020216 001002      BNE    1$      ;YES--CLEAR 'TRE'
4272 020220 005726      TST    (SP)+      ;CLEAN OFF THE STACK
4273 020222 000402      BR     2$
4274 020224 112664 000001      1$: MOVB    (SP)+,1(R4)      ;CLEAR 'TRE'
4275 020230 012604      2$: MOV    (SP)+,R4      ;RESTORE R4
4276 020232 000207      RTS     PC      ;RETURN TO CALLER
4277
4278      ;QUEUE COUNT
4279 020234      000      QCNT: .BYTE    0      ;DRIVE 0
4280 020235      000      .BYTE    0      ;DRIVE 1
4281 020236      000      .BYTE    0      ;DRIVE 2
4282 020237      000      .BYTE    0      ;DRIVE 3
4283 020240      000      .BYTE    0      ;DRIVE 4
4284 020241      000      .BYTE    0      ;DRIVE 5
4285 020242      000      .BYTE    0      ;DRIVE 6
4286 020243      000      .BYTE    0      ;DRIVE 7
4287
4288      ;QUEUE INPUT POINTERS
4289
4290 020244 020326      QINPT: .WORD    QDRV0      ;DRIVE 0
4291 020246 020346      .WORD    QDRV1      ;DRIVE 1
4292 020250 020366      .WORD    QDRV2      ;DRIVE 2
4293 020252 020406      .WORD    QDRV3      ;DRIVE 3
4294 020254 020426      .WORD    QDRV4      ;DRIVE 4
4295 020256 020446      .WORD    QDRV5      ;DRIVE 5
4296 020260 020466      .WORD    QDRV6      ;DRIVE 6
4297 020262 020506      .WORD    QDRV7      ;DRIVE 7
4298
4299      ;QUEUE OUTPUT POINTERS
4300

```



```

4301 020264 020326      QOUTPT: .WORD  QDRV0      ;DRIVE 0
4302 020266 020346      .WORD  QDRV1      ;DRIVE 1
4303 020270 020366      .WORD  QDRV2      ;DRIVE 2
4304 020272 020406      .WORD  QDRV3      ;DRIVE 3
4305 020274 020426      .WORD  QDRV4      ;DRIVE 4
4306 020276 020446      .WORD  QDRV5      ;DRIVE 5
4307 020300 020466      .WORD  QDRV6      ;DRIVE 6
4308 020302 020506      .WORD  QDRV7      ;DRIVE 7
4309
4310 020304 020326      QSTART: .WORD  QDRV0      ;DRIVE 0 START ADDRESS
4311 020306 020346      QSTOP:  .WORD  QDRV1      ;DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
4312 020310 020366      .WORD  QDRV2      ;STOP DRIVE 1--START DRIVE 2
4313 020312 020406      .WORD  QDRV3      ;STOP DRIVE 2--START DRIVE 3
4314 020314 020426      .WORD  QDRV4      ;STOP DRIVE 3--START DRIVE 4
4315 020316 020446      .WORD  QDRV5      ;STOP DRIVE 4--START DRIVE 5
4316 020320 020466      .WORD  QDRV6      ;STOP DRIVE 5--START DRIVE 6
4317 020322 020506      .WORD  QDRV7      ;STOP DRIVE 6--START DRIVE 7
4318 020324 020526      .WORD  QTERM      ;STOP DRIVE 7
4319
4320      ;DRIVE REQUEST QUEUES
4321
4322 020326 000010      QDRV0:  .BLKW  10
4323 020346 000010      QDRV1:  .BLKW  10
4324 020366 000010      QDRV2:  .BLKW  10
4325 020406 000010      QDRV3:  .BLKW  10
4326 020426 000010      QDRV4:  .BLKW  10
4327 020446 000010      QDRV5:  .BLKW  10
4328 020466 000010      QDRV6:  .BLKW  10
4329 020506 000010      QDRV7:  .BLKW  10
4330      QTERM=.
4331
4332      ;ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
4333
4334      ;CALL
4335      ;      JSR      PC,CLRQUE
4336
4337 020526 104412      CLRQUE: SAVREG      ;SAVE R0 - R5
4338 020530 012702 020234      MOV      #QCNT,R2      ;ZERO THE QUEUE COUNTS
4339 020534 005022      CLR      (R2)+      ;DRIVES 0 & 1
4340 020536 005022      CLR      (R2)+      ;DRIVES 2 & 3
4341 020540 005022      CLR      (R2)+      ;DRIVES 4 & 5
4342 020542 005022      CLR      (R2)+      ;DRIVES 6 & 7
4343 020544 012703 000010      MOV      #8,R3      ;MOVE THE STARTING
4344 020550 012701 020304      MOV      #QSTART,R1      ;ADDRESS OF THE QUEUE INTO
4345 020554 012122      1$:  MOV      (R1)+,(R2)+      ;THE QUEUE INPUT POINTER
4346 020556 005303      DEC      R3
4347 020560 001375      BNE      1$
4348 020562 012703 000010      MOV      #8,R3      ;MOVE THE STARTING ADDRESS
4349 020566 012701 020304      MOV      #QSTART,R1      ;OF THE QUEUE INTO THE
4350 020572 012122      2$:  MOV      (R1)+,(R2)+      ;QUEUE OUTPUT POINTER
4351 020574 005303      DEC      R3
4352 020576 001375      BNE      2$
4353 020600 104413      RISRREG
4354 020602 000207      RTS      PC      ;RESTORE R0 - R5
4355
4356      ;EMPTY THE QUEUE SPECIFIED BY R1

```



```

4357
4358
4359
4360
4361
4362 020604 105061 020234
4363 020610 006301
4364 020612 016161 020244 020264
4365 020620 006201
4366 020622 000207
4367
4368
4369
4370
4371
4372
4373
4374
4375
4376
4377 020624 122761 000010 020234
4378 020632 001421
4379 020634 105261 020234
4380 020640 006301
4381 020642 010271 020244
4382 020646 062761 000002 020244
4383 020654 026161 020244 020306
4384 020662 001003
4385 020664 016161 020304 020244
4386 020672 006201
4387 020674 005720
4388 020676 000200
4389
4390
4391
4392
4393
4394
4395
4396
4397
4398 020700 005002
4399 020702 105761 020234
4400 020706 001404
4401 020710 006301
4402 020712 017102 020264
4403 020716 006201
4404 020720 000207
4405
4406
4407
4408
4409
4410
4411
4412

;CALL
;      MOV      DRVNUM,R1      ;DRIVE NUMBER TO R1
;      JSR      PC,EMPTYQ
EMPTYQ: CLR      QCNT(R1)      ;CLEAR NUMBER OF ITEMS IN QUEUE
        ASL      R1
        MOV      QINPT(R1),QOUTPT(R1) ;SET OUTPUT QUEUE POINTER=INPUT POINTER
        ASR      R1
        RTS      PC

;ROUTINE TO PUT A REQUEST IN QUEUE
;CALL
;      MOV      #DRVNUM,R1      ;DRIVE NUMBER
;      MOV      #DPB,R2        ;ADDRESS OF PARAMETER BLOCK
;      JSR      R0,DRVQUE      ;GO PUT REQUEST IN QUEUE
;      RETURN1     ;RETURN HERE IF QUEUE IS FULL
;      RETURN2     ;RETURN HERE IF REQUEST IS IN QUEUE

DRVQUE: CMP      #10,QCNT(R1)   ;IS QUEUE FULL?
        BEQ      2$            ;BR IF YES-TAKE RETURN1
        INCB     QCNT(R1)      ;INCREMENT QUEUE COUNT
        ASL      R1
        MOV      R2,@QINPT(R1) ;PUT THIS REQUEST IN QUEUE
        ADD      #2,QINPT(R1)  ;UPDATE THE QUEUE POINTER
        CMP      QINPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER
        BNE      1$            ;BRANCH IF NO
        MOV      QSTART(R1),QINPT(R1) ;YES--RESET POINTER
1$:      ASR      R1
        TST      (R0)+          ;TAKE RETURN 2
2$:      RTS      R0            ;RETURN TO USER

;ROUTINE TO GET THE 'DPB' ADDRESS OF NEXT REQUEST IN QUEUE
;CALL
;      MOV      #DRVNUM,R1      ;DRIVE NUMBER TO R1
;      JSR      PC,GETREQ      ;GO GET THE REQUEST
;      RETURN      ;R2='DPB' ADDRESS OF THE REQUEST
;                      ;R2=0 IF NO REQUEST IN QUEUE

GETREQ: CLR      R2
        TST      QCNT(R1)      ;IS THERE ANY REQUEST IN QUEUE?
        BEQ      2$            ;NO---BRANCH
1$:      ASL      R1
        MOV      @QOUTPT(R1),R2 ;PICKUP 'DPB' POINTER FOR THIS DRIVE
        ASR      R1
2$:      RTS      PC            ;RETURN TO USER

;ROUTINE TO 'POP' THE REQUEST FROM QUEUE
;CALL
;      MOV      #DRVNUM,R1      ;DRIVE NUMBER TO R1
;      JSR      PC,POPQUE      ;CALL TO REMOVE REQUEST
;      RETURN      ;R2=ADDRESS OF DPB REMOVED

```



```

4413 020722 105361 020234 POPQUE: DECB QCNT(R1) ;DECREMENT QUEUE COUNT
4414 020726 006301 ASL R1
4415 020730 017102 020264 MOV @QOUTPT(R1),R2 ;GET THE 'DPB' POINTER
4416 020734 062761 000002 020264 ADD #2,QOUTPT(R1) ;UPDATE THE QUEUE POINTER
4417 020742 026161 020264 020306 CMP QOUTPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER?
4418 020750 001003 BNE 1$ ;NO--BRANCH TO EXIT
4419 020752 016161 020304 020264 MOV QSTART(R1),QOUTPT(R1) ;YES--RESET THE POINTER
4420 020760 006201 1$: ASR R1
4421 020762 000207 RTS PC ;RETURN TO USER
4422
4423 ;ROUTINE TO SAVE THE CONTENTS OF '$ESCAPE' WHEN THE DRIVER
4424 ;REPORTS AN ERROR DIRECTLY.
4425
4426 ;CALL
4427 ;
4428 ; JSR RO,ES.SAV ;:THE ERROR CALL
4429 ; ERROR N ;:THE RETURN IS PAST THE ERROR CALL
4430 ; RETURN
4431 020764 012037 021000 ES.SAV: MOV (R0)+,1$ ;GET THE ERROR CALL
4432 020770 013746 001160 MOV $ESCAPE,-(SP) ;SAVE THE ADDRESS IN '$ESCAPE'
4433 020774 005037 001160 CLR $ESCAPE ;CLEAR THE ESCAPE RETURN
4434 021000 000000 1$: .WORD 0 ;THE ERROR CALL IS MOVED HERE
4435 021002 012637 001160 MOV (SP)+,$ESCAPE ;RESTORE THE ESCAPE ADDRESS
4436 021006 000200 RTS R0 ;RETURN
4437
4438
4439 ;:*****
4440
4441 .SBTTL TELETYPE MESSAGES
4442
4443 021010 005015 040527 047122 USE: .ASCIZ <CR><LF>/WARNING: PROGRAMMABLE DRIVES MAY BE USED/<CR><LF>
4444 021016 047111 035107 050040
4445 021024 047522 051107 046501
4446 021032 040515 046102 020105
4447 021040 051104 053111 051505
4448 021046 046440 054501 041040
4449 021054 020105 051525 042105
4450 021062 005015 000
4451 021065 040 051120 043517 NOUSE: .ASCIZ / PROGRAMMABLE-DRIVE WILL NOT BE USED/
4452 021072 040522 046515 041101
4453 021100 042514 042055 044522
4454 021106 042526 053440 046111
4455 021114 020114 047516 020124
4456 021122 042502 052440 042523
4457 021130 000104
4458 021132 047440 043106 044514 UNTOFF: .ASCIZ / OFFLINE/
4459 021140 042516 000
4460 021143 040 047117 044514 UNTON: .ASCIZ / ONLINE/
4461 021150 042516 000
4462 021153 040 047516 020124 NOTRP: .ASCIZ @ NOT AN RP04/5/6@
4463 021160 047101 051040 030120
4464 021166 027464 027465 000066
4465 021174 047040 052117 050040 NOTPRS: .ASCIZ / NOT PRESENT/
4466 021202 042522 042523 052116
4467 021210 000
4468 021211 040 047125 040523 NOTSAF: .ASCIZ / UNSAFE/

```


4469	021216	042506	000		
4470	021221	122	030120	000064	RP04B: .ASCIZ /RP04/
4471	021226	050122	032460	000	RP05: .ASCIZ /RP05/
4472	021233	122	030120	000066	RP06: .ASCIZ /RP06/
4473	021240	047125	052111	051440	SYSTAT: .ASCIZ /UNIT STATUS/<CR><LF><LF>
4474	021246	040524	052524	006523	
4475	021254	005012	000		
4476	021257	015	042012	044522	MUNIT: .ASCIZ <CR><LF>/DRIVE: /
4477	021264	042526	020072	000	
4478	021271	040	020057	000	SLASH: .ASCIZ @ / @
4479	021275	103	000		C: .ASCIZ /C/
4480	021277	124	000		T: .ASCIZ /T/
4481	021301	060	000		MDRVD: .ASCIZ /O/
4482	021303	040	040		LIN4SP: .ASCII / /
4483	021305	040	000040		LINSP: .ASCIZ / /
4484	021310	047105	042524	020122	ENTADR: .ASCIZ /ENTER ADDRESS LIMITS:/<CR><LF>
4485	021316	042101	051104	051505	
4486	021324	020123	044514	044515	
4487	021332	051524	006472	000012	
4488	021340	020040	051104	053111	MOFFLN: .ASCIZ / DRIVE OFFLINE/<CR><LF>
4489	021346	020105	043117	046106	
4490	021354	047111	006505	000012	
4491	021362	042040	044522	042526	MDRNP: .ASCIZ / DRIVE NOT PRESENT/<CR><LF>
4492	021370	047040	052117	050040	
4493	021376	042522	042523	052116	
4494	021404	005015	000		
4495	021407	040	051104	053111	MER11: .ASCIZ / DRIVE NOT AVAILABLE/<CR><LF>
4496	021414	020105	047516	020124	
4497	021422	053101	044501	040514	
4498	021430	046102	006505	000012	
4499	021436	042040	044522	042526	MNRP04: .ASCIZ @ DRIVE NOT AN RP04/5/6@<CR><LF>
4500	021444	047040	052117	040440	
4501	021452	020116	050122	032060	
4502	021460	032457	033057	005015	
4503	021466	000			
4504	021467	040	051104	053111	MUSDR: .ASCIZ / DRIVE UNSAFE/<CR><LF>
4505	021474	020105	047125	040523	
4506	021502	042506	005015	000	
4507	021507	040	042523	042514	MSELD: .ASCIZ / SELECTED/
4508	021514	052103	042105	000	
4509	021521	015	050012	047522	MMODE: .ASCIZ <CR><LF>/PROGRAM MODE (C OR F): /
4510	021526	051107	046501	046440	
4511	021534	042117	020105	041450	
4512	021542	047440	020122	024506	
4513	021550	020072	000		
4514	021553	040	047506	046522	MFORMAT: .ASCIZ / FORMAT & VERIFY/
4515	021560	052101	023040	053040	
4516	021566	051105	043111	000131	
4517	021574	044103	041505	020113	MHECK: .ASCIZ /CHECK ONLY/
4518	021602	047117	054514	000	
4519	021607	106	051117	040515	MORMAT: .ASCIZ /FORMAT & VERIFY/
4520	021614	020124	020046	042526	
4521	021622	044522	054506	000	
4522	021627	015	005012	050117	MSIZE: .ASCIZ <CR><LF><LF>/OPERATE IN 22 SECTOR MODE (Y OR N) ? /
4523	021634	051105	052101	020105	
4524	021642	047111	031040	020062	

4525	021650	042523	052103	051117	
4526	021656	046440	042117	020105	
4527	021664	054450	047440	020122	
4528	021672	024516	037440	000040	
4529	021700	050117	051105	052101	MSEC22: .ASCIZ /OPERATION WILL BE IN 22 SECTOR (16 BIT) MODE/<CR><LF>
4530	021706	047511	020116	044527	
4531	021714	046114	041040	020105	
4532	021722	047111	031040	020062	
4533	021730	042523	052103	051117	
4534	021736	024040	033061	041040	
4535	021744	052111	020051	047515	
4536	021752	042504	005015	000	
4537	021757	117	042520	040522	MSEC20: .ASCIZ /OPERATION WILL BE IN 20 SECTOR (18 BIT) MODE/<CR><LF>
4538	021764	044524	047117	053440	
4539	021772	046111	020114	042502	
4540	022000	044440	020116	030062	
4541	022006	051440	041505	047524	
4542	022014	020122	030450	020070	
4543	022022	044502	024524	046440	
4544	022030	042117	006505	000012	
4545	022036	047111	040526	044514	BADENT: .ASCIZ /INVALID ENTRY/<CR><LF>
4546	022044	020104	047105	051124	
4547	022052	006531	000012		
4548	022056	047105	044504	043516	MADRER: .ASCII /ENDING DSK ADRS MUST BE EQUAL TO OR GREATER/<CR><LF>
4549	022064	042040	045523	040440	
4550	022072	051104	020123	052515	
4551	022100	052123	041040	020105	
4552	022106	050505	040525	020114	
4553	022114	047524	047440	020122	
4554	022122	051107	040505	042524	
4555	022130	006522	012		
4556	022133	124	040510	020116	.ASCIZ /THAN STARTING ADRS/<CR><LF>
4557	022140	052123	051101	044524	
4558	022146	043516	040440	051104	
4559	022154	006523	000012		
4560	022160	042523	042514	052103	MSELP: .ASCII /SELECT DATA PATTERN/<CR><LF>
4561	022166	042040	052101	020101	
4562	022174	040520	052124	051105	
4563	022202	006516	012		
4564	022205	040	030050	020051	.ASCII / (0) ZERO'S/<CR><LF>
4565	022212	042532	047522	051447	
4566	022220	005015			
4567	022222	024040	024461	047440	.ASCII / (1) ONE'S/<CR><LF>
4568	022230	024516	051447	005015	
4569	022236	024040	024462	053440	.ASCIZ / (2) WORST CASE: /
4570	022244	051117	052123	041440	
4571	022252	051501	035105	000040	
4572					
4573	022260	047527	051522	020124	MPATD: .ASCIZ /WORST CASE/
4574	022266	040503	042523	000	
4575	022273	015	005012	052123	MSFOU: .ASCIZ <CR><LF><LF>/STARTING FORMAT ON DRIVE /
4576	022300	051101	044524	043516	
4577	022306	043040	051117	040515	
4578	022314	020124	047117	042040	
4579	022322	044522	042526	000040	
4580	022330	005015	051412	040524	MSCHK: .ASCIZ <CR><LF><LF>/STARTING CHECK ON DRIVE /

4581	022336	052122	047111	020107	
4582	022344	044103	041505	020113	
4583	022352	047117	042040	044522	
4584	022360	042526	000040		
4585	022364	005015	043012	051117	MFCMPT: .ASCIIZ <CR><LF><LF>/FORMAT COMPLETE, /
4586	022372	040515	020124	047503	
4587	022400	050115	042514	042524	
4588	022406	020054	000		
4589	022411	015	005012	044103	MCCMPT: .ASCIIZ <CR><LF><LF>/CHECK COMPLETE, /
4590	022416	041505	020113	047503	
4591	022424	050115	042514	042524	
4592	022432	020054	000		
4593	022435	124	052117	046101	NUMERR: .ASCIIZ /TOTAL ERRORS DETECTED: /
4594	022442	042440	051122	051117	
4595	022450	020123	042504	042524	
4596	022456	052103	042105	020072	
4597	022464	000			
4598	022465	120	042522	042523	ADDRIS: .ASCIIZ /PRESENT ADDRESS IS: /
4599	022472	052116	040440	042104	
4600	022500	042522	051523	044440	
4601	022506	035123	000040		
4602					
4603					*****
4604					
4605					.SBTTL ERROR MESSAGES
4606					
4607					*****
4608					
4609	022512	044122	030461	044440	EM1: .ASCIIZ /RH11 INTERRUPT OCCURRED (RPAS=0)/
4610	022520	052116	051105	052522	
4611	022526	052120	047440	041503	
4612	022534	051125	042522	020104	
4613	022542	051050	040520	036523	
4614	022550	024460	000		
4615	022553	125	042516	050130	EM2: .ASCIIZ /UNEXPECTED ATTENTION OCCURRED/
4616	022560	041505	042524	020104	
4617	022566	052101	042524	052116	
4618	022574	047511	020116	041517	
4619	022602	052503	051122	042105	
4620	022610	000			
4621	022611	115	051501	041123	EM3: .ASCIIZ /MASSBUS PARITY ERROR (MCPE=1)/
4622	022616	051525	050040	051101	
4623	022624	052111	020131	051105	
4624	022632	047522	020122	046450	
4625	022640	050103	036505	024461	
4626	022646	000			
4627	022647	115	051501	041123	EM4: .ASCIIZ /MASSBUS PARITY ERROR (PAR=1)/
4628	022654	051525	050040	051101	
4629	022662	052111	020131	051105	
4630	022670	047522	020122	050050	
4631	022676	051101	030475	000051	
4632	022704	042101	051104	051505	EM5: .ASCIIZ /ADDRESS PLUG CHANGE BIT SET/
4633	022712	020123	046120	043525	
4634	022720	041440	040510	043516	
4635	022726	020105	044502	020124	
4636	022734	042523	000124		

4637	022740	044122	030461	042040	EM6:	.ASCIZ /RH11 DIDN'T RESPOND TO ADDRESSING/
4638	022746	042111	023516	020124		
4639	022754	042522	050123	047117		
4640	022762	020104	047524	040440		
4641	022770	042104	042522	051523		
4642	022776	047111	000107			
4643	023002	051104	053111	020105	EM10:	.ASCIZ /DRIVE OFFLINE/
4644	023010	043117	046106	047111		
4645	023016	000105				
4646	023020	042520	051522	051511	EM11:	.ASCIZ /PERSISTENT DRIVE UNSAFE ERROR/
4647	023026	042524	052116	042040		
4648	023034	044522	042526	052440		
4649	023042	051516	043101	020105		
4650	023050	051105	047522	000122		
4651	023056	047125	047503	051122	EM12:	.ASCIZ /UNCORRECTABLE MASSBUS PARITY ERROR/
4652	023064	041505	040524	046102		
4653	023072	020105	040515	051523		
4654	023100	052502	020123	040520		
4655	023106	044522	054524	042440		
4656	023114	051122	051117	000		
4657	023121	123	043117	053524	EM13:	.ASCIZ /SOFTWARE TIMEOUT/
4658	023126	051101	020105	044524		
4659	023134	042515	052517	000124		
4660	023142	051104	053111	020105	EM14:	.ASCIZ /DRIVE UNSAFE ERROR/
4661	023150	047125	040523	042506		
4662	023156	042440	051122	051117		
4663	023164	000				
4664	023165	103	047117	051124	EM15:	.ASCIZ @CONTROLLER/DRIVE ERROR DURING WRITE@
4665	023172	046117	042514	027522		
4666	023200	051104	053111	020105		
4667	023206	051105	047522	020122		
4668	023214	052504	044522	043516		
4669	023222	053440	044522	042524		
4670	023230	000				
4671	023231	103	047117	051124	EM16:	.ASCIZ @CONTROLLER/DRIVE ERROR DURING WRITE CHECK@
4672	023236	046117	042514	027522		
4673	023244	051104	053111	020105		
4674	023252	051105	047522	020122		
4675	023260	052504	044522	043516		
4676	023266	053440	044522	042524		
4677	023274	041440	042510	045503		
4678	023302	000				
4679	023303	122	052105	044522	EM17:	.ASCIZ @RETRIES NOT SUCCESSFUL - SECTOR NOT ACCEPTABLE@
4680	023310	051505	047040	052117		
4681	023316	051440	041525	051505		
4682	023324	043123	046125	026440		
4683	023332	051440	041505	047524		
4684	023340	020122	047516	020124		
4685	023346	041501	042503	052120		
4686	023354	041101	042514	000		
4687	023361	104	052101	020101	EM20:	.ASCIZ @DATA ERROR DURING WRITE CHECK@
4688	023366	051105	047522	020122		
4689	023374	052504	044522	043516		
4690	023402	053440	044522	042524		
4691	023410	041440	042510	045503		
4692	023416	000				

4693	023417	103	047117	051124	EM21:	.ASCIIZ	@CONTROLLER/DRIVE ERROR VERIFYING HEADERS@
4694	023424	046117	042514	027522			
4695	023432	051104	053111	020105			
4696	023440	051105	047522	020122			
4697	023446	042526	044522	054506			
4698	023454	047111	020107	042510			
4699	023462	042101	051105	000123			
4700	023470	042510	042101	051105	EM22:	.ASCIIZ	@HEADER COMPARE ERROR VERIFYING HEADERS@
4701	023476	041440	046517	040520			
4702	023504	042522	042440	051122			
4703	023512	051117	053040	051105			
4704	023520	043111	044531	043516			
4705	023526	044040	040505	042504			
4706	023534	051522	000				
4707	023537	103	046131	047111	EM23:	.ASCIIZ	@CYLINDER FIELD IN HEADER IS NOT CORRECT@
4708	023544	042504	020122	044506			
4709	023552	046105	020104	047111			
4710	023560	044040	040505	042504			
4711	023566	020122	051511	047040			
4712	023574	052117	041440	051117			
4713	023602	042522	052103	000			
4714	023607	127	044522	042524	EM24:	.ASCIIZ	@WRITE CHECK ERROR@
4715	023614	041440	042510	045503			
4716	023622	042440	051122	051117			
4717	023630	000					
4718	023631	110	051101	053504	EM25:	.ASCIIZ	@HARDWARE ERROR DURING WRITE CHECK@
4719	023636	051101	020105	051105			
4720	023644	047522	020122	052504			
4721	023652	044522	043516	053440			
4722	023660	044522	042524	041440			
4723	023666	042510	045503	000			
4724		023674					
4725						.EVEN	

::*****

4728	023674	050122	051501	000	DH1:	.ASCIIZ	/RPAS/
4729	023701	104	044522	042526	DH2:	.ASCIIZ	/DRIVE RPDS1 RPER1 RPER2 RPER3 RPAS/
4730	023706	020040	051040	042120			
4731	023714	030523	020040	051040			
4732	023722	042520	030522	020040			
4733	023730	051040	042520	031122			
4734	023736	020040	051040	042520			
4735	023744	031522	020040	051040			
4736	023752	040520	000123				
4737	023756	051104	053111	020105	DH3:	.ASCIIZ	/DRIVE REG ADR DATA/
4738	023764	020040	042522	020107			
4739	023772	042101	020122	042040			
4740	024000	052101	000101				
4741	024004	051104	053111	020105	DH4:	.ASCIIZ	/DRIVE REG ADR GOOD BAD/
4742	024012	020040	042522	020107			
4743	024020	042101	020122	020040			
4744	024026	043440	047517	020104			
4745	024034	020040	041040	042101			
4746	024042	000					
4747	024043	122	020110	042101	DH6:	.ASCIIZ	/RH ADDRESS/
4748	024050	051104	051505	000123			

4749	024056	051104	053111	020105	DH10:	.ASCIIZ	/DRIVE	ERR PC	RPCS1	RPCS2	RPDS1	RPER1	RPER2	RPER3/
4750	024064	020040	051105	020122										
4751	024072	041520	020040	050122										
4752	024100	051503	020061	020040										
4753	024106	050122	051503	020062										
4754	024114	020040	050122	051504										
4755	024122	020061	020040	050122										
4756	024130	051105	020061	020040										
4757	024136	050122	051105	020062										
4758	024144	020040	050122	051105										
4759	024152	000063												
4760	024154	050122	041505	020061	DH10A:	.ASCIIZ	/RPEC1	RPEC2	RPWC	RPBA	RPDA	RPAS	RPLA/	
4761	024162	020040	050122	041505										
4762	024170	020062	020040	050122										
4763	024176	041527	020040	020040										
4764	024204	050122	040502	020040										
4765	024212	020040	050122	040504										
4766	024220	020040	020040	050122										
4767	024226	051501	020040	020040										
4768	024234	050122	040514	000										
4769	024241	122	042120	020102	DH10B:	.ASCIIZ	/RPDB	RPMR	RPDT	RPSN	RPOF	RPCA	RPCC/	
4770	024246	020040	051040	046520										
4771	024254	020122	020040	051040										
4772	024262	042120	020124	020040										
4773	024270	051040	051520	020116										
4774	024276	020040	051040	047520										
4775	024304	020106	020040	051040										
4776	024312	041520	020101	020040										
4777	024320	051040	041520	000103										
4778	024326	051104	053111	020105	DH17:	.ASCIIZ	/DRIVE	ERR PC	CYLINDER	TRACK	SECTOR/			
4779	024334	020040	051105	020122										
4780	024342	041520	041440	046131										
4781	024350	047111	042504	020122										
4782	024356	051124	041501	020113										
4783	024364	020040	042523	052103										
4784	024372	051117	000											
4785	024375	104	044522	042526	DH20:	.ASCIIZ	/DRIVE	ERR PC	CYLINDER	TRACK	SECTOR/			
4786	024402	020040	042440	051122										
4787	024410	050040	020103	054503										
4788	024416	044514	042116	051105										
4789	024424	052040	040522	045503										

4805	024556	020040	020040	050122
4806	024564	040504	020040	020040
4807	024572	050122	051501	020040
4808	024600	020040	050122	040514
4809	024606	020040	020040	050122
4810	024614	041104	020040	020040
4811	024622	050122	051115	020040
4812	024630	020040	050122	052104
4813	024636	000		
4814	024637	122	051520	020116
4815	024644	020040	051040	047520
4816	024652	020106	020040	051040
4817	024660	041520	020101	020040
4818	024666	051040	041520	000103
4819	024674	020040	020040	020040
4820	024702	020040	020040	020040
4821	024710	020040	020040	054105
4822	024716	052120	042047	005015
4823	024724	051104	053111	020105
4824	024732	020040	051105	020122
4825	024740	041520	020040	054503
4826	024746	047114	051104	020040
4827	024754	041501	052524	046101
4828	024762	044040	040505	042504
4829	024770	000122		
4830	024772	020040	020040	020040
4831	025000	020040	020040	020040
4832	025006	020040	020040	020040
4833	025014	020040	020040	020040
4834	025022	020040	020040	020040
4835	025030	020040	020040	020040
4836	025036	020040	020040	042515
4837	025044	047515	054522	020040
4838	025052	044504	045523	005015
4839	025060	051104	053111	020105
4840	025066	020040	051105	020122
4841	025074	041520	020040	054503
4842	025102	047114	051104	020040
4843	025110	051124	041501	020113
4844	025116	020040	042523	052103
4845	025124	051117	020040	040504
4846	025132	040524	020040	020040
4847	025140	040504	040524	000

DH20C: .ASCIZ /RPSN RPOF RPCA RPCC/

DH23: .ASCII / EXPT'D/<CR><LF>

```
.ASCIZ /DRIVE ERR PC CYLNDR ACTUAL HEADER/
```

[illegible]

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

4850	025146	001276		
4851	025150	001274	012326	012330
4852	025156	012332	012334	001276
4853	025164	001274	017516	017520
4854	025172	001274	017742	017740
4855	025200	017520		
4856	025202	001172		
4857	025204	001214	001116	001306
4858	025212	001316	001320	001322
4859	025220	001346	001350	
4860	025224	001352	001354	001310

```

DT1:  .WORD  ATTN
DT2:  .WORD  DDRIIVE,RPERRS,RPERRS+2,RPERRS+4,RPERRS+6,ATTN

DT3:  .WORD  DDRIIVE,RD.ADR,RD.WRD
DT4:  .WORD  DDRIIVE,WRT.AD,WRT.WD,RD.WRD

DT6:  .WORD  $RPADR
DT10: .WORD  DRIVE,$ERRPC,RP.REG,RP.REG+10,RP.REG+12,RP.REG+14,RP.REG+40,RP.REG+42

      .WORD  RP.REG+44,RP.REG+46,RP.REG+2,RP.REG+4,RP.REG+6,RP.REG+16,RP.REG+20

```


4861	025232	001312	001314	001324		
4862	025240	001326				
4863	025242	001330	001332	001334	.WORD	RP.REG+22,RP.REG+24,RP.REG+26,RP.REG+30,RP.REG+32,RP.REG+34,RP.REG+36
4864	025250	001336	001340	001342		
4865	025256	001344				
4866	025260	001214	001116	001270	DT17:	.WORD DRIVE,\$ERRPC,DS.CYL,DS.TRK,SAVSEC
4867	025266	001272	001252			
4868	025272	001214	001116	001270	DT20:	.WORD DRIVE,\$ERRPC,DS.CYL,DS.TRK,SAVSEC
4869	025300	001272	001252			
4870	025304	001306	001316	001320	.WORD	RP.REG,RP.REG+10,RP.REG+12,RP.REG+14,RP.REG+40,RP.REG+42,RP.REG+44,RP.RE
4871	025312	001322	001346	001350		
4872	025320	001352	001354			
4873	025324	001310	001312	001314	.WORD	RP.REG+2,RP.REG+4,RP.REG+6,RP.REG+16,RP.REG+20,RP.REG+22,RP.REG+24,RP.RE
4874	025332	001324	001326	001330		
4875	025340	001332	001334			
4876	025344	001336	001340	001342	.WORD	RP.REG+30,RP.REG+32,RP.REG+34,RP.REG+36
4877	025352	001344				
4878	025354	001214	001116	025574	DT23:	.WORD DRIVE,\$ERRPC,BUFP,RBUF,RBUF+2,RBUF+4,RBUF+6
4879	025362	025564	025566	025570		
4880	025370	025572				
4881	025372	001214	001116	001270	DT24:	.WORD DRIVE,\$ERRPC,DS.CYL,DS.TRK,SAVSEC,\$GDDAT,RP.REG+22
4882	025400	001272	001252	001124		
4883	025406	001330				
4884	025410	001306	001316	001320	.WORD	RP.REG,RP.REG+10,RP.REG+12,RP.REG+14,RP.REG+40,RP.REG+42,RP.REG+44,RP.RE
4885	025416	001322	001346	001350		
4886	025424	001352	001354			
4887	025430	001310	001312	001314	.WORD	RP.REG+2,RP.REG+4,RP.REG+6,RP.REG+16,RP.REG+20,RP.REG+22,RP.REG+24,RP.RE
4888	025436	001324	001326	001330		
4889	025444	001332	001334			
4890	025450	001336	001340	001342	.WORD	RP.REG+30,RP.REG+32,RP.REG+34,RP.REG+36
4891	025456	001344				
4892						
4893	025460	000001			DF1:	.WORD 1
4894	025462	001	000			.BYTE 1,0
4895	025464	000001			DF2:	.WORD 1
4896	025466	006	000			.BYTE 6,0
4897	025470	000001			DF3:	.WORD 1
4898	025472	003	000			.BYTE 3,0
4899	025474	000001			DF4:	.WORD 1
4900	025476	004	000			.BYTE 4,0
4901	025500	000003			DF10:	.WORD 3
4902	025502	010	000			.BYTE 8,0
4903	025504	024154				.WORD DH10A
4904	025506	007	000			.BYTE 7,0
4905	025510	024241				.WORD DH10B
4906	025512	007	000			.BYTE 7,0
4907	025514	000001			DF17:	.WORD 1
4908	025516	005	034			.BYTE 5,34
4909	025520	000004			DF20:	.WORD 4
4910	025522	005	034			.BYTE 5,34
4911	025524	024444				.WORD DH20A
4912	025526	010	000			.BYTE 8,0
4913	025530	024542				.WORD DH20B
4914	025532	010	000			.BYTE 8,0
4915	025534	024637				.WORD DH20C
4916	025536	004	000			.BYTE 4,0

4917 025540 000001
 4918 025542 007 000
 4919 025544 000004
 4920 025546 007 034
 4921 025550 024444
 4922 025552 010 000
 4923 025554 024542
 4924 025556 010 000
 4925 025560 024637
 4926 025562 004 000

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

4934 025564 000000 000000 000000
 4935 025572 000000

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

4936
 4937 025574

BUFP: ;FORMAT AND CHECK BUFFER STARTS HERE

4938
 4939 025574 005015 055012 026532
 4940 025602 055103 045122 026502

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

4941 025610 006504 012
 4942 025613 122 030120 027464

.ASCIIZ @RP04/5/6 FORMATTER PROGRAM@<CR><LF><LF>

4943 025620 027465 020066 047506
 4944 025626 046522 052101 042524
 4945 025634 020122 051120 043517
 4946 025642 040522 006515 005012

4947 025650 000
 4948 025651 015 052012 020117

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

4949 025656 043047 051117 040515
 4950 025664 023524 047440 020122

4951 025672 041447 042510 045503
 4952 025700 020047 051104 053111

4953 025706 020105 026060 051040
 4954 025714 050105 040514 042503

4955 025722 052040 042510 023440
 4956 025730 054130 050104 006447

4957 025736 012
 4958 025737 120 041501 020113

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

4959 025744 047117 042040 044522
 4960 025752 042526 030040 053440

4961 025760 052111 020110 047101
 4962 025766 052117 042510 020122

4963 025774 040520 045503 020054
 4964 026002 046103 040505 020122

4965 026010 042515 047515 054522
 4966 026016 046040 041517 052101

4967 026024 047511 020116 030064
 4968 026032 006454 012

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

4969 026035 101 042116 051040
 4970 026042 051505 040524 052122

4971 026050 052040 042510 050040
 4972 026056 047522 051107 046501

4973 026064 005015 000

4974

4975

4976

4977

4978

4979

4980

4981

4982

4983

4984

4985

4986

4987 026070 177700

4988 026072 000776

4989 026074 000000

4990 026076 005737 001302

4991 026102 001460

4992 026104 005037 001302

4993 026110 012700 001172

4994 026114 104401 026316

4995 026120 012046

4996 026122 104402

4997 026124 104401 021305

4998 026130 104411

4999 026132 012601

5000 026134 013737 026070 026074

5001 026142 004537 026340

5002 026146 026166

5003 026150 026244

5004 026152 026110

5005 026154 026162

5006 026156 026110

5007 026160 026240

5008 026162 010260 177776

5009 026166 104401 026327

5010 026172 012700 001174

5011 026176 012046

5012 026200 104402

5013 026202 104401 021305

5014 026206 104411

5015 026210 012601

5016 026212 013737 026072 026074

5017 026220 004537 026340

5018 026224 026244

5019 026226 026244

5020 026230 026166

5021 026232 026240

5022 026234 026166

5023 026236 026240

5024 026240 010260 177776

5025 026244 013701 000004

5026 026250 012737 026304 000004

5027 026256 005777 152710

5028 026262 010137 000004

.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

BEQ 7\$

CLR CHGADR

1\$: MOV #SRPADR,R0

TYPE ,MRPCS1

MOV (R0)+,-(SP)

TYPOC

TYPE ,LINSR

RDLIN

MOV (SP)+,R1

MOV HIAD,LIMIT

JSR R5,CK.NUM

3\$

7\$

1\$

2\$

1\$

4\$

2\$: MOV R2,-2(R0)

3\$: TYPE ,MRHVEC

MOV #SRPVEC,R0

MOV (R0)+,-(SP)

TYPOC

TYPE ,LINSR

RDLIN

MOV (SP)+,R1

MOV HIVEC,LIMIT

JSR R5,CK.NUM

7\$

7\$

3\$

4\$

3\$

4\$

4\$: MOV R2,-2(R0)

7\$: MOV ERRVEC,R1

MOV #8\$,ERRVEC

TST @SRPADR

MOV R1,ERRVEC

:INPUT FROM TTY REQUESTED?

:NO--BRANCH

:YES--CLEAR THE REQUEST FLAG

:FIRST ADDRESS

:RPCS1=

:PRESENT RPCS1 ADDRESS

:TYPE IT

:2 SPACES

:GET THE ENTRY

:ADDRESS OF ASCII TEXT

:THIS IS THE ADDRSS HIGH LIMIT

:CHECK THE NUMBER

:CARRIAGE RETURN ONLY ENTERED

:PERIOD ONLY ENTERED

:ILLEGAL INPUT

:TERMINATED WITH A CARRIAGE RETURN

:TERMINATED WITH A '...'

:TERMINATED WITH A '...'

:SAVE NEW RPCS1

:RHVEC=

:FIRST VECTOR

:PRESENT RH11 VECTOR ADDRESS ON THE STACK

:TYPE IT

:2 SPACES

:READ THE ENTRY

:ASCII TEXT ADDRESS

:VECTOR LIMIT

:CHECK THE NUMBER

:CARRIAGE RETURN ONLY ENTERED

:PERIOD ONLY ENTERED

:ILLEGAL INPUT

:TERMINATED WITH A CARRIAGE RETURN

:TERMINATED WITH A '...'

:TERMINATED WITH A '...'

:SAVE INPUT

:SAVE THE ERROR VECTOR

:SETUP FOR TRAP

:CHECK FOR RH11

:RESTORE ERROR VECTOR


```

5029 026266 012700 001172      MOV    #SRPADR,R0      ;FIRST ADDRESS OF NEW PARAMETERS
5030 026272 012701 012474      MOV    #RPADR,R1      ;FIRST ADDRESS OF WHERE TO PUT THEM
5031 026276 012021              MOV    (R0)+,(R1)+      ;BUS ADDRESS
5032 026300 012021              MOV    (R0)+,(R1)+      ;VECTOR ADDRESS
5033 026302 000207              RTS     PC              ;RETURN
5034 026304 010137 000004      8$:    MOV    R1,ERRVEC      ;RESTORE ERROR VECTOR
5035 026310 022626              CMP    (SP)+,(SP)+      ;CLEAN OFF THE STACK
5036 026312 104006              ERROR  6              ;REPORT THE ERROR
5037 026314 000675              BR     1$              ;ASK FOR BUS ADDRESS
5038
5039 026316 050122 051503 020061 MRPCS1: .ASCIZ @RPCS1 = @
5040 026324 020075      000
5041 026327      122 053110 041505 MRHVEC: .ASCIZ @RHVEC = @
5042 026334 036440 000040
5043
5044      .SBTTL CK.NUM - CHECK NUMBER (OCTAL)
5045      ;THIS ROUTINE CHECKS AN ASCIZ STRING FOR LEGAL CHARACTERS
5046      ;AND FORMS AN OCTAL NUMBER IN R2
5047      ;CALL:
5048      :      MOV    #ADR,R1      ;ADDRESS OF ASCIZ STRING
5049      :      MOV    #NUM,R2      ;MAX SIZE OF INPUT NUMBER
5050      :      JSR    R5,CK.NUM      ;GO FORM THE NUMBER
5051      :      RETURN ADR1          ;'CR' ONLY ENTERED -- R2 = 0
5052      :      RETURN ADR2          ;'PERIOD' ONLY ENTERED -- R2 = 0
5053      :      RETURN ADR3          ;ILLEGAL CHARACTER IN THE INPUT STRING
5054      :      RETURN ADR4          ;'CR' ENTERED -- R2 = NUMBER
5055      :      RETURN ADR5          ;'COMMA' -- R2 = NUMBER
5056      :      RETURN ADR6          ;'PERIOD' -- R2 = NUMBER
5057
5058 026340 010446      CK.NUM: MOV    R4,-(SP)      ;SAVE R4
5059 026342 010346      MOV    R3,-(SP)      ;SAVE R3
5060 026344 010246      MOV    R2,-(SP)      ;SAVE R2
5061 026346 005004      CLR     R4              ;RETURN POINTER
5062 026350 005003      CLR     R3              ;START NUMBER AT ZERO
5063 026352 005002      CLR     R2              ;STORE RESULT
5064 026354 004537 006654      JSR    R5,CK.CHR      ;CHECK ONE CHARACTER
5065 026360 026460      6$              ;ILLEGAL CHARACTER
5066 026362 026464      8$              ;CARRIAGE RETURN
5067 026364 026460      6$              ;...
5068 026366 026462      7$              ;...
5069 026370 026374      1$              ;DIGIT 0-7
5070 026372 026460      6$              ;DIGIT 8-9
5071 026374 062705 000004      1$:    ADD    #4,R5      ;INCREMENT RETURN PAST 'CR' AND 'PERIOD' ONLY RETURNS
5072 026400 006303      2$:    ASL     R3              ;FOR THE OCTAL NUMBER IN R3
5073 026402 103426      BCS     6$              ;DON'T LET IT GET TO BIG
5074 026404 006303      ASL     R3
5075 026406 103424      BCS     6$
5076 026410 006303      ASL     R3
5077 026412 103422      BCS     6$
5078 026414 060203      ADD     R2,R3
5079 026416 004537 006654      JSR    R5,CK.CHR      ;CHECK ONE CHARACTER
5080 026422 026464      8$              ;ILLEGAL CHARACTER
5081 026424 026446      5$              ;CARRIAGE RETURN
5082 026426 026444      4$              ;...
5083 026430 026436      3$              ;...
5084 026432 026400      2$              ;DIGIT 0-7

```


5085	026434	026464		8\$		:DIGIT 8-9
5086	026436	105711		3\$: TSTB	(R1)	:DOES A 'CR' FOLLOW THE 'PERIOD'
5087	026440	001011		BNE	8\$	:BR IF NOT
5088	026442	005724		TST	(R4)+	:INCREMENT THE RETURN
5089	026444	005724		4\$: TST	(R4)+	:INCREMENT THE RETURN INDEX
5090	026446	005724		5\$: TST	(R4)+	:INCREMENT THE RETURN INDEX
5091	026450	023703	026074	CMP	LIMIT,R3	:INPUT VALUE TOO LARGE ?
5092	026454	101003		BHI	8\$	:BR IF IT IS
5093	026456	000401		BR	7\$	:BR IF NOT
5094	026460	005725		6\$: TST	(R5)+	:INCREMENT THE RETURN ADDRESS
5095	026462	005725		7\$: TST	(R5)+	:INCREMENT THE RETURN ADDRESS
5096	026464	060405		8\$: ADD	R4,R5	:SETUP FOR PROPER RETURN
5097	026466	010302		MOV	R3,R2	:LOAD ENTERED VALUE
5098	026470	005726		TST	(SP)+	:CLEAN OFF THE STACK
5099	026472	012603		MOV	(SP)+,R3	:RESTORE R3
5100	026474	012604		MOV	(SP)+,R4	:RESTORE R4
5101	026476	011505		MOV	(R5),R5	:GET RETURN ADDRESS
5102	026500	000205		RTS	R5	:RETURN
5103						
5104						
5105						
5106	000001			.END		

[illegible]

[illegible]

F5	=	000040	714#																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																									</
----	---	--------	------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	----

MPATD	022260	1555	4573#				
MPE	= 000400	688#					
MRD	= 000020	761#					
MRHVEC	026327	5009	5041#				
MRPCS1	026316	4994	5039#				
MSCHK	022330	1571	4580#				
MSE	= 000020	820#					
MSEC20	021757	1465	4537#				
MSEC22	021700	1459	4529#				
MSELD	021507	4507#					
MSELP	022160	1545	4560#				
MSFOU	022273	1569	4575#				
MSIZE	021627	1448	4522#				
MSTCK	= 000010	760#					
MUNIT	021257	1482	4476#				
MUSDR	021467	1510	4504#				
MWC	001260	994#	1461*	1467*	1641	1730	1905
MWR	= 000040	762#					
MXDLTA	012504	3208#	3757				
MXF	= 001000	689#					
MXLACT	012502	3205#	3744				
MXWWDW	012510	3214#	3562				
MO	003232	1475	1479#	1496	1511	1515	1517 1848
M1	002760	1427#	1438				
M1A	003050	1440	1448#	1458			
M1B	003176	1464	1470#	2023			
M2	003520	1519	1529	1533#	1541		
M4	003560	1536	1539	1545#	1561		
M5	003642	1566#	1850				
NBA	= 100000	793#					
NED	= 010000	692#					
NEM	= 004000	691#					
NHS	= 002000	826#	844#				
NOTPRS	021174	1396	4465#				
NOTRP	021153	1394	4462#				
NOTSAF	021211	1404	4468#				
NOUSE	021065	1400	4451#				
NUMERR	022435	1824	4593#				
OCYL	= 100000	878#	888#				
OENTER	006344	1479	2019#	2028			
OFFSET	= 000115	908#					
OFREV	= 000200	856#					
OF100	= 000004	852#					
OF200	= 000010	853#					
OF25	= 000001	850#					
OF400	= 000020	854#					
OF50	= 000002	851#					
OF800	= 000040	855#					
OPE	= 020000	886#					
OPI	= 020000	751#	1696				
OPT	013604	3432	3467#	3801	3961	4002	
OR	= 000200	687#					
PAR	= 000010	741#					
PARENT	006464	1534	2049#				
PAR1	001430	1055	1063#				
PAR2	001443	1056	1065#				

SW12 =	010000	589#																			
SW13 =	020000	588#																			
SW14 =	040000	587#																			
SW15 =	100000	586#																			
SW2 =	000004	609#																			
SW3 =	000010	608#																			
SW4 =	000020	607#																			
SW5 =	000040	606#																			
SW6 =	000100	605#																			
SW7 =	000200	604#																			
SW8 =	000400	603#																			
SW9 =	001000	602#																			
SYSTAT	021240	1380	4473#																		
T	021277	2035	4480#																		
TABLE	001376	1055#	1527*	1528*	2050	2076															
TAP =	040000	792#																			
TBITVE=	000014	644#																			
TD	015452	3775	3784#																		
TDF =	000040	821#	839#																		
TIMER	012440	3164#	3377*	3511*	3654*	3700*	3787*	3880*	3924*	3975*	3983*	4016	4018*	4076*							
		4103*	4109*	4115*																	
TITLE	025574	1342	4939#																		
TKVEC =	000060	651#	2571*	2572*																	
TPVEC =	000064	652#																			
TRAPVE=	000034	650#	1312*	1313*	1333*																
TRE =	040000	669#																			
TRKCNT	001236	985#	1911*	1920	1922*	1926*															
TRKTST	005720	1761	1918#																		
TRK1 =	004000	808#																			
TRK10 =	040000	811#																			
TRK2 =	010000	809#																			
TRK20 =	100000	812#																			
TRK4 =	020000	810#																			
TRNSWT	012406	2020	3096#	3533*	3678*	3688	3705*	3714*	3789	3790*	4054	4069	4082*								
TRTVEC=	000014	645#																			
TSTPGM	013310	1293*	1296*	1299*	1340*	1343	3343	3392#													
TTRKS	001232	983#	1584*	1585*	1586*	15															

[illegible]

\$DTBL	010374	2508	2543#																
\$ENDAD	005422	528	1839#																
\$ENDCT	005406	1834#																	
\$EOP	005326	1805	1818#	1898															
\$EOPCT	005400	1831#	1835																
\$ERFLG	001103	934#	2235*	2262															
\$ERMAX	001115	940#	1315*																
\$ERROR	007064	1310	2224#																
\$ERRPC	001116	941#	2242*	2243*	2244	2262	4857	4866	4868	4878	4881								
\$ERRTB	001630	1138#	2276																
\$ERTTL	001112	938#	1480*	1706*	1825	2241*	2262												
\$ESCAP	001160	961#	1314*	1654*	1667*	1686*	1703*	1709*	1755*	1786*	1796*	1892*	2257	2259					
		2262	3821	3872	3876	4154	4214	4423	4432	4433*	4435*								
\$FILLC	001156	959#	2371	2402															
\$FILLS	001155	958#	2402																
\$GDADR	001120	942#																	
\$GDDAT	001124	944#	2234*	4881															
\$GET42	005412	1836#																	
\$GTSWR	011010	2655#	2994																
\$HD =	000000	499																	
\$ICNT	001104	935#																	
\$INTAG	001135	949#	2652*	2671	2691	2804													
\$ITEMB	001114	939#	2244*	2262	2271														
\$LF	001170	965#	2262	2402	2787	2797													
\$LKCSB	001200	972#	1995*																
\$LKCSR	001176	971#	1990	1996*															
\$LKS	001206	974#	2000	2005*															
\$LLVEC	001210	975#	2001																
\$LPADR	001106	936#																	
\$LPERR	001110	937#	1646*	1673*	1778*	1891	2256												
\$LPVEC	001202	973#	1991																
\$MAIL =	***** U	1332	1355	2250	2355														
\$MNEW	011645	2658	2802#																
\$MSWR	011634	2655	2800#																
\$NULL	001154	957#	2373	2402															
\$OCNT	010164	2435*	2464*	2477#															
\$OMODE	010166	2430*	2434*	2439	2442*	2453*	2479#												
\$PASS	001100	932#	1828*	1829*	1846														
\$QUES	001166	963#	1437	1457	1495	1560	2262	2402	2709	2780	2797								
\$RDCHR	011262	2722#	2997																
\$RDDEC=	***** U	2999																	
\$RDLIN	011352	2745#	2998																
\$RDOCT=	***** U	2999																	
\$RDSZ =	000007	2738#																	
\$RESRE	012204	2944#	3000																
\$RPADR	001172	969#	1364	4856	4993	5027	5029												
\$RPVEC	001174	970#	1365	5010															
\$RTNAD	005434	1845#																	
\$R2A =	***** U	3001																	
\$SAVRE	012146	2928#	2999																
\$SB2D	011716	2032	2038	2840#															
\$SETUP=	000106	1292#	1309	1310	1312	1314	1316	1352	1828	2225	2253	2261	2592	2597					
		2598	2628	2804															
\$STUP =	177777	1292#																	
\$SUPRS	011656	2033	2039	2815#															
\$SVPC =	000200	526#	531																

\$SWR = 123000	489#	499	504	505	506	507	508	961	962	1314	1815	1828	1838
\$TKB 001146	1844	1846	2216	2217	2218	2219	2220	2238	2245	2250	2254	2262	
\$TKCNT 010414	954#	2552	2573	2584	2609	2637	2664						
\$TKINT 010432	2553#	2568*	2598	2615*	2729	2731*							
\$TKQEN= 010431	1351	1481	2568#	2589	2650								
\$TKQIN 010416	2557#	2623	2734										
\$TKQOU 010420	2554#	2569*	2570	2621*	2622*	2623	2625*						
\$TKQSR 010422	2555#	2570*	2732	2733*	2734	2736*							
\$TKS 001144	2556#	2569	2625	2736									
\$TKSRV 010502	953#	2552	2574*	2605*	2607	2613*	2635	2651*	2661	2673*	2693*		
\$TN = 000000	2571	2584#											
\$TNPWR 012062	489#	499											
\$TPB 001152	2868	2869	2889#										
\$TPFLG 001157	956#	2391*	2402										
\$TPS 001150	960#	2349	2402										
\$TRAP 012242	955#	2389	2402										
\$TRAP2 012264	1312	2965#											
\$TRP = 000014	2976#	2987											
	2980#	2989#	2990#	2991#	2992#	2993#	2994	2995#	2996	2997#	2998#	2999#	3000#
	3001#												
\$TRPAD 012276	2970	2987#											
\$TSTNM 001102	933#	2237	2262										
\$TTYIN 011606	2747	2748	2760	2778	2792	2796#							
\$TYPBN= ***** U	2993												
\$TYPDS 010170	2493#	2992											
\$TYPE 007522	2349#	2980	2988										
\$TYPEC 007672	2370	2377	2384	2389#	2390	2695							
\$TYPEX 007740	2395	2397	2400#										
\$TYPOC 007766	2433#	2989											
\$TYPON 010002	2432	2435#	2991										
\$TYPOS 007742	2428#	2990											
\$GET4= 000000	1838#												
\$OFILL 010165	2429*	2433*	2443	2478#									
\$40CAT= ***** U	2247												
. = 026502	514#	518#	526	527#	529#	531#	534#	541#	930#	966	1307	1646	1673
	1778	1846	2262	2402	2547#	2552	2556#	2557	2558#	2796#	2797	2804	2909#
	4322#	4323#	4324#	4325#	4326#	4327#	4328#	4329#	4330	4724#	4849#	4975#	

CKCHR	489#	1485	1548	2168	2182	5064	5079										
CKDIG	489#	2062															
CKNUM	489#	5001	5017														
COMMEN	1#	654#															
ENDCOM	1#	654#															
ENDTYP	1811#	1819															
ERRCAL	3002#	3821	3872	3875	4153	4214											
ERRENT	2209#	2226															
ERROR	548#	1656	1657	1658	1659	1660	1661	1688	1689	1690	1691	1692	1698	1708	1715		
	1722	1725	1788	1789	1790	1791	1792	1795	1797	1801	3822	3873	3877	4155	4215		
	5036																
ESCAPE	1#	654#	1654	1686	1703	1786											
GETPRI	1#	654#															
GETSWR	1#	489#	654#	1352													
LOOP	489#	1646	1673	1778													
MORETA	924#	967															
MULT	1#	654#															
NEWTST	1#	654#															
POP	1#	654#	2534	2949													
PUSH	1#	654#	2493	2929													
REPORT	1#	654#															
RP04.D	1#	3002															
SCOPE	549#																
SETPRI	1#	654#	2725														
SETTRA	2980#	2989	2990	2991	2992	2994	2996	2997	2998	2999	3000						
SETUP	1#	654#	1302														
SKIP	1#	654#															
SLASH	1#	654#															
SPACE	654#																
STARS	1#	524	654#	655	659	701	705	896	900	926	966	1286	1290	1813	1852		
	1856	2212	2334	2405	2483	2551	2628	2643	2714	2738	2807	2831	2850	2913	2959		
	3011	4439	4603	4607	4726	4928	4932										
SWRSU	1#	654#	1316#														
TRMTRP	2980#																
TYPBIN	1#	654#															
TYPDEC	1#	654#	1825	2302													
TYPNAM	1#	654#															
TYPNUM	1#	654#															
TYPOCS	1#	654#	1381	1572													
TYPOCT	1#	654#	2299	2656													
TYPTXT	1#	654#															
\$\$CMRE	924#																
\$\$CMTM	924#																
\$\$ESCA	1#	654#															
\$\$NEWT	1#	654#															
\$\$SET	2980#	2989	2990	2991	2992	2994	2996	2997	2998	2999	3000						
\$\$SKIP	1#	654#															
.EQUAT	1#	489#	544														
.HEADE	1#	489#															
.KT11	1#																
.SETUP	1#	489#	1292														
.SWRHI	1#	489#	500														
.SWRLO	508#	509	510														
.\$ACT1	1#	489#	522														
.\$APT8	1#																
.\$APTH	1#																

.\$APTY	1#		
.\$ASTA	1#		
.\$CATC	1#	489#	512
.\$CMTA	1#	489#	924
.\$DB2D	1#	489#	2848
.\$DB2O	1#		
.\$DIV	1#		
.\$EOP	1#	489#	1811
.\$ERRO	1#	489#	2210
.\$ERRT	1#	489#	
.\$MULT	1#		
.\$POWE	1#		
.\$RAND	1#		
.\$RDDE	1#		
.\$RDOC	1#		
.\$READ	1#	489#	2549
.\$R2AZ	1#		
.\$SAVE	1#	489#	2911
.\$SB2D	1#	489#	2829
.\$SB2O	1#		
.\$SCOP	1#		
.\$SIZE	1#		
.\$SUPR	1#	489#	2805
.\$TRAP	1#	489#	2957
.\$TYPB	1#		
.\$TYPD	1#	489#	2481
.\$TYPE	1#	489#	2332
.\$TYPO	1#	489#	2403
.\$4OCA	1#		
.1170	1#		

. ABS. 026502 000

ERRORS DETECTED: 0

DSKZ:CZRJBD.BIN,DSKZ:CZRJBD.LST/CRF/SOL/NL:TOC:MD:MC:CND/LI:ME=CZRJBD.SML,CZRJBD.011,CZRJBD.P11
 RUN-TIME: 48 65 4 SECONDS
 RUN-TIME RATIO: 252/118=2.1
 CORE USED: 47K (93 PAGES)