RP07

RP07 PERF EXER
CZRJOB0

AH-F965B-MC
FICHE 1 OF 2

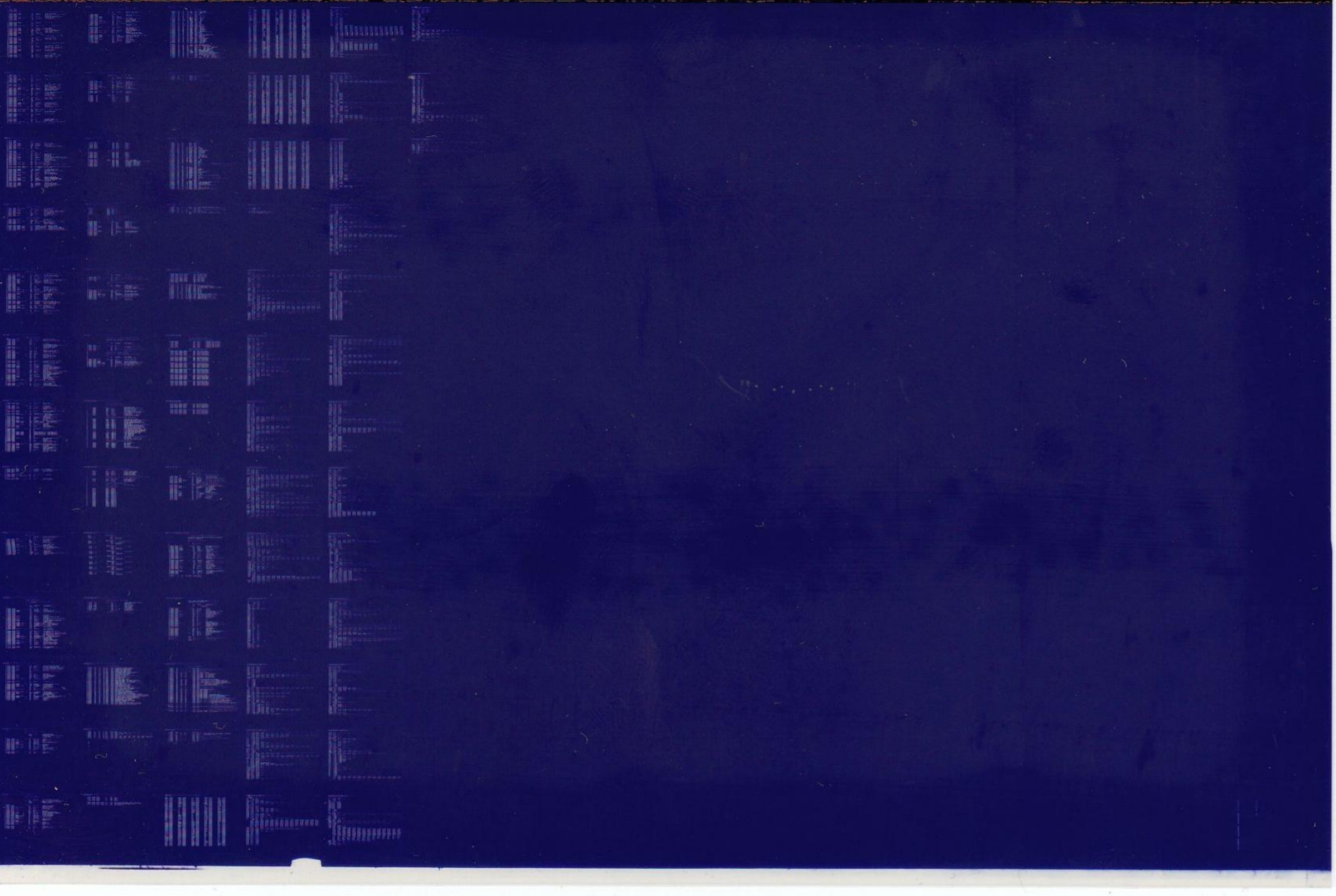
APR 1984
digital
Made In USA

RP07

RP07 PERF EXER
CZRJOB0

AH-F965B-MC
FICHE 2 OF 2

APR 1984
digital
Made In USA



USER DOCUMENTATION

.REM @

IDENTIFICATION

PRODUCT CODE: AC F960B-MC

PRODUCT NAME: CZRJMBO RP07 FRONT-END/ISOLATOR TEST

PRODUCT DATE: DECEMBER 1, 1983

MAINTAINER: CX DIAGNOSTIC ENGINEERING

AUTHOR: MIKE LEAVITT

USER DOCUMENTATION

.REM @

IDENTIFICATION

PRODUCT CODE: AC-F964B-MC

PRODUCT NAME: CZRJOBO RP07 PERFORMANCE EXERCISER

PRODUCT DATE: DECEMBER 1, 1983

MAINTAINER: CX DIAGNOSTIC ENGINEERING

AUTHOR: MIKE LEAVITT

AUTHOR: MIKE LEAVITT

.REM @

CONTENTS

1. ABSTRACT

THE RP07 PERFORMANCE EXERCISER PROGRAM IS DESIGNED TO PERFORM AN INTERACTIVE TEST ON RP DISK DRIVES CONNECTED TO A MASSBUS SUBSYSTEM. THE DRIVES MAY BE CONTROLLED BY AN RH70 CONTROLLER. IN ADDITION TO PERFORMING AN INTERACTIVE TEST OF THE DISK DRIVES ON THE SUBSYSTEM, THE PROGRAM IS INTENDED TO BE USED TO VERIFY THAT THE DRIVES UNDER TEST ARE PERFORMING TO THEIR DATA ERROR RATE AND SEEK ERROR RATE (SEE ERROR RATE SPECIFICATIONS).

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

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

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

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

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

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

1.1 GENERAL DOCUMENT NOTES

A. IN REFERENCE TO ALL NUMBERS IN THIS DOCUMENTION. TO INDICATE THE BASE OF A NUMBER LARGER THAN SEVEN. A PERIOD(.) WILL FOLLOW THE NUMBER TO INDICATE DECIMAL OR NO PERIOD WILL FOLLOW THE NUMBER TO INDICATE OCTAL. IF THE NUMBER OCCURS AT THE END OF A SENTENCE, A DOUBLE PERIOD(. .) INDICATES DECIMAL AND A SINGLE PERIOD(.) INDICATES OCTAL. ALSO, ANY REFERENCES TO TIME ARE ALWAYS IN DECIMAL.

2. REQUIREMENTS

2.1 EQUIPMENT

PDP-11 PROCESSOR
20K MEMORY
KW11-L OR KW11 P CLOCK
PROGRAM LOADING DEVICE
TERMINAL
RH11 OR RH70 CONTROLLER
1 TO 8 DISK DRIVES (RP07'S)

2.2 MEDIA

THE PERFORMANCE EXERCISER PROGRAM REQUIRES FORMATTED DISK PACKS GENERATED BY THE RP07 FORMATTER PROGRAM (ISSFMT). THE PACKS MUST BE FORMATTED IN 32. SECTOR (16 BIT) MODE;THE ALTERNATE (30. SECTOR - 18 BIT) MODE IS NOT SUPPORTTED.

2.3 PRELIMINARY PROGRAMS

RP07 FRONT-END TEST

RP07 FUNCTIONAL TEST

3. OPERATING PROCEDURE

3.1 LOADING THE PROGRAM

THE PROGRAM MAY BE LOADED BY EITHER OF THE FOLLOWING MEDIA:

.PAPER TAPE, USING THE STANDARD PAPER TAPE PROCEDURE
.XXDP MEDIA, USING ANY XXDP DEVICE

3.2 STARTING ADDRESSES

200 - START ADDRESS, ALL SWITCHES CLEAR (SEE SECTION 3.4)

WHEN THE PROGRAM IS STARTED, A DATA PATTERN WILL BE WRITTEN TO ALL ON-LINE DRIVES IN A SEQUENTIAL SEEK MODE. UPON COMPLETION OF THE WRITE, THE PROGRAM GOES INTO A TESTING MODE.

204 - RESTART ADDRESS, THE RESTART ADDRESS PROVIDES THE OPERATOR WITH THE ABILITY TO CHANGE THE DEFAULT RP/RH ADDRESSES (SEE SECTION 4.1.2), ANY PROGRAM PARAMETERS (SEE SECTION 4.1) OR CHANGE DRIVE LIMIT PARAMETERS (SEE SECTION 4.2).

3.3 PROGRAM CONTROL

PROVIDED THE PROGRAM HAS BEEN LOADED AND STARTED VIA THE APT DUMP MODE OR THE DIAGNOSTIC IS RUNNING IN STAND ALONE PROCESSOR/DRIVE OPERATIONS ARE INITIATED AND CONTROLLED BY KEYBOARD COMMANDS AND SWITCH REGISTER SWITCH SETTINGS.

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

3.4 SWITCH OPTIONS

IF THE PROGRAM IS BEING RUN ON A SWITCHLESS PROCESSOR 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 . THE SETTINGS OF THE 'SOFTWARE' SWITCHES ARE CONTROLLED THROUGH A KEYBOARD ROUTINE WHICH IS CALLED BY TYPING A 'CONTROL G'. THE PROGRAM WILL RECOGNIZE THE 'CONTROL G' AT ANY TIME EXCEPT WHEN THE PROGRAM IS IN KEYBOARD ENTRY MODE, OR IS AT A HIGHER PRIORITY PROCESSING AN DRIVE INTERRUPT. THE 'SOFTWARE' SWITCH VALUES ARE ENTERED AS AN OCTAL NUMBER IN RESPONSE TO THE PROMPT FROM THE SWITCH ENTRY ROUTINE:

'SWR = NNNNNN NEW ='

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

ON PROCESSORS WITH HARDWARE SWITCH REGISTERS, THE 'SOFTWARE' SWITCH REGISTER MAY BE USED, IF THE PROGRAM FINDS ALL 1'S IN THE SWITCHES. ALL SWITCH REGISTER REFERENCES WILL BE TO THE 'SOFTWARE' REGISTER AND THE PROCEDURES DESCRIBED ABOVE MUST BE FOLLOWED.

SW<15>=1	HALT ON ERROR
SW<14>	NOT USED
SW<13>=1	INHIBIT ERROR TYPEOUT
SW<12>	NOT USED
SW<11>	NOT USED
SW<10>=1	BELL ON ERROR
SW<09>=1	CHANGE END OF PASS TO 1/4 OF NORMAL
SW<09>	NOT USED
SW<08>=1	INHIBIT END OF PASS MESSAGES
SW<07>=1	DISPLAY ALL DATA COMPARE ERRORS
SW<06>=1	DO NOT ALTER THE CURRENT OPERATION PARAMETERS
SW<05>=1	PARTIAL REGISTER DISPLAY IF ERROR; DO NOT DISPLAY ECC CORRECTION RESULTS
SW<04>=1	INHIBIT MAXIMUM ERROR COUNT CHECK; DO NOT DEASSIGN DRIVES WHEN END OF TEST IS REACHED
SW<03>=1	DISPLAY THE SECTOR IN ERROR (BEFORE RETRY ATTEMPTS) IF 'DCK', 'DTE', OR 'WCF' ERRORS OR AFTER THE 28TH RETRY IF UNCORRECTABLE 'DCK' ERROR IF DATA COMPARE ERRORS & SW<07> SET, DISPLAY REST OF BUFFER
SW<02>=1	INHIBIT SUBSYSTEM STATUS TYPEOUT DURING STARTUP

INHIBIT PERFORMANCE REPORT AFTER SPECIFIED TIME
PROMPT 'WRITE ANYWHERE' QUESTION DURING AUTO TEST MODE
SW<01>=1 INHIBIT DATA COMPARE AFTER READ COMMAND, W/O ERROR
SW<00>=1 READ ONLY MODE

3.5 PASS/TEST TERMINATION:

A PASS IN RANDOM 'T' COMMAND MODE OR SEQUENTIAL 'T' COMMAND MODE IS DETERMINED BY EITHER BITS READ OR SEEKS PERFORMED. THE NUMBER OF BITS OR SEEKS REQUIRED FOR A PASS IS DERIVED FROM EITHER THE SOFT ERROR RATE SPECIFICATION OR THE SEEK ERROR RATE SPECIFICATION.

THE SOFT ERROR SPECIFICATION FOR THE RP DRIVE IS NO MORE THAN 1 SOFT ERROR (NON-DISK RELATED) IN 1 X 10↑10 BITS READ. (SEE SECTION 3.5.1 FOR THE 90% CONFIDENCE LEVEL)

THE SEEK ERROR SPECIFICATION FOR THE RP DRIVE IS NO MORE THAN 1 SEEK ERROR IN 1 X 10↑6 SEEKS. (SEE SECTION 3.5.1 FOR THE 90% CONFIDENCE LEVEL)

A PASS IN 'W' OR 'R' COMMAND MODE IS DETERMINED BY THE MAXIMUM DISK ADDRESS LIMITS SETUP BY THE OPERATOR.

3.5.1 PASS TERMINATION

END OF PASS FOR A SINGLE DRIVE IN THE RANDOM 'T' COMMAND MODE OR SEQUENTIAL 'T' COMMAND MODE, IS DETERMINED BY ONE OF THE FOLLOWING CONDITIONS.

A. IF PARAMETER 'ENDING' IS 1, END OF PASS OCCURS WHEN THE DRIVE HAS READ 4.128 X 10↑9 BITS (2.58 X 10↑8 WORDS). IF SW09=1, THE END OF PASS OCCURS WHEN THE DRIVE HAS READ 1.032 X 10↑9 BITS (.645 X 10↑8 WORDS).

B. IF PARAMETER 'ENDING' IS 0, END OF PASS OCCURS WHEN THE DRIVE HAS PERFORMED 1 X 10↑6 SEEKS.

END OF PASS FOR A SINGLE DRIVE IN 'W' OR R' COMMAND MODE, IS DETERMINED AS FOLLOWS.

A. WHEN A SEQUENTIAL SEEK IS MADE BEYOND THE MAXIMUM DISK ADDRESS LIMITS SET BY THE OPERATOR, THE PASS IS CONSIDERED ENDED.

3.5.2 TEST TERMINATION

IF SW04 IS CLEAR(0), THE TEST FOR A DRIVE IS TERMINATED WHEN:

A. THE DRIVE HAS COMPLETED THE NUMBER OF PASSES SPECIFIED IN PARAMETER 'PASSES'.
B. THE TOTAL ERRORS ACCUMULATED EXCEED 25. .
C. A FATAL ERROR OCCURS: EM12 OR EM14.
D. OPERATOR DEASSIGNS THE DRIVE
E. THE NUMBER OF PASSES SPECIFIED BY THE MONITOR HAVE BEEN REACHED, WHEN RUNNING IN 'XXDP' CHAIN MODE, 'ACT11' CHAIN MODE OR 'APT' SCRIPT MODE(ANY AUTO MODE).

3.6 RUN TIME

THE EXERCISER PROGRAM MAY BE RUN IN TWO MODES. (SEE SECTION 3.5.1)
THE PROGRAM RUN TIME VARIES GREATLY DEPENDING ON THE OPERATION
MODE SELECTED, THE READ/WRITE RATIO PARAMETER (RATIO'), AND BY
SWR SWITCHES 0, 1, AND 2.

3.6.1 DATA TRANSFER MODE (DEFAULT)

ONE DRIVE - APPROX. 45 MIN. (TO REACH 4.128 x 10↑9 BITS (2.58 x 10↑8 WORDS)

3.6.2 SEEK VERIFICATION MODE

PARAMETER 'MAX WRD CNT' = 256. (1 SECTOR)
PARAMETER 'MAX TRK' = 'MIN TRK' (SAME VALUES)
PARAMETER 'MAX SEC' = 'MIN SEC' (SAME VALUES)
SW<01> =1 (NO DATA COMPARE)
SW<00> =1 (READ ONLY MODE)

ONE DRIVE - APPROX. 4.0 HRS (TO REACH 1 X 10↑6 SEEKS)

3.7 DUAL PORT OPERATION

A. LOAD THE PERFORMANCE EXERCISER PROGRAM INTO BOTH PROCESSORS.

B. SWITCH THE 'CONTROLLER SELECT' SWITCH TO 'A/B' ON EACH DRIVE WHICH IS TO BE TESTED AS A DUAL PORT DRIVE AND CYCLE THE DRIVES UP.

C. START THE PROGRAM IN EACH PROCESSOR. RUN THE PROGRAM AS THOUGH EACH PROCESSOR WERE RUNNING INDEPENDENTLY OF THE OTHER.

3.8 XXDP, ACT11, APT11 COMPATIBILITY

THIS PROGRAM IS COMPATIBLE WITH ACT11 AND APT11 IN BOTH DUMP AND AUTOMATIC MODES.

THIS PROGRAM IS ALSO, COMPATIBLE WITH XXDP IN DUMP AND CHAIN MODES, AND PROVIDES MEDIA PROTECTION IN THE CASE WHERE THE RP07 IS THE XXDP LOADING DEVICE.

AUTOMATIC MODE OR CHAIN MODE (MONITOR)

1. THE BUS ADDRESS AND CONTROLLER INTERRUPT VECTOR ARE DEFAULTED TO 176700 AND 254 RESPECTIVELY.

DUMP MODE (NO MONITOR)

1. INPUT DIALOGUE PROMPTED AFTER PROGRAM STARTS

3.9 APT ETABLE DEFINITIONS

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

K L

1. SOFTWARE ENVIRONMENT:

   = 1 IF APT SCRIPT MODE
   = 0 IF STANDLONE MODE

2. ENVIRONMENT MODE:

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

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

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

   BIT 4 TO BIT 0 ARE NOT USED

3. SWITCH 1 (SOFTWARE SWITCH REGISTER)
   IF ENVIRONMENT MODE BIT 7 (SIZING BIT) IS SET TO 1,
   THE SOFTWARE SWITCH REGISTER WILL BF USED, INSTEAD
   OF THE HARDWARE TTY CONSOLE SWITCH REGISTER.

4. SWITCH 2 (USER SWITCH REGISTER)
   NOT USED

5. CPU OPTIONS
   NOT USED

6. MEMORY TYPES 1-4 AND MAX MEMORY ADDRESSES
   NOT USED

7. INTERRUPT VECTOR 1:
   USED WHEN ENVIRONMENT MODE BIT 7 = 1;DEFAULT = 254

8. BUS PRIORITY 1:
   NOT USED.

9. INTERRUPT VECTOR 2:
   NOT USED

10. BUS PRIORITY 2:
    NOT USED

11. BASE ADDRESS:
    USED WHEN ENVIRONMENT MODE BIT 7 = 1;DEFAULT = 176700

12. DEVICE MAP:
    NOT USED

13. CONTROLLER DESCRIPTOR WORDS:
    NOT USED

14. CONTROLLER DESCRIPTOR WORDS:
    NOT USED

```
                                        ;COMMAND INITIATOR
                                        ;
                                        ;CALL
                                        ;       MOV     #DPB,R2         ;ADDRESS OF DRIVE PARAMETER BLOCK
                                        ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER
                                        ;       JSR     PC,CI????       ;CI???? = CI1, CI3, OR CI4
                                        ;                               ;WHERE:
                                        ;                               ; CI1 = DATA TRANSFER
                                        ;                               ; CI3 = SEARCH REQUESTED BY DATA XFER
                                        ;                               ; CI4 = NO DATA TRANSFER

042066  004737  046246          CI1:    JSR     PC,POPQUE       ;REMOVE REQUEST FROM "DRIVES WAIT" QUEUE
042072  010237  040574                  MOV     R2,TRNSWT       ;PUT REQ. IN TRANSFER WAIT QUEUE
042076  010203                          MOV     R2,R3           ;DPB ADDRESS TO R3
042100  013704  040640                  MOV     RPADR,R4        ;RPCS1 ADDRESS
042104  010164  000010                  MOV     R1,RPCS2(R4)    ;SELECT DRIVE
042110  122762  000135  000002          CMPB    #135,2(R2)      ;IS IT A DIAGNOSTIC COMMAND ?
042116  001011                          BNE     1$              ;BRANCH IF NOT
042120  016246  000004                  MOV     4(R2),-(SP)     ;GET THE ROUTINE NUMBER, PARAMETERS
042124  052716  100000                  BIS     #BIT15,(SP)     ;SET THE DIAGNOSTIC MODE BIT
042130  004037  045210                  JSR     R0,WRT.RP       ;WRITE THE RPMR1 REG
042134  000024                          RPMR1
042136  042750                          CI7
042140  000422                          BR      2$              ;LOAD THE COMMAND AND EXIT

042142  062703  000004          1$:     ADD     #4,R3           ;DESIRED WORD COUNT
042146  062704  000002                  ADD     #2,R4           ;RPWC ADDRESS
042152  012324                          MOV     (R3)+,(R4)+     ;LOAD WORD COUNT
042154  012324                          MOV     (R3)+,(R4)+     ;LOAD BUFFER ADDRESS
042156  012346                          MOV     (R3)+,-(SP)     ;LOAD SECTOR AND TRACK
042160  004037  045210                  JSR     R0,WRT.RP       ;CALL THE LOAD(WRITE) ROUTINE
042164  000006                          RPDA                    ;INDEX OF REGISTER TO LOAD
042166  042750                          CI7                     ;ERROR RETURN ADDRESS
042170  012346                          MOV     (R3)+,-(SP)     ;LOAD CYLINDER ADDRESS
042172  004037  045210                  JSR     R0,WRT.RP
042176  000034                          RPDC
042200  042750                          CI7
042202  004737  042232                  JSR     PC,CI2          ;SEE IF BIT15 SHOULD BE SET IN RPMR1

042206  016246  000002          2$:     MOV     2(R2),-(SP)     ;LOAD "COMMAND+GO", "A17&A16", AND "PSEL"
042212  004037  045210                  JSR     R0,WRT.RP
042216  000000                          RPCS1
042220  042750                          CI7
042222  010137  040626                  MOV     R1,DTUW         ;SET "DATA TRANSFER UNDERWAY"
042226  000137  042712                  JMP     CI5

042232  026262  000012  000156  CI2:    CMP     12(R2),FE1(R2)  ;DATA XFER TO FE CYLINDERS ?
042240  002406                          BLT     1$              ;BR IF NO
042242  012746  100000                  MOV     #BIT15,-(SP)    ;SET THE DIAGNOSTIC MODE BIT
042246  004037  045210                  JSR     R0,WRT.RP       ;WRITE THE RPMR1 REG
042252  000024                          RPMR1
042254  042750                          CI7
042256  000207                  1$:     RTS     PC

042260  013704  040640          CI3:    MOV     RPADR,R4        ;RPCS1 ADDRESS
042264  010164  000010                  MOV     R1,RPCS2(R4)    ;SELECT DRIVE
042270  016246  000012                  MOV     12(R2),-(SP)    ;DESIRED CYLINDER ADDRESS
```

EASILY. THE USER CAN SCAN SCAN DOWN A COL'PN, INSTEAD OF LOOKING THRU THE TYPICAL LONG ERROR REPORT. THE ONE DISADVANTAGE OF THE 132. CHARACTER REPORT, IS THE AMOUNT OF ERROR DATA WHICH CAN BE REPORTED TO THE USER.

THE FOLLOWING QUESTION AND WARNING MESSAGE ONLY APPEAR IN THE FIELD VERSION OF THIS DIAGNOSTIC.

'DO YOU WANT TO WRITE ANYWHERE ON MEDIA (L) N ?'

A 'N' ANSWER WILL PROCEED WITH TESTING ONLY THE FE CYLINDER AND SKIP THE FOLLOWING QUESTION. A 'Y' ANSWER WILL PROCEED TO NEXT WARNING MESSAGE AND QUESTION. IF THE PROGRAM IS IN "READ ONLY" MODE (SWO=1), THE WARNING MESSAGE WILL BE OMITTED BUT THE QUESTION WILL BE ASKED.

'! CUSTOMER DATA WILL BE OVERWRITTEN !

CONTINUE (L) ?'

A 'Y' ANSWER WILL PROCEED WITH TESTING THE ENTIRE DISK. A 'N' ANSWER WILL PROCEED WITH TESTING ONLY THE FE CYLINDER.

IF ONLY THE FE CYLINDER IS TO BE TESTED, THE FOLLOWING MESSAGE WILL BE PRINTED.

'* TESTING FE CYLINDER ONLY *'

AT THIS POINT, IF THE PROGRAM IS LOCKED IN "READ ONLY" MODE, THE FOLLOWING MESSAGE WILL BE TYPED. IF THE PROGRAM IS NOT LOCKED IN "READ ONLY" MODE, THE FOLLOWING MESSAGE WILL BE OMITTED.

'* PROGRAM IS LOCKED IN 'READ ONLY' MODE *'

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

'CHANGE PARAMETERS (L) N ?'

IF THE ENTRY IS A 'N' FOLLOWED BY A CARRIAGE RETURN OR JUST A CARRIAGE RETURN (DEFAULT), THE PROGRAM WILL NOT ALLOW ANY PARAMETERS TO CHANGED AND WILL CONTINUE.
IF THE ENTRY IS A 'Y' FOLLOWED BY CARRIAGE RETURN, THE OPERATOR WILL BE ALLOWED TO CHANGE THE PROGRAM PARAMETERS.

THE PROGRAM WILL IDENTIFY THE PARAMETER BY THE NAME GIVEN BELOW, DISPLAY THE CURRENT VALUE OF THE PARAME'ER AND WAIT FOR THE ENTRY. THE PROGRAM WILL TYPE 'INVALID ENTRY' IF THE ENTRY IS NOT CORRECT AND WAIT FOR A CORRECT ENTRY TO BE TYPED. (SEE SECTION 4.1.1)

IF THIS IS THE PROGRAM'S FIRST START, THE STATUS OF THE DRIVES ON THE SELECTED MASSBUS SUBSYSTEM WILL BE PRINTED. ON ALL SUBSEQUENT STARTS, THIS TYPEOUT MAY BE INHIBITED BY SETTING SW<02> =1.

THE FOLLOWING IS AN EXAMPLE DRIVE STATUS PRINTOUT:

```
DRIVE STATUS:
 0    ONLINE RP07
 1    LOAD DEVICE
 2    OFFLINE RP07
 3    ONLINE RP07      NON INTERLEAVED
 4    NOT PRESENT
 5    NOT AN RP07
 6    NOT PRESENT
 7    NOT PRESENT
```

THE ABOVE DRIVE STATUS SHOWS THAT DRIVE 0 WILL BE TESTED, WHILE DRIVES 1 7 WILL NOT BE TESTED.

4.1.1 KEYBOARD ENTRY PARAMETERS

QUESTION #	BASE	DEFAULT VALUE	VALUE RANGE	FUNCTION
1	10.	12800. (SEE NOTE)	6 - 12800.	CONTROLS THE MAXIMUM WORD COUNT USED FOR DATA TRANSFERS NOTE: THE PROGRAM WILL SELECT A MAXIMUM WORD COUNT, WHICH IS DETERMINED BY THE MEMORY AVAILABLE. THE MAX. WORD COUNT ASSIGNED BY THE PROGRAM IS 12800. (1 TRK) WORDS. THE OPERATOR MAY SPECIFIY ANY OTHER MAX. WORD COUNT AS LONG AS THE VALUE SPECIFIED IS AT LEAST 6 WORDS BUT NO LARGER THAN 12800. WORDS OR MEMORY AVAILABLE. (WHICH EVER VALUE IS SMALLER)
2	10.	0	0 - 32767.	DETERMINES THE INTERVAL (IN MINUTES) BETWEEN AUTOMATIC PERFORMANCE REPORT TYPEOUTS; NO TYPEOUT IF THIS PARAMETER IS 0 OR IF SW<02> =1
3	10.	15.	0 - 32767.	DETERMINES THE INTERVAL (IN MINUTES) BETWEEN DATA COMPARES TO MEMORY AFTER A READ DATA COMMAND. ALWAYS DO COMPARE IF THIS PARAMETER IS 0. IF SW01 =1, THEN NO DATA COMPARES WILL BE GRANTED,UNLESS A 'DCK' ERROR OCCURS.
4	10.	1	1 - 32767.	NUMBER OF PASSES TO END OF TEST. (THIS PARAMETER IS NOT USED WHEN THE PROGRAM IS

				OPERATING IN AUTO RUN MODE)
5	10.	0	0 15.	IF PARAMETER=0,DATA PATTERN IS RANDOMLY SELECTED. IF PARAMETER>0,SPECIFIES ONE OF THE 15. PATTERNS. THE SELECTED DATA PATTERN IS POINTED BY THE PARAMETER "PATTERN". (SEE SECTION 8.4)
6	8	000000	0 OR :	IF PARAMETER = 0, THE WORD COUNT IS RANDOMLY SELECTED BETWEEN 6 AND THE VALUE 'WRDCNT' (MAX WRD CNT). IF PARAMETER = 1, THE WORD COUNT WILL BE THE VALUE 'WRDCNT' (MAX WRD CNT).
7	8	000002	0 7	CONTROLS THE APPROXIMATE RATIO OF READ TO WRITE COMMANDS.

VALUE	R/W RATIO
0	15/1
1	7/1
2	6/2
3	5/3
4	4/4
5	3/5
6	2/6
7	1/7

8	8	000001	0 OR 1	IF PARAMETER = 1, END OF PASS DETERMINED BY THE 'WORDS READ' COUNT. IF PARAMETER = 0, END OF PASS IS DETERMINED BY THE NUMBER OF SEEKS.
9	8	000001	0 OR 1	IF EQ 1, DO AN APPROPRIATE WRITE CHECK AFTER EACH WRITE COMMAND. IF EQ 0, SELECT WRITE CHECK COMMAND RANDOMLY.
10	8	000000	0 OR 1	IF PARAMETER=0,RANDOM DATA BLOCK ADDRESS IS USED IN 'T' COMMAND IF PARAMETER=1,SEQUENTIAL DATA BLOCK IS USED IN 'T' COMMAND.

4.1.2 CHANGE DEVICE ADDRESS

THE RP/RH ADDRESS AND VECTOR MAY BE CHANGED WHEN THE PROGRAM IS STARTED AT ADDRESS 204 OR IF THE PROGRAM DOES NOT RECEIVE A RESPONSE WHEN IT ACCESSES THE DEFAULT RP/RH ADDRESS.

(DEFAULT ADDRESS = 176700, VECTOR = 254)

ADDRESS SELECTION EXAMPLES

EXAMPLE 1

```
RPCS1=176700 <CR>        ;NO CHANGE IN ADDRESS
RPVEC=000254 <CR>        ;NO CHANGE IN ADDRESS
```

EXAMPLE 2

```
RPCS1=176700 172400<CR> ;CHANGE BASE ADDRESS TO 172400
RPVEC=000254 224<CR>    ;CHANGE VECTOR ADDRESS TO 224
```

4.2 KEYBOARD COMMANDS

THROUGH THE KEYBOARD COMMANDS, THE OPERATOR MAY ASSIGN DRIVES FOR TEST ('T' COMMAND), WRITE SEQUENTIAL DATA ('W' COMMAND), PERFORM A SEQUENTIAL READ ('R' COMMAND), PERFORM WRITE DATA AND FOLLOWED BY TEST ('WT' COMMAND), REQUEST A DRIVE PERFORMANCE SUMMARY ('S' COMMAND), OR DEASSIGN A DRIVE ('D' COMMAND).

THE 'T', 'W', 'R' AND 'WT' COMMANDS ARE EXCLUSIVE TO ONE ANOTHER ON THE SAME DRIVE UNDER TEST. THE 'D' COMMAND MUST BE ENTERED IN ORDER TO ISSUE A DIFFERENT COMMAND TO THE SAME DRIVE UNDER TEST. EXCEPT FOR THE 'S' COMMAND, WHICH CAN BE ENTERED AT ANY TIME DURING THE TEST.

IF THE PROGRAM WAS STARTED AT ADDRESS 204 OR IF NO DRIVES ARE ASSIGNED FOR TESTING, THE FOLLOWING MESSAGE WILL BE TYPE BEFORE ENTERING THE COMMAND MODE. HOWEVER, IF A 'CONTROL C' IS TYPED WHILE TESTING IS IN PROGRESS, THE FOLLOWING MESSAGE WILL BE OMITTED AND THE PROGRAM WILL ENTER COMMAND MODE.

```
'NO DRIVES ASSIGNED'
```

WHEN THE PROGRAM ENTERS THE COMMAND MODE, THE FOLLOWING PROMPT WILL BE TYPED:

```
'HH:MM:SS
 ENTER COMMANDS:'
```

THE PROGRAM WILL THEN ACCEPT ANY OF THE VALID COMMANDS. AT THE COMPLETION OF A COMMAND, THE PROGRAM WILL EXIT COMMAND MODE AND TRY TO ASSIGN THE DRIVE(S) THAT WERE REQUESTED. IF THE DRIVE(S) CANNOT BE ASSIGNED, ONE OF THE FOLLOWING ERROR MESSAGES WILL BE REPORTED AND THE PROCESS CONTINUES FOR EACH DRIVE.

RESPONSE	COMMAND(S)
?DRIVE N LOAD DEVICE	T, W, R, WT
?DRIVE N OFFLINE	T, W, R, WT
?DRIVE N NOT ASSIGNED	D, S

?DRIVE N ALREADY ASSIGNED T. W. R. WT
?DRIVE N NOT PRESENT T. W. R. WT
?DRIVE N UNSAFE T. W. R. WT
?DRIVE N NOT AN RPO7 T. W. R. WT

NEXT, THE PROGRAM WILL PROCESS ALL THE ASSIGNED DRIVES AS FOLLOWS:

WHEN THE PROGRAM IS ASSIGNING THE DRIVES, THE OPERATOR WILL BE ASKED TO CHANGE THE DRIVE PARAMETERS WITH THE FOLLOWING PROMPT:

'CHANGE DRIVE PARAMETERS (L) N ?'

IF THE ENTRY IS A 'N' FOLLOWED BY A CARRIAGE RETURN OR JUST A CARRIAGE RETURN (DEFAULT), THE PROGRAM WILL NOT ALLOW ANY DRIVE PARAMETERS TO BE CHANGED AND WILL PROCEED TO TEST THE DRIVES AS COMMANDED.
IF THE ENTRY IS A 'Y' FOLLOWED BY CARRIAGE RETURN, THE OPERATOR WILL BE ALLOWED TO CHANGE THE DRIVE PARAMETERS AS FOLLOWS.

THE PROGRAM WILL FIRST TELL THE OPERATOR WHICH DRIVE IS BEING REFERENCED FOR CHANGES AND THE HARD WIRED DRV SERIAL NUMBER FORMAT:

'DRIVE # N, PGXXXX, ADDRESS LIMITS;'

WHERE 'XXXX' IS THE HARD WIRED DECIMAL SERIAL NUMBER CONTAINED IN THE RPSN REGISTER OF THE MBA. IF THE DRV SERIAL NUMBER IS NOT JUMPERED IN THE RPSN REGISTER, 'XXXX' WILL APPEAR AS '????'.
THE PROGRAM WILL REQUEST VALUES FOR THE FOLLOWING ADDRESS LIMIT PARAMETERS.

NAME	DEFAULT VALUE	VALUE RANGE	FUNCTION
MIN CYL	*	* - 630.	THE MINIMUM CYLINDER ADDRESS
MAX CYL	630.	* - 630.	THE MAXIMUM CYLINDER ADDRESS
MIN TRK	0	0 - 31.	THE MINIMUM TRACK ADDRESS
MAX TRK	31.	0 - 31.	THE MAXIMUM TRACK ADDRESS
MIN SEC	0	0 - 49.	THE MINIMUM SECTOR ADDRESS
MAX SEC	49.	0 - 49.	THE MAXIMUM SECTOR ADDRESS

* IF RUNNING THE FIELD VERSION OF THIS PROGRAM AND TESTING OCCURS ONLY ON THE FE CLYINDER, THIS VALUE WILL BE 630. IF RUNNING THE MANUFACTURES VERSION OR THE FIELD VERSION OF THIS PROGRAM AND TESTING IS ANYWHERE ON THE MEDIA, THIS VALUE WILL BE 0.

4.2.1 'T' COMMAND

USED TO ASSIGN A DRIVE(S) FOR A TEST. THIS COMMAND IS REQUIRED TO PERFORM THE TEST OF THE DRIVE(S).

FORMAT: TN<CR>

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

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

4.2.2 D' COMMAND

USED TO DEASSIGN A DRIVE(S) BEING EXERCISED.

FORMAT: DN<CR>

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

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

4.2.3 'S' CUMMAND

USED TO REQUEST A PERFORMANCE SUMMARY TYPEOUT FOR THE REFERENCED DRIVE(S). AFTER THE 'S' COMMAND HAS BEEN PERFORMED, THE PROGRAM WILL AUTOMATICALLY RESUME TESTING THE DRIVE(S) WHICH WERE UNDER TEST.

FORMAT: SN<CR>

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

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

4.2.4 W' COMMAND

USED TO PERFORM A SEQUENTIAL WRITE OF THE DISK, WITH DATA ACCEPTABLE TO THE PERFORMANCE EXERCISER PROGRAM.

FORMAT: WN<CR>

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

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

4.2.5 'R' COMMAND

USED TO PERFORM A SEQUENTIAL READ OF THE DISK.

FORMAT: RN<CR>

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

EXAMPLE: RO<CR> - READ THE DATA ON DRIVE 0.
RA<CR> - READ THE DATA ON ALL AVAILABLE DRIVES.

4.2.6 'WT' COMMAND

USED TO PERFORM A SEQUENTIAL WRITE DATA, FOLLOWED BY A 'T' COMMAND.

FORMAT: WTN<CR>

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

EXAMPLE: WT0<CR> - WRITE A DATA PATTERN AND TEST DRIVE 0
WTA<CR> - WRITE A DATA PATTERN AND TEST ALL DRIVES

5. PERFORMANCE SUMMARY TYPEOUT

5.1 THE PROGRAM WILL DISPLAY A PERFORMANCE REPORT FOR THE DRIVES BEING EXERCISED. THIS REPORT WILL BE DISPLAYED AUTOMATICALLY IF THE PARAMETER 'INTRVL' IS NOT ZERO AND SW<02>=0, OR IF THE DRIVE HAS REACHED THE DEFINED NUMBER OF PASSES AND SW<08>=0, OR IF THE OPERATOR REQUESTS TO DO SO BY USE OF THE 'S' COMMAND.

THE REPORT TYPEOUT CONTAINS THE FOLLOWING FIELDS:

'TIME'	ELAPSED TIME OF PROGRAM
'DRIVE'	DRIVE NUMBER - DRIVE TYPE
'DRV S/N'	HARD WIRED MASSBUS ADAPTER SERIAL NUMBER(RMSN)
'PASS'	PRESENT PASS COUNT FOR THE DRIVE
'WRDS WRITN /'	
' PASS'	NUMBER OF WORDS WRITTEN EACH PASS BY THE DRIVE
' TOTAL'	TOTAL NUMBER OF WORDS (X10**6) WRITTEN BY THE DRIVE
'WRDS READ /'	
' PASS'	NUMBER OF WORDS READ EACH PASS BY THE DRIVE
' TOTAL'	TOTAL NUMBER OF WORDS (X10**6) READ BY THE DRIVE
'SEEKS'	
' PASS'	NUMBER OF SEEK OPERATIONS EACH PASS BY THE DRIVE
' TOTAL'	TOTAL NUMBER OF SEEK OPERATIONS BY THE DRIVE
'SOFT'	NUMBER OF SOFT DATA ERRORS
'HARD'	NUMBER OF HARD DATA ERRORS
'SKI'	NUMBER OF 'SKI' ERRORS
'MISP'	NUMBER OF PROGRAM DETECTED POSITIONING ERRORS
'OTHER'	TOTAL ERRORS OF OTHER TYPES

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

5.2 SOFT/HARD ERROR DEFINITIONS

5.2.1 HARD ERRORS

A. A 'DTE' (DRIVE TIMING ERROR) OR A 'DCK' (DATA CHECK ERROR) WHICH OCCURS DURING A READ DATA OR A READ HEADER & DATA OPERATION AND IS NOT CORRECTABLE OR DOES NOT BECOME CORRECTABLE AFTER THE PROGRAM HAS PERFORMED THE COMPLETE RETRY SEQUENCE ON THE BAD SECTOR.

THE RETRY SEQUENCE IS 16. RE READS AT TRACK CENTER AND 2 ATTEMPTS

BOTH AT POSITIVE AND NEGATIVE OFFSETS.

5.2.2 SOFT ERRORS

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

6. DATA CHECKING & ERROR RECOVERY

6.1 DATA COMPARISON

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

A. THE COMMAND TERMINATED WITH NO ERRORS AND SW<01>=0

B. THE OPERATION TERMINATED WITH 'DCK' SET AND THE ERROR IS ECC CORRECTABLE OR THE SECTOR IN ERROR IS READ CORRECTLY AFTER RETRY ATTEMPTS.

6.2 VERIFICATION OF DATA WRITTEN

DATA VERIFICATION IS DONE EITHER THROUGH READING THE DATA BACK AND MATCHING THE DATA WITH ONE OF THE 15. PATTERNS OR THROUGH ISSUING A WRITE CHECK COMMAND AFTER DOING A WRITE DATA COMMAND.

6.3 BAD ADDRESS FLAGGING

ACCOMPLISHED BY THE RP07 HARDWARE SKIP DEFECT.

7. ERROR MESSAGES

ERRORS ARE REPORTED ON THE TTY CONSOLE. THE PROGRAM CONTAINS NO CODED ERROR HALTS. IF THE PROGRAM HALTS (ASSUMING, OF COURSE, THAT SW<15> IS NOT SET), AN UNRECOVERABLE PROGRAM CONDITION HAS OCCURRED OR A CENTRAL PROCESSOR FAILURE HAS OCCURRED.

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

7.1 ERROR DESCRIPTION LINES

(THE MESSAGE TAGS ARE GIVEN FOR REFERENCE.)

MESSAGE

TAG	TEXT
EM1	RH CONTROLLER INTERRUPT OCCURRED (ATTN=0)
	THE RH CONTROLLER INTERRUPTED AND THE ATTENTION SUMMARY REGISTER (RPAS) WAS CLEARED.
EM2	UNEXPECTED ATTENTION OCCURRED
	THE INDICATED DRIVE INTERRUPTED BUT THE DRIVE WAS NOT PERFORMING AN OPERATION.
EM3	MASSBUS PARITY ERROR (MCPE=1)
	THE RH DETECTED A CONTROL BUS PARITY ERROR WHEN READING THE INDICATED REGISTER FROM THE INDICATED DRIVE.
EM4	MASSBUS PARITY ERROR (PAR=1)
	THE INDICATED RP DETECTED A CONTROL BUS PARITY ERROR WHEN THE RH LOADED THE SPECIFIED REGISTER.
EM5	ADDRESS PLUG CHANGE BIT SET
	THE 'OPE' BIT WAS SET WHEN THE INDICATED DRIVE INTERRUPTED
EM6	RH DIDN'T RESPOND TO ADDRESSING
	WHEN THE PROGRAM ADDRESSED THE RH, NO RESPONSE WAS RECEIVED FROM THE INDICATED ADDRESS.
EM10	UNCORRECTABLE MASSBUS PARITY ERROR
	THE PROGRAM HAS TRIED 3 TIMES TO READ OR WRITE THE INDICATED REGISTER.
EM11	FATAL MASSBUS PARITY ERROR
	A CONTROL BUS PARITY ERROR OCCURRED WHEN THE RH ATTEMPTED TO PROCESS A PREVIOUS, DIFFERENT PARITY ERROR.
EM12	PERSISTENT DEVICE UNSAFE
	THE DRIVE BECAME UNSAFE; DRIVE CLEAR TO THE DRIVE DID NOT CLEAR THE UNSAFE CONDITION. THE PROGRAM WILL AUTOMATICALLY DEASSIGN THE DRIVE. THE DRIVE CANNOT BE EXERCISED UNTIL THE UNSAFE CONDITION HAS BEEN CLEARED BY MANUAL INTERVENTION.
EM13	OPERATION NOT COMPLETED WITHIN TIME LIMIT
	THE DRIVE DID NOT COMPLETE THE OPERATION WITHIN 10. SECONDS AFTER THE OPERATION WAS INITIATED.
EM14	UNIT WENT OFFLINE

THE DRIVE WENT OFFLINE DURING THE INDICATED OPERATION (THE 'RDY' BIT BECAME ZERO). THE PROGRAM WILL AUTOMATICALLY DEASSIGN THE DRIVE. THE OPERATOR MUST REASSIGN THE DRIVE WITH THE 'I' COMMAND TO RE INITIATE TESTING.

EM15 NO RESPONSE TO PORT REQUEST

THE PROGRAM IS TESTING A DUAL PORT DRIVE WHICH WAS NOT SWITCHED TO THE REQUESTING PORT WITHIN 15. SECONDS AFTER PORT REQUEST TO THE DRIVE FROM THE REPORTING PORT.

EM20 HEADER CRC ERROR

A HEADER CRC ERROR WAS DETECTED AT THE INDICATED DISK ADDRESS. THE CONTENTS OF THE HEADER ARE DISPLAYED. THE OPERATION WILL BE RETRIED 5 TIMES.

EM21 DATA CHECK ('DCK') ERROR

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

EM22 WRITE CHECK ERROR - DATA CHECK ('DCK') SET

A WRITE CHECK ERROR OCCURRED AND THE DATA CHECK ('DCK') BIT WAS SET. IF 'ECH' IS NOT SET, THE OPERATION WILL BE RETRIED UP TO 5 TIMES. IF THE 'ECH' BIT IS SET, THE OPERATION WILL BE RETRIED UP TO 16. TIMES.

EM23 WRITE CHECK ERROR - DATA CHECK ('DCK') NOT SET

A WRITE CHECK ERROR OCCURRED AND 'DCK' WAS NOT SET. THE WORDS WHICH CAUSED THE ERROR ARE DISPLAYED IN THE ERROR MESSAGE. THE OPERATION WILL BE RETRIED 5 TIMES.

EM24 HEADER READ ERROR - 'FMT' BIT DROPPED

A WRITE DATA, WRITE CHECK DATA, OR A READ DATA WAS BEING PERFORMED AND A 'FMT' ERROR OCCURRED. THE PROGRAM RE READ THE HEADER OF THE ERROR SECTOR AND THE 'HCRC' BIT WAS SET. THE CONTENTS OF THE HEADER ARE DISPLAYED. THE OPERATION WILL BE RETRIED 5 TIMES.

EM25 HEADER READ ERROR - HEADER COMPARE ('HCE') ERROR

SIMILAR TO EM24, EXCEPT THAT THE 'HCE' ERROR BIT WAS SET INITIALLY. THE OPERATION WILL BE RETRIED 5 TIMES.

EM26 FORMAT ERROR ('FER')

FORMAT ERROR OCCURRED. WHEN THE HEADER WAS RE READ, THE 'HCRC' BIT WAS NOT SET. THE CONTENTS OF THE HEADER ARE DISPLAYED. THE OPERATION WILL BE RETRIED 5 TIMES.

EM27 HEADER COMPARE ('HCE') ERROR

[illegible] AS TO ERCA EXCEPT THAT THE 'ACE' BIT WAS SET INITIALLY.
THE OPERATION WILL BE RETRIED 5 TIMES.

EM50 MISCELLANEOUS DRIVE ERROR

THIS MESSAGE IS GIVEN FOR THE FOLLOWING ERROR BITS:
ACE, DPE, IAE, OR BLE

EM51 OPERATION INCOMPLETE ('OPI') ERROR

AN OPERATION INCOMPLETE ERROR OCCURRED AT THE INDICATED SECTOR

EM52 DRIVE TIMING ('DTE') ERROR

DRIVE TIMING ERROR OCCURRED ON THE INDICATED SECTOR. THE OPERATION WILL BE RETRIED 5 TIMES.

EM53 PARITY ('PAR') ERROR AFTER OPERATION STARTED

THE 'PAR' BIT WAS SET WHEN THE OPERATION WAS COMPLETED. THE OPERATION WILL BE RETRIED 5 TIMES.

EM54 WRITE CLOCK FAILURE ('WCF')

A WRITE CLOCK FAILURE OCCURRED DURING THE OPERATION. THE OPERATION WILL BE RETRIED 5 TIMES.

EM55 INVALID ADDRESS ('IAE') ERROR

AN INVALID ADDRESS ERROR OCCURRED DURING THE OPERATION

EM56 WRITE LOCK ('WLE') ERROR

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

EM40 RH CONTROLLER OR UNIBUS TRANSFER ERROR

'TRE' IS SET IN THE RH CONTROL REGISTER AND NO DRIVE ERROR HAS OCCURRED. THE OPERATION WILL BE RETRIED 5 TIMES IF THE ERROR WAS CAUSED BY 'DLT', 'UPE', 'NEM', OR 'MDPE'.

EM41 BUS ADDRESS OR WORD COUNT INCORRECT

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

EM42 DATA COMPARE ERRORS - NO DRIVE ERROR DETECTED

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

EM43 CAN'T MATCH DATA READ WITH A PATTERN

K2

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

EM44 ERROR BIT(S) SET, BUT NO ERROR SIGNALED BY THE RH CONTROLLER

THE OPERATION COMPLETED NORMALLY, HOWEVER, THE PROGRAM FOUND EITHER ERROR BITS IN THE MP SET OR ERROR BITS IN THE RH CONTROLLER SET.

EM45 ECC LOGIC FAILURE

DURING 'BCE' ERROR PROCESSING, THE CONTENTS OF THE ECC POSITION REGISTER (SPEC1) OR THE CONTENTS OF ECC PATTERN REGISTER (SPEC2) WERE NOT VALID. THE POSITION REGISTER WAS EITHER 0 OR GREATER THAN 010040, OR THE PATTERN REGISTER CONTAINED ZEROS.

EM46 BUS ADDRESS OR WORD COUNT NOT CONSISTENT

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

EM50 SEEK INCOMPLETE ERROR

THE DRIVE SIGNALED EITHER 'SKI' ERROR.

EM51 NOT USED

EM60 DEVICE UNSAFE

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

7.2 DETAIL ERROR LINES

THE LINE NUMBERS GIVEN BELOW ARE FOR REFERENCE ONLY

LINE 1

HH:MM:SS

'HH MM SS' IS THE TIME SINCE THE PROGRAM WAS STARTED.
(HOURS, MINUTES, SECONDS)

LINE 2

'PRSNT COMMAND= XXXX PREV COMMAND= YYYY'

MNEMONICS USED FOR THE COMMANDS ARE DEFINED BELOW:

L2

SEEK	SEEK (OCTAL 5)
RECAL	RECALIBRATE (OCTAL 7)
DRVCLR	DRIVE CLEAR (OCTAL 11)
RELEASE	RELEASE (OCTAL 13)
OFFSET	OFFSET (OCTAL 15)
RTC	RETURN TO CENTERLINE (OCTAL 17)
RDHDIN	READIN PRESET (OCTAL 21)
PACK	PACK ACKNOWLEDGE (OCTAL 23)
SEARCH	SEARCH (OCTAL 31)
•GETSTS	GET REGISTERS (OCTAL 41)
•SETFMT	SET FORMAT (ECI OR NCI) (OCTAL 43)
•SELDRV	SELECT DRIVE (OCTAL 45)
WCHKD	WRITE CHECK DATA (OCTAL 51)
WCHKHD	WRITE CHECK HEADER & DATA (OCTAL 53)
WRTDAT	WRITE DATA (OCTAL 61)
FMTRK	FORMAT TRACK (WRITE HEADER & DATA) (OCTAL 63)
RDDAT	READ DATA (OCTAL 71)
RDHD	READ HEADER & DATA (OCTAL 73)

• SPECIAL RP DRIVER COMMAND (NOT A CONTROLLER COMMAND)

(DISPLAY OF THE RM/RP REGISTERS IN TWO GROUPS;
RPCS1, RPCS2, RPDS, RPER1, RPER2, RPEC1 AND RPEC2 FORM THE FIRST GROUP; ALL THE OTHER REGISTERS ARE IN THE SECOND GROUP.
IF SW<05> IS SET, ONLY THE REGISTERS IN THE FIRST GROUP WILL BE DISPLAYED.)

THE ABOVE LINE WILL BE TYPED IF THE ERROR OCCURRED DURING THE NON-DATA TRANSFER PART OF THE OPERATION.

'• ERROR AT BAD TRACK/SECTOR'

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

A WORD CALLED 'STATUS' IS DISPLAYED WITH THE RP REGISTERS. THE CONTENTS OF THIS WORD IDENTIFY HOW THE ERROR WAS PROCESSED BY THE RP DRIVE HANDLER ROUTINE. (SEE SECTION 9.7)

LINE 3

ERROR AT CXXX TYY SZZ PREV ADDR• CUUU TVV SWW

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

LINE 4

PRSNT ADDR• CXXX TYY SZZ PREV ADDR• CUUU TVV SWW

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

M2

ADDRESSES ARE GIVEN IN DECIMAL.

LINE 5

START CYL= XXX END CYL= YYY

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

LINE 6

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

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

LINE 7
-- ----

RPBA= XXXX RPWC= YYYY

THIS LINE GIVES THE CONTENTS OF THE RM CONTROLLER BUFFER ADDRESS
REGISTER AND THE RM CONTROLLER WORD COUNT REGISTER. THIS LINE IS
NOT PRINTED IF SW<05> IS NOT SET.

LINE 8

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

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

LINE 9

RPDA= XXXX RPCA= YYYY

THIS LINE GIVES THE CONTENTS OF THE RP TRACK AND SECTOR
ADDRESS REGISTER AND THE CONTENTS OF THE DESIRED CYLINDER
ADDRESS REGISTER. THIS LINE IS NOT PRINTED IF SW<05> IS NOT
SET.

LINE 10
- ----

BUFFER ADDR= XXXX WRD CNT= YYYY ACTUAL NUMBR WRDS XFRD= ZZZZ

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

LINE 11
-

EXPCTD DATA= XXXX RECEVD DATA= YYYY WORD POS= ZZZ

THIS LINE GIVES THE EXPECTED DATA, THE RECIEVED DATA FROM THE DISK, AND THE LOCATION OF THE WORD IN THE SECTOR. THE WORD POSITION IS IN DECIMAL.

LINE 12

HEADER CONTENTS OF ERROR SECTOR= XXXX XXXX XXXX XXXX

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

LINE 13

RPEC1= XXXX RPEC2= YYYY

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

LINE 14
- ----

ECC CORRECTABLE WITHOUT OFFSET

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

LINE 15

READ CORRECTLY AT (NEG OR POS) OFFSET

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

LINE 16

ECC CORRECTABLE AT (NEG OR POS) OFFSET

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

LINE 17

CORRECTED ON X RETRY

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

LINE 18

UNCORRECTABLE AFTER x RETRIES

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

LINE 19

DIFFERENT ERROR DURING RETRY

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

LINE 20

DATA COMPARISON ERRORS

A PRINTOUT OF THE DATA COMPARISON ERRORS FOLLOW THIS LINE.

LINE 21

TOTAL COMPARE ERRORS= XXXX

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

LINE 22

THE DATA COMPARED OK

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

LINE 23

ECC CORRECTION RESULTS

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

LINE 24

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

THIS IS AN INFORMATIONAL LINE WHICH WILL BE PRINTED FOR 'DCK' ERRORS
WHICH ARE ECC CORRECTABLE OR WHICH BECOME ECC CORRECTABLE DURING

RETRY. XXX IS THE WORD COUNT VALUE FROM RMFC1' AND IS IN
DECIMAL.

LINE 25

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

THE DATA ERROR WAS NOT IN DATA TRANSFERED TO MEMORY.

LINE 26

CONTENTS OF THE ERROR SECTOR (REPORTED ABOVE)

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

LINE 27

TOTALS; ERRORS:X WRDS WRITN: YYYY WRDS READ: ZZZZ

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

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

'WRDS WRITN' IS THE TOTAL NUMBER OF WORDS WRITTEN THE DRIVE.

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

LINE 28

TOTALS; SEEKS: XXX TOTAL POS ERR• YYY TOTAL SKI ERR• Z

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

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

'TOTAL POS ERR' IS THE TOTAL NUMBER OF PROGRAM DETECTED POSITIONING
ERROR BY THE DRIVE.

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

8. PROGRAM DESCRIPTION

8.1 PROGRAM OPERATION

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

THE PROGRAM SCANS ITS INTERNAL ASSIGNMENT TABLES, LOOKING FOR:

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

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

WHEN A DRIVE IS ASSIGNED, THE KEYBOARD ENTRY ROUTINE VERIFIES THAT THE DRIVE IS PRESENT, IS AN RP07, AND IS ONLINE. THE ASSIGNMENT ROUTINE THEN ISSUES A 'READIN PRESET' INSTRUCTION, SETS 'FMT16', AND ISSUES A 'RECALIBRATE' INSTRUCTION.

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

THE COMMAND INITIATION ROUTINE FIRST LOOKS AT THE CYLINDER ADDRESS OF THE REQUESTED OPERATION. IF THE DRIVE MUST SEEK TO ANOTHER CYLINDER TO PERFORM THE OPERATION, THE PROGRAM ISSUES A SEARCH INSTRUCTION TO THE DRIVE WITH A 'TARGET' SECTOR WHICH IS 1 SECTOR EARLIER THAN THE 'TRANSFER' SECTOR. (THIS ALLOWS THE PROGRAM TO INITIATE OPERATIONS ON ANOTHER DRIVE WHILE THE PRESENT DRIVE, OR OTHER DRIVES, ARE SEARCHING FOR 'TARGET' SECTORS. ALL SEEKS ISSUED BY THE PROGRAM ARE IMPLIED SEEK SEARCH OPERATIONS.) WHEN A SEARCHING DRIVE FINDS THE 'TARGET' SECTOR AND INTERRUPTS, THE PROGRAM THEN ISSUES THE REQUESTED COMMAND TO THE DRIVE THAT INTERRUPTED.

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

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

ERRORS WHICH OCCUR ARE PROCESSED IN THE FOLLOWING ORDER. MULTIPLE

ERRORS WILL BE REPORTED AS THE FIRST ERROR TYPE CHECKED.

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

OPERATION NOT COMPLETED WITHIN TIME LIMIT - EM13
UNIT WENT OFFLINE - EM14

B. ERRORS REPORTED FOR OPERATIONS WHICH COMPLETE NORMALLY.

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

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

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

8.2 DUAL PORT OPERATION

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

WHEN THE DUAL PORT HANDLER ROUTINE IN THE EXERCISER PROGRAM RECEIVES A REQUEST FOR A DRIVE, THE PROGRAM VERIFIES THAT THE DRIVE IS ONLINE. THE DRIVE IS SELECTED AND THE RPCS1 REGISTER IS READ TO TEST THE "DVA" BIT. IF THE DRIVE IS IN NEUTRAL, THIS WILL SEIZE THE DRIVE. IF THE DRIVE IS SEIZED BY THE OTHER PORT, A DRIVE CLEAR COMMAND IS ISSUED TO THE DRIVE TO SET 'PORT REQUEST'. THE PROGRAM THEN CHECKS 'DVA' IN 'RPCS1'. IF THE DRIVE IS AVAILABLE AS INDICATED BY THE 'DVA' BIT, THE COMMAND SEQUENCE WILL BE INITIATED IN THE NORMAL MANNER (SEE SECTION 8.1 ABOVE). IF 'DVA' WAS NOT SET, THE PROGRAM MAKES AN ENTRY FOR THE DRIVE IN AN INTERNAL 'PORT REQUEST PENDING' TABLE AND STARTS A 15. SECOND TIMER FOR THE DRIVE. IF THE DRIVE HAS NOT SWITCHED TO THE REQUESTING SYSTEM WITHIN THE 15. SECOND INTERVAL, THE PROGRAM REPORTS A 'NO RESPONSE TO PORT REQUEST' ERROR. NORMALLY THIS ERROR MESSAGE INDICATES A FAILURE IN THE DUAL PORT CONTROL LOGIC IN THE DRIVE BEING TESTED; HOWEVER, UNDER CERTAIN CONDITIONS

(E.G. MASSBUS PARITY ERRORS BEING REPORTED ON THE OTHER SYSTEM ON A TTY). THE OTHER PROCESSOR WAS UNABLE TO PROCESS THE DRIVE AFTER IT HAD REQUESTED THE DRIVE. THE OPERATOR MUST BE AWARE OF WHAT THE OTHER SYSTEM IS DOING AT ALL TIMES TO INTERPRET THE PORT RELATED ERROR MESSAGES PROPERLY.

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

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

8.3 SELECTION OF OPERATION VARIABLES

A. SECTOR ADDRESS SELECTION IS RANDOM BETWEEN THE VALUES IN 'MINSEC' AND 'MAXSEC'. TRACK ADDRESS SELECTION IS RANDOM BETWEEN THE VALUES IN 'MINTRK' AND 'MAXTRK'. CYLINDER ADDRESS SELECTION IS RANDOM BETWEEN 'MINCYL' AND 'MAXCYL'. IF A MINIMUM ADDRESS IS GREATER THAN THE CORRESPONDING MAXIMUM ADDRESS, THE PROGRAM WILL SWAP 'MAX' AND 'MIN' ADDRESSES AND CONTINUE.

B. THE WORD COUNT IS RANDOMLY SELECTED BETWEEN 6 AND THE VALUE 'WRDCNT' (MAX WRD CNT). THIS IS NECESSARY AS THE PROGRAM REQUIRES 4 LOCATIONS IN THE DATA PORTION OF THE SECTOR TO BE ABLE TO MATCH THE DATA TO A PATTERN FOR DATA COMPARISON PURPOSES AND NEEDS 2 MORE LOCATIONS IF A READ HEADER & DATA COMMAND IS ISSUED.

C. THE DATA WRITTEN IS RANDOMLY SELECTED AMONG THE 15 STANDARD PATTERNS. THE PARAMETER 'PATTERN' ENABLES THE RANDOM PATTERN SELECTION, IF THIS PARAMETER IS 0.

D. THE COMMANDS ARE SELECTED RANDOMLY WRITE CHECK DATA COMMAND IS PERFORMED ONLY IF THE PREVIOUS COMMAND WAS THE APPROPRIATE WRITE DATA COMMAND.

8.4 DATA PATTERNS

THE PROGRAM SELECTS ONE OF THE FOLLOWING DATA PATTERNS TO WRITE WHEN A WRITE COMMAND IS SELECTED. THE ENTIRE BUFFER IS FILLED WITH THE SELECTED PATTERN. WHEN DATA IS READ FROM THE DISK, THE PROGRAM COMPARES DATA ON A SECTOR BASIS. IF THE PARAMETER 'PATTERN' IS 0 THE PROGRAM WILL ATTEMPT TO MATCH THE FIRST 4 DATA WORDS OF EACH SECTOR, TO ONE OF THE FOLLOWING PATTERNS. HOWEVER, IF THE PARAMETER 'PATTERN' IS NOT 0, THE PROGRAM WILL ASSUME THAT THE DESIRED DATA PATTERN IN LOCATION 'PATTERN' IS THE DATA TO LOOK FOR AND WILL NOT TRY TO MATCH ANY PATTERNS. THIS ALLOWS THE OPERATOR TO SCAN THE DISK FOR ANY SPECIFIC PATTERN.

PAT 1 PAT 2 PAT 3 PAT 4 PAT 5 PAT 6 PAT 7 PAT 8

000001	177776	000000	111111	052525	147756	155555	030221
000003	177774	000000	111111	052525	147756	111111	030221
000007	177770	000000	111111	052525	147756	155555	030221
000017	177760	177777	111111	125252	147756	111111	030221
000037	177740	177777	111111	125252	147756	155555	030221
000077	177700	177777	111111	125252	147756	111111	030221
000177	177600	000000	111111	052525	147756	155555	030221
000377	177400	000000	111111	052525	147756	111111	030221
000777	177000	177777	111111	125252	147756	155555	030221
001777	176000	177777	111111	125252	147756	111111	030221
003777	174000	000000	111111	052525	147756	155555	030221
007777	170000	177777	111111	125252	147756	111111	030221
017777	160000	000000	111111	052525	147756	155555	030221
037777	140000	177777	111111	125252	147756	111111	030221
077777	100000	000000	111111	052525	147756	155555	030221
177777	000000	177777	111111	125252	147756	111111	030221

PAT 9	PAT 10	PAT 11	PAT 12	PAT 13	PAT 14	PAT 15
000001	177776	172666	077777	155555	000000	177777
000002	177775	155555	137777	066667	177777	000000
000004	177773	172666	157777	155555	177777	000000
000010	177767	155555	167777	066667	177777	000000
000020	177757	172666	173777	155555	177777	000000
000040	177737	155555	175777	066667	177777	000000
000100	177677	172666	176777	155555	177777	000000
000200	177577	155555	177577	066667	177777	000000
000400	177377	172666	177577	155555	177777	000000
001000	176777	155555	177677	066667	177777	000000
002000	175777	172666	177737	155555	177777	000000
004000	173777	155555	177757	066667	177777	000000
010000	167777	172666	177767	155555	177777	000000
020000	157777	155555	177773	066667	177777	000000
040000	137777	172666	177775	155555	177777	000000
100000	077777	155555	177776	066667	177777	000000

• WORST CASE PATTERN

9.1 RM/RP DRIVER

THIS DOCUMENT IS THE USER'S GUIDE FOR THE RM/RP DRIVER.

9.2 TO INITIALIZE THE DRIVER:

```
        JSR     PC,RPINIT
        RETURN
```

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

DRVSTA	DRIVE STATE

```
                -0              ONLINE
                -0              OFFLINE, DRIVE
                                IS NOT AN RP07, OR
                                NONEXISTENT DRIVE
                -0              UNSAFE
```

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

DRVTYP	CONDITION
0	NONEXISTENT DRIVE
4	RM03
5	RP07
-1	NOT AN RP07

THE 'RPDRST' ROUTINE WILL DO A RESTORE AND WILL SET DRVTYP.

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

CALL:

```
        JSR     R0,@DPTR        ;MAKE THE CALL
        DPTDPB                  ;ADDRESS OF DPDB
        RETURN1                 ;RETURN IF QUEUE IS FULL
        RETURN2                 ;RETURN IF REQUEST IS IN
                                ;OR A HARD ERROR IS AN
                                ;ERROR CONDITION
```

DPDB (DATA PARAMETER BLOCK)

```
DPTDPB: .BYTE   0               ;(0) DRIVE NUMBER
        .BYTE   0               ;(1) OFFSET VALUE OR FMT16, ECT, AND HCI
        .BYTE   0               ;(2) COMMAND
        .BYTE   0               ;(3) PSEL AND A17 AND A16
        .WORD   0               ;(4) WORD COUNT (MUST BE NEG.)
        .WORD   0               ;(6) BUFFER ADDRESS OR
                                ;REGISTER TABLE POINTER
        .BYTE   0               ;(10) SECTOR ADDRESS OR
                                ;FIRST REG. INDEX
        .BYTE   0               ;(11) TRACK ADDRESS OR
                                ;LAST REG. INDEX
        .WORD   0               ;(12) CYLINDER ADDRESS
        .WORD   0               ;(14) ERROR TABLE POINTER
                                ;POINTS TO THE FIRST OF TWENTY
                                ;LOCATIONS OF WHERE THE DRIVER
                                ;IS TO STORE THE RP/RM
                                ;REGISTERS ON AN ERROR. IF LEFT
                                ;ZERO REGISTERS ARE NOT SAVED.
        .WORD   0               ;(16) STATUS/ERROR INDICATOR
                                ;BIT15=1=ERROR OCCURRED
                                ;BIT07=1=DONE
                                ;BIT14-BIT09 AND BIT06-BIT03
                                ;INDICATE TYPE OF ERROR
```

9.4 [illegible]
[illegible]
[illegible]

```
        [illegible]     [illegible]     ;[illegible]
                                        ;[illegible]
        [illegible]     [illegible]     ;[illegible]
```

[illegible]
[illegible]

9.4.1 [illegible]

```
10$:    [illegible]     [illegible]     ;[illegible]
        [illegible]                     ;[illegible]
        [illegible]     10              ;[illegible]
20$:    [illegible]     [illegible]     ;[illegible]
        [illegible]     #0              ;[illegible]
        [illegible]     [illegible]     ;[illegible]
                .
                .
                .
[illegible]:    .BYTE   5               ;[illegible]
                .BYTE   0               ;
                .BYTE   101             ;[illegible]
                .BYTE   0               ;
                .WORD   1000            ;[illegible]
                .WORD   [illegible]     ;[illegible]
                .BYTE   3               ;[illegible]
                .BYTE   5               ;[illegible]
                .WORD   400             ;[illegible]
                .WORD   [illegible]     ;[illegible]
                .WORD   0               ;[illegible]

        [illegible]

[illegible]:    .WORD   5               ;[illegible]
                .WORD   [illegible]     ;[illegible]
                .WORD   -1000.          ;[illegible]
                .WORD   [illegible]     ;[illegible]
                .BYTE   3,5
                .WORD   400.[illegible].0
```

9.5 [illegible]

[illegible]	[illegible]
[illegible]	0
[illegible]	2
[illegible]	4
[illegible]	6
[illegible]	10
[illegible]	12
[illegible]	14
[illegible]	16

[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]

* [illegible] CONTROLLER REGISTERS

9.6 COMMANDS PERFORMED BY THE DRIVER

COMMAND	CODE	COMMAND TYPE
SEEK	105	P
RECALIBRATE	107	P
DRIVE CLEAR	111	H
RELEASE	113	H
OFFSET	115	P
RETURN TO CENTER	117	P
READIN PRESET	121	H
PACK ACKNOWLEDGE	123	H
SEARCH	131	P
GET REGISTER(S)	141	S
SET FORMAT	143	S
SELECT DRIVE	145	S
WRITE CHECK DATA	151	D
WRITE CHECK HEADER & DATA	153	D
WRITE DATA	161	D
WRITE HEADER & DATA	163	D
READ DATA	171	D
READ HEADER & DATA	173	D

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

9.7 DPD STATUS/ERROR INDICATOR WORD

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

BIT NO.	MEANING IF ON A "1"
15	ERROR OCCURRED

[illegible] (BITS [illegible]), BITS [illegible] SPECIFIES TYPE
[illegible] (BITS [illegible]), BITS [illegible] SPECIFIES TYPE

14(1)	USER MADE A REQUEST FOR A FUNCTION TO BE PERFORMED ON AN OFFLINE OR UNSAFE DRIVE
13(1)	USER MADE A REQUEST FOR A FUNCTION TO BE PERFORMED ON A DRIVE THAT HAS AN UNLOAD REQUEST IN QUEUE
12(2)	PERSISTENT UNSAFE CONDITION EXIST
11(2)	UNCORRECTABLE PARITY ERROR OCCURRED
10(2)(4)	FATAL PARITY ERROR. A MASSBUS INIT WAS PERFORMED, ALL QUEUES WERE EMPTIED, AND ALL DRIVES SET TO THE IDLE STATE
9(3)(4)	SOFTWARE TIMEOUT OCCURRED ON THIS DRIVE
8(4)	SOFTWARE TIMEOUT OCCURRED ON ANOTHER DRIVE
7	NONE
6(2)	ERROR OCCURRED DURING AN I/O OPERATION
5(2)	ERROR OCCURRED DURING AN OPERATION OTHER THAN I/O
4(2)	CORRECTABLE UNSAFE CONDITION OCCURRED
3(2)	DRIVE ERROR OCCURRED THAT CAUSED AN AUTOMATIC "RECALIBRATE" SEQUENCE
2	PORT REQUEST TIMEOUT. THE DRIVER REQUESTED THE DRIVE BUT THE OPPOSITE PORT DID NOT RELEASE THE DRIVE WITHIN 15. SECONDS
1	NON-EXISTENT DRIVE REQUESTED. USER MADE A REQUEST FOR A NON-EXISTENT DRIVE.

....

NOTES FOR ABOVE

....

(1) •	REQUEST WASN'T PUT IN QUEUE. (RH/RP REGISTERS WERE NOT SAVED)
(2) •	REQUEST QUEUE HAS BEEN EMPTIED. THE DRIVER ISSUED A "DRIVE CLEAR" TO THE DRIVE. NOTE: ALL RH/RP REGISTERS ARE SAVED AS PER BPO-14 BEFORE THE "DRIVE CLEAR".
(3) •	REQUEST QUEUE HAS BEEN EMPTIED. THE DRIVER ISSUED A MASSBUS INIT. ALL RH/RP REGISTERS FOR THE DRIVE WERE SAVED AS PER BPO-14 BEFORE THE INIT.

(4) • [illegible]

9.0 [illegible]

[illegible]

[illegible]

#	TYPE	DATA AVAILABLE
1	[illegible] OCCURRED (ERR<1>=0)	<BA> UNIBUS ADDRESS
2	[illegible] OCCURRED	DR> DRIVE NUMBER BA> BAD BIT <BA> UNIBUS ADDRESS ER> (ERR<15:0>) ERR<15> =DRDY ERR<14:3>=ERR<14:3> ERR<2>=ERR<2> ERR<1:0>=ERR<1:0>
3	[illegible] (ERR<15> (PE)=1)	RD <BA> ADDRESS OF REG. READ RD <RD> WORD READ
4	[illegible] (ERR<15> (PAR)=1)	WRT <BA> ADDRESS OF REG. WRITTEN WRT <WD> WORD WRITTEN RD <RD> WORD READ BACK
5	ADDRESS PLUG CHANGE BIT SET ('DPE' ERROR)	DR> DRIVE NUMBER BA> BAD BIT <BA> UNIBUS ADDRESS ER> (ERR<15:0>) ERR<15> =DRDY ERR<14:3>=ERR<14:3> ERR<2>=ERR<2> ERR<1:0>=ERR<1:0>

• THIS IS THE ACTUAL UNIBUS ADDRESS (176700)

.REM @

VERSION (CZRJ0-A 01)

1. THIS VERSION IS THE STARTING POINT FOR CX DIAGNOSTIC SUPPORT OF THE RP07 DISK DRIVE.

VERSION (CZRJ0-B-0)

1. WHEN A BAD SECTOR ERROR (BSE) AND A DATA CHECK (DCK) ARE BOTH SET DURING A READ HEADER AND DATA COMMAND, THE PROGRAM REPORTS THE DCK ERROR, WHILE IGNORING THE BSE. THE PROGRAM HAS BEEN MODIFIED TO LOOK AT THE BSE BIT AFTER THE DCK BIT IS DETECTED, SO THE ERROR CAN BE TREATED AS A BAD SECTOR.

@

```
                        ;*LAST REVISION 25-MAY-83
                        ;
                        .TITLE  CZRJOBO RP07 PERF EXER
                        ;*COPYRIGHT (C) 1983
                        ;*DIGITAL EQUIPMENT CORPORATION
                        ;*COLORADO SPGS., CO. 80963
                        ;*
                        ;*PROGRAM BY MIKE LEAVITT
                        ;*
                        ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                        ;*PACKAGE (MAINDEC 11 DZQAC-C5), 18-MAR 81
                        ;*
                        .SBTTL  OPERATIONAL SWITCH SETTINGS
                        ;*
                        ;*      SWITCH                  USE
                        ;*      ------ -        ------------------
                        ;*        15             HALT ON ERROR
                        ;*        13             INHIBIT ERROR TYPEOUTS
                        ;*        10             BELL ON ERROR
                        ;*         8             INHIBIT END OF PASS MESSAGES
                        ;*         7             DISPLAY ALL DATA COMPARE ERRORS
                        ;*         6             DON'T CHANGE PARAMETERS (LOOP ON PRESENT VALUES)
                        ;*         5             A. PARTIAL REGISTER DISPLAY IF ERROR
                        ;*                       B. NO ECC CORRECTION RESULTS DISPLAYED IF ERROR
                        ;*         4             A. DO NOT CHECK FOR MAXIMUM ERROR COUNTS
                        ;*                       B. DO NOT DROP DRIVE AT END OF TEST
                        ;*         3             A. DISPLAY ERROR SECTOR IF 'DCK', 'DTE', OR 'WCF' ERROR
                        ;*                       B. DISPLAY SECTOR IF 'DCK' ERR UNCORRECTABLE AFTER
                        ;*                          28TH RETRY
                        ;*                       C. IF DATA COMPARE ERROR & SW07 SET, DISPLAY
                        ;*                          REMAINDER OF BUFFER
                        ;*         2             A. DO NOT TYPE DRIVE STATUS AT PROGRAM START
                        ;*                       B. DO NOT TYPE PERFORMANCE REPORT AFTER SPECIFIED TIME
                        ;*         1             INHIBIT DATA COMPARE AFTER READ W/O 'DCK' ERROR
                        ;*         0             READ ONLY MODE
                        .SBTTL  BASIC DEFINITIONS

                        ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100                  STACK   = 1100
104000                  ERROR   = EMT           ;;BASIC DEFINITION OF ERROR CALL
000004                  SCOPE   = IOT           ;;BASIC DEFINITION OF SCOPE CALL

                        ;*MISCELLANEOUS DEFINITIONS
000011                  HT      = 11            ;;CODE FOR HORIZONTAL TAB
000012                  LF      = 12            ;;CODE FOR LINE FEED
000015                  CR      = 15            ;;CODE FOR CARRIAGE RETURN
000200                  CRLF    = 200           ;;CODE FOR CARRIAGE RETURN-LINE FEED
177776                  PS      = 177776        ;;PROCESSOR STATUS WORD
177776                  PSW=PS
177774                  STKLMT  = 177774        ;;STACK LIMIT REGISTER
177772                  PIRQ    = 177772        ;;PROGRAM INTERRUPT REQUEST REGISTER
177570                  DSWR    = 177570        ;;HARDWARE SWITCH REGISTER
177570                  DDISP   = 177570        ;;HARDWARE DISPLAY REGISTER

                        ;*GENERAL PURPOSE REGISTER DEFINITIONS
000000                  R0      = %0            ;;GENERAL REGISTER
```

```
000001                  R1      = %1            ;;GENERAL REGISTER
000002                  R2      = %2            ;;GENERAL REGISTER
000003                  R3      = %3            ;;GENERAL REGISTER
000004                  R4      = %4            ;;GENERAL REGISTER
000005                  R5      = %5            ;;GENERAL REGISTER
000006                  R6      = %6            ;;GENERAL REGISTER
000007                  R7      = %7            ;;GENERAL REGISTER
000006                  SP      = %6            ;;STACK POINTER
000007                  PC      = %7            ;;PROGRAM COUNTER

                        ;*PRIORITY LEVEL DEFINITIONS
000000                  PR0     = 0             ;;PRIORITY LEVEL 0
000040                  PR1     = 40            ;;PRIORITY LEVEL 1
000100                  PR2     = 100           ;;PRIORITY LEVEL 2
000140                  PR3     = 140           ;;PRIORITY LEVEL 3
000200                  PR4     = 200           ;;PRIORITY LEVEL 4
000240                  PR5     = 240           ;;PRIORITY LEVEL 5
000300                  PR6     = 300           ;;PRIORITY LEVEL 6
000340                  PR7     = 340           ;;PRIORITY LEVEL 7

                        ;*"SWITCH REGISTER" SWITCH DEFINITIONS
100000                  SW15    = 100000
040000                  SW14    = 40000
020000                  SW13    = 20000
010000                  SW12    = 10000
004000                  SW11    = 4000
002000                  SW10    = 2000
001000                  SW09    = 1000
000400                  SW08    = 400
000200                  SW07    = 200
000100                  SW06    = 100
000040                  SW05    = 40
000020                  SW04    = 20
000010                  SW03    = 10
000004                  SW02    = 4
000002                  SW01    = 2
000001                  SW00    = 1
001000                  SW9=SW09
000400                  SW8=SW08
000200                  SW7=SW07
000100                  SW6=SW06
000040                  SW5=SW05
000020                  SW4=SW04
000010                  SW3=SW03
000004                  SW2=SW02
000002                  SW1=SW01
000001                  SW0=SW00

                        ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
100000                  BIT15   = 100000
040000                  BIT14   = 40000
020000                  BIT13   = 20000
010000                  BIT12   = 10000
004000                  BIT11   = 4000
002000                  BIT10   = 2000
001000                  BIT09   = 1000
000400                  BIT08   = 400
```

```
000200                  BIT07   = 200
000100                  BIT06   = 100
000040                  BIT05   = 40
000020                  BIT04   = 20
000010                  BIT03   = 10
000004                  BIT02   = 4
000002                  BIT01   = 2
000001                  BIT00   = 1
001000                  BIT9=BIT09
000400                  BIT8=BIT08
000200                  BIT7=BIT07
000100                  BIT6=BIT06
000040                  BIT5=BIT05
000020                  BIT4=BIT04
000010                  BIT3=BIT03
000004                  BIT2=BIT02
000002                  BIT1=BIT01
000001                  BIT0=BIT00

                        ;*BASIC "CPU" TRAP VECTOR ADDRESSES
000004                  ERRVEC  = 4             ;;TIME OUT AND OTHER ERRORS
000010                  RESVEC  = 10            ;;RESERVED AND ILLEGAL INSTRUCTIONS
000014                  TBITVEC = 14            ;;"T" BIT
000014                  TRTVEC  = 14            ;;TRACE TRAP
000014                  BPTVEC  = 14            ;;BREAKPOINT TRAP (BPT)
000020                  IOTVEC  = 20            ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
000024                  PWRVEC  = 24            ;;POWER FAIL
000030                  EMTVEC  = 30            ;;EMULATOR TRAP (EMT) **ERROR**
000034                  TRAPVEC = 34            ;;"TRAP" TRAP
000060                  TKVEC   = 60            ;;TTY KEYBOARD VECTOR
000064                  TPVEC   = 64            ;;TTY PRINTER VECTOR
000240                  PIRQVEC = 240           ;;PROGRAM INTERRUPT REQUEST VECTOR
105
106                     .SBTTL  RP07 REGISTERS
107
108                     ;CONTROL AND STATUS REGISTER 1 (RPCS1)
109
110     000100          IE      = 100                   ;INTERRUPT ENABLE (BIT #6)
111     000200          RDY     = 200                   ;READY (BIT #7)
112     000400          A16     = 400                   ;HIGH ORDER BUS ADDRESS BIT (BIT #8)
113     001000          A17     = 1000                  ;HIGH ORDER BUS ADDRESS BIT (BIT #9)
114     002000          PSEL    = 2000                  ;PORT SELECT (BIT #10)
115     020000          MCPE    = 20000                 ;MASSBUSS PARITY ERROR (BIT #13)
116     040000          TRE     = 40000                 ;TRANSFER ERROR (BIT #14)
117                     ;SC     = 100000                ;SPECIAL CONDITION (BIT #15)
118
119                     ;WORD COUNT REGISTER (RPWC)
120                     ;(EACH BIT IS CALLED BY BIT NUMBER)
121
122                     ;BUS ADDRESS REGISTER (RPBA)
123                     ;(EACH BIT IS CALLED BY BIT NUMBER)
124
125                     ;CONTROL AND STATUS REGISTER 2 (RPCS2)
126
127     000001          US1     = 1                     ;UNIT SELECT (BIT #0)
128     000002          US2     = 2                     ;UNIT SELECT (BIT #1)
129     000004          US4     = 4                     ;UNIT SELECT (BIT #2)
```

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

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

                .SBTTL  RP07 REGISTERS

                ;CONTROL AND STATUS 1 REGISTER. (#00)

000001          GO      = 1             ;GO BIT (BIT #0)
000002          F0      = 2             ;FUNCTION CODE BIT #1
000004          F1      = 4             ;FUNCTION CODE BIT #2
000010          F2      = 10            ;FUNCTION CODE BIT #3
000020          F3      = 20            ;FUNCTION CODE BIT #4
000040          F4      = 40            ;FUNCTION CODE BIT #5
004000          DVA     = 4000          ;DEVICE AVAILABLE (BIT #11)

                ;DRIVE STATUS REGISTER (RPDS1) (#01)

000001          OFFON   = 1             ;OFFSET ON (BIT #0)
000002          EWN     = 2             ;EARLY WARNING (BIT #1)
000004          ILV     = 4             ;INTERLEAVE (BIT #2)
000100          VV      = 100           ;VOLUME VALID (BIT #6)
000200          DRY     = 200           ;DRIVE READY (BIT #7)
000400          DPR     = 400           ;DRIVE PRESENT (BIT #8)
001000          PGM     = 1000          ;PROGRAMABLE (BIT #9)
002000          LBT     = 2000          ;LAST SECTOR TRANSFERRED (BIT #10)
004000          WRL     = 4000          ;WRITE LOCK (BIT #11)
010000          MOL     = 10000         ;MEDIUM ON-LINE (BIT #12)
020000          PIP     = 20000         ;POSITIONING OPERATION IN PROGRESS (BIT #13)
040000          ERR     = 40000         ;COMPOSITE ERROR (BIT #14)
100000          ATA     = 100000        ;ATTENTION ACTIVE (BIT #15)

                ;ERROR REGISTER #01 (RPER1) (#02)

000001          ILF     = 1             ;ILLEGAL FUNCTION (BIT #0)
000002          ILR     = 2             ;ILLEGAL REGISTER (BIT #1)
000004          RMR     = 4             ;REGISTER MODIFICATION REFUSED (BIT #2)
000010          PAR     = 10            ;PARITY ERROR (BIT #3)
000020          FER     = 20            ;FORMAT ERROR (BIT #4)
000040          WCF     = 40            ;WRITE CLOCK FAIL (BIT #5)
000100          ECH     = 100           ;ECC HARD ERROR (BIT #6)
000200          HCE     = 200           ;HEADER COMPARE ERROR (BIT #7)
000400          HCRC    = 400           ;HEADER CRC ERROR (BIT #8)
001000          AOE     = 1000          ;ADDRESS OVERFLOW ERROR (BIT #9)
```

```
187          002000                  IAE     = 2000                  ;INVALID ADDRESS ERROR (BIT #10)
188          004000                  WLE     = 4000                  ;WRITE LOCK ERROR (BIT #11)
189          010000                  DTE     = 10000                 ;DRIVE TIMING ERROR (BIT #12)
190          020000                  OPI     = 20000                 ;OPERATION INCOMPLETE (BIT #13)
191          040000                  UNS     = 40000                 ;DRIVE UNSAFE (BIT #14)
192          100000                  DCK     = 100000                ;DATA CHECK ERROR (BIT 15)
193
194                                  ;MAINTENANCE REGISTER (RPMR1)(#03)
195
196                                  ;ATTENTION SUMMARY PSEUDO REGISTER (RPAS) (#04)
197
198          000001                  AT0     = 1                     ;DEVICE 0 (BIT #0)
199          000002                  AT1     = 2                     ;DEVICE 1 (BIT #1)
200          000004                  AT2     = 4                     ;DEVICE 2 (BIT #2)
201          000010                  AT3     = 10                    ;DEVICE 3 (BIT #3)
202          000020                  AT4     = 20                    ;DEVICE 4 (BIT #4)
203          000040                  AT5     = 40                    ;DEVICE 5 (BIT #5)
204          000100                  AT6     = 100                   ;DEVICE 6 (BIT #6)
205          000200                  AT7     = 200                   ;DEVICE 7 (BIT #7)
206
207                                  ;DESIRED SECTOR/TRACK ADDRESS REGISTER (RPDA) (#05)
208
209                                  ;DRIVE TYPE REGISTER (RPDT) (#06)
210
211          000001                  DT00    = 1                     ;DRIVE TYPE NUMBER BIT 1
212          000002                  DT01    = 2                     ;DRIVE TYPE NUMBER BIT 2
213          000004                  DT02    = 4                     ;DRIVE TYPE NUMBER BIT 3
214          000010                  DT03    = 10                    ;DRIVE TYPE NUMBER BIT 4
215          000020                  DT04    = 20                    ;DRIVE TYPE NUMBER BIT 5
216          000040                  DT05    = 40                    ;DRIVE TYPE NUMBER BIT 6
217          000100                  DT06    = 100                   ;DRIVE TYPE NUMBER BIT 7
218          000200                  DT07    = 200                   ;DRIVE TYPE NUMBER BIT 8
219          000400                  DT08    = 400                   ;DRIVE TYPE NUMBER BIT 9
220          004000                  DRQ     = 4000                  ;DRIVE REQUEST REQUIRED (BIT #11)
221          020000                  MOH     = 20000                 ;MOVING HEAD (BIT #13)
222          040000                  TAP     = 40000                 ;TAPE DRIVE (BIT #14)
223          100000                  NSA     = 100000                ;NOT SECTOR ADDRESSED (BIT #15)
224
225                                  ;LOOK-AHEAD REGISTER (RPLA) (#07)
226
227          000100                  SC1     = 100                   ;SECTOR COUNT FIELD 0 (BIT #6)
228          000200                  SC2     = 200                   ;SECTOR COUNT FIELD 1 (BIT #7)
229          000400                  SC04    = 400                   ;SECTOR COUNT FIELD 2 (BIT #8)
230          001000                  SC10    = 1000                  ;SECTOR COUNT FIELD 3 (BIT #9)
231          002000                  SC20    = 2000                  ;SECTOR COUNT FIELD 4 (BIT #10)
232          004000                  SC40    = 4000                  ;SECTOR COUNT FIELD 5 (BIT #11)
233          010000                  SC100   = 10000                 ;SECTOR COUNT FIELD 6 (BIT #12)
234
235                                  ;SERIAL NUMBER REGISTER (RPSN) (#10)
236                                  ;(EACH IS CALLED BY BIT NUMBER)
237
238                                  ;OFFSET REGISTER (RPOF) (#11)
239
240          000200                  OFFDIR  = 200                   ;RP07 OFFSET DIRECTION
241          002000                  HCI     = 2000                  ;HEADER COMPARE INHIBIT (BIT #10)
242          004000                  ECI     = 4000                  ;ERROR CORRECTION CODE INHIBIT (BIT #11)
243          010000                  FMT16   = 10000                 ;FORMAT BIT (BIT #12)
```

```
244
245                             ;DESIRED CYLINDER ADDRESS (RPDC) (012)
246                             ;(EACH BIT IS CALLED BY BIT NUMBER)
247
248                             ;CURRENT CYLINDER ADDRESS (RPCC) (013)
249                             ;(EACH BIT IS CALLED BY BIT NUMBER)
250
251                             ;RP07 ERROR REGISTER 02 (RPER2) (014)
252
253     000400                  WRTURS  = 400                   ;WRITE READY UNSAFE (BIT 8)
254     001000                  WOR     = 1000                  ;WRITE OVERRUN (BIT 9)
255     002000                  RWU1    = 2000                  ;READ/WRITE UNSAFE 01 (BIT 10)
256     004000                  RWU2    = 4000                  ;READ/WRITE UNSAFE 02 (BIT 11)
257     010000                  RWU3    = 10000                 ;READ/WRITE UNSAFE 03 (BIT 12)
258     100000                  PGE     = 100000                ;PROGRAM ERROR (BIT 15)
259
260                             ;RP07 ERROR REGISTER 03 (RPER3) (015)
261
262     000001                  DGE     = 1                     ;DIAGNOSTIC ERROR (BIT 0)
263     000002                  TPE     = 2                     ;TEMPERATURE WARNING ERROR(BIT 1)
264     000004                  AIR     = 4                     ;AIR SYSTEM WARNING ERROR (BIT 2)
265     000010                  DPE     = 10                    ;DATA PARITY ERROR (BIT 3)
266     000020                  BPE     = 20                    ;BUFFER PARITY ERROR (BIT 4)
267     000040                  DCU     = 40                    ;DC UNSAFE (BIT 5)
268     000100                  IXU     = 100                   ;INDEX UNSAFE (BIT 6)
269     000200                  DVC     = 200                   ;DEVICE CHECK (BIT 7)
270     000400                  TCF     = 400                   ;TACH CALIBRATION FAILURE (BIT 8)
271     001000                  LCE     = 1000                  ;LOSS OF CYLINDER ERROR (BIT 9)
272     002000                  LBC     = 2000                  ;LOSS OF BIT CLOCK (BIT 10)
273     040000                  SKI     = 40000                 ;SEEK INCOMPLETE (BIT 14)
274     100000                  BSE     = 100000                ;BAD SECTOR ERROR (BIT 15)
275
276                             ;ECC POSITION REGISTER (RPEC1) (016)
277                             ;(EACH BIT IS CALLED BY BIT NUMBER)
278
279                             ;ECC PATTERN REGISTER (RPEC2) (017)
280                             ;(EACH BIT IS CALLED BY BIT NUMBER)
```

```
                        .SBTTL  RP07 DRIVER COMMANDS

000101          NOOP    = 101           ;NO OPERATION
000105          SEEK    = 105           ;SEEK
000107          RECAL   = 107           ;RECALIBRATE
000111          DRVCLR  = 111           ;DRIVE CLEAR
000113          RELSE   = 113           ;RELEASE
000117          RTC     = 117           ;RETURN TO CENTER LINE
000121          READIN  = 121           ;READ IN PRESET
000131          SEARCH  = 131           ;SEARCH
000151          WCHKD   = 151           ;WRITE CHECK DATA
000153          WCHKHD  = 153           ;WRITE CHECK HEADER & DATA
000161          WRTDAT  = 161           ;WRITE DATA
000163          FMTHDR  = 163           ;FORMAT TRACK
000171          RDDAT   = 171           ;READ DATA
000173          RDHD    = 173           ;READ HEADER & DATA

176700          RBASE   = 176700
000254          AVECTI  = 254
```

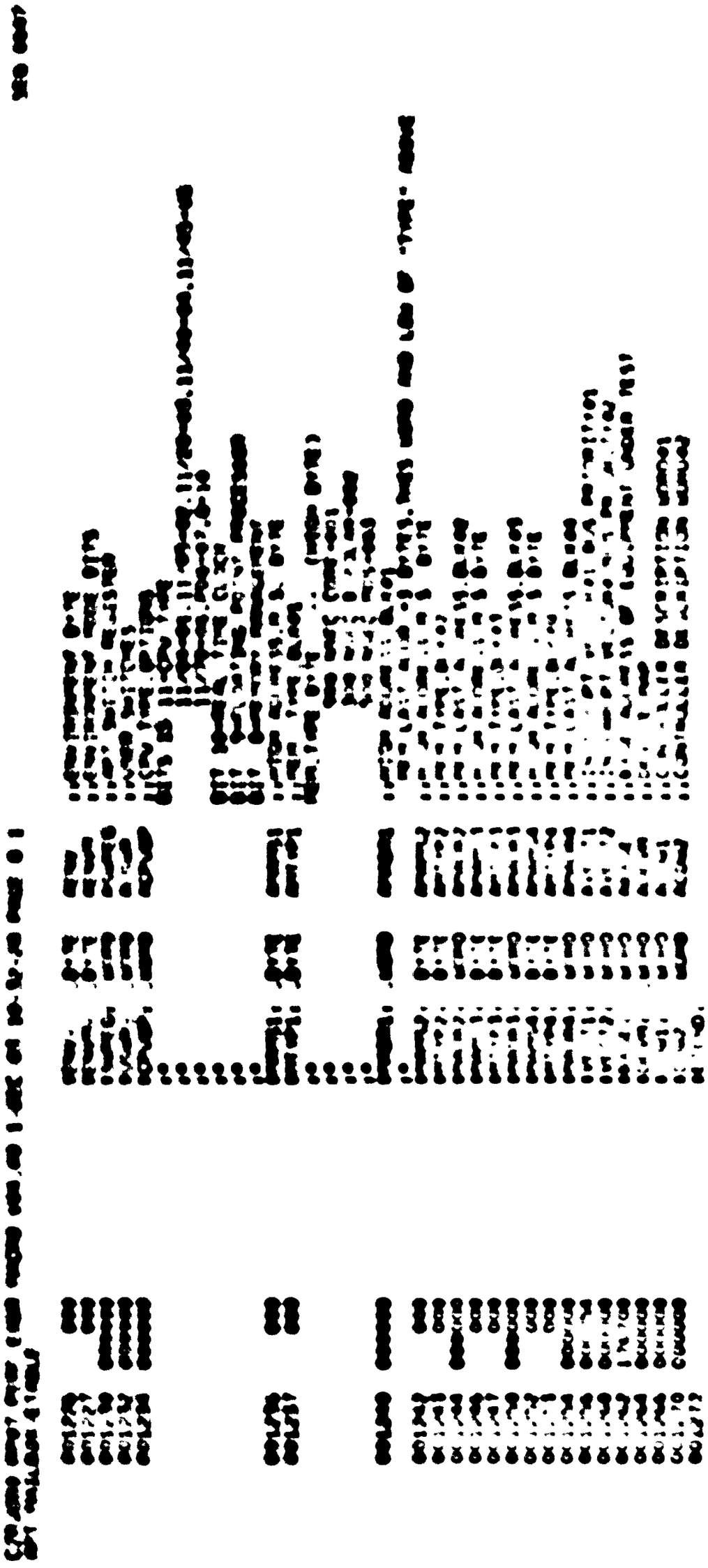
```
                        ;
                        .SBTTL  TRAP CATCHER
                        ;
000000                          .=0
                        ;CALL THIS AREA LOCATIONS FROM 0 - 776 CONTAIN A "JSR R5,2(R5)"
                        ;SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                        ;LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
000176                          .=176
000176  000000          DISPREG:: .WORD   0             ;;SOFTWARE DISPLAY REGISTER
000176  000000          SWREG::   .WORD   0             ;;SOFTWARE SWITCH REGISTER
                        ;
                        .SBTTL  STARTING ADDRESS(ES)
                        ;
000200  000137  009532          JMP     @#STARTA        ;;JUMP TO STARTING ADDRESS OF PROGRAM
                        ;
000204  000137  009522          JMP     @#STARTA        ;CHANGE THE RUN/REPOT ADDRESS
                        ;
                        .SBTTL  ACT11 HOOKS
                        ;
                        ;;*****************************************************************
                        ;HOOKS REQUIRED BY ACT11
000210                          $SAVPC=.                ;SAVE PC
000046                          .=46
000046  052056                  RERUNAD                 ;;RESET LOC.46 TO ADDRESS OF RERUNAD IN .REOP
000052                          .=52
000052  040000                  .WORD   40000           ;;RESET LOC.52 TO 40000
000210                          .=$SAVPC                ;; RESTORE PC
                        ;
001100                  .=1100
                        .SBTTL  APT PARAMETER BLOCK
                        ;;*****************************************************************
                        ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                        ;;*****************************************************************
001100                          .=.-2           ;;SAVE CURRENT LOCATION
000024                          .=24            ;;SET POWER FAIL TO POINT TO START OF PROGRAM
000024  000200                  200             ;;FOR APT START UP
000044                          .=44            ;;POINT TO APT INDIRECT ADDRESS PNTR.
000044  001100                  $APTHDR         ;;POINT TO APT HEADER BLOCK
001100                          .=.-2           ;;RESET LOCATION COUNTER
                        ;;*****************************************************************
                        ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                        ;INTERFACE SPEC.
                        ;
001100                  $APTHD:
001100  000000          $HIBTS: .WORD   0               ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
001102  001206          $MBADR: .WORD   $MAIL           ;;ADDRESS OF APT MAILBOX (BITS 0 15)
001104  014250          $ITSTM: .WORD   6500.           ;;RUN TIM OF LONGEST TEST
001106  014250          $PASTM: .WORD   6500.           ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
001110  014250          $UNITM: .WORD   6500.           ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDED UNIT
001112  000052                  .WORD   $ETEND-$MAIL/2  ;;LENGTH MAILBOX-ETABLE(WORDS)
001114                  TAB.EV=.                        ;CNTAGSTARING ADDRESS
```

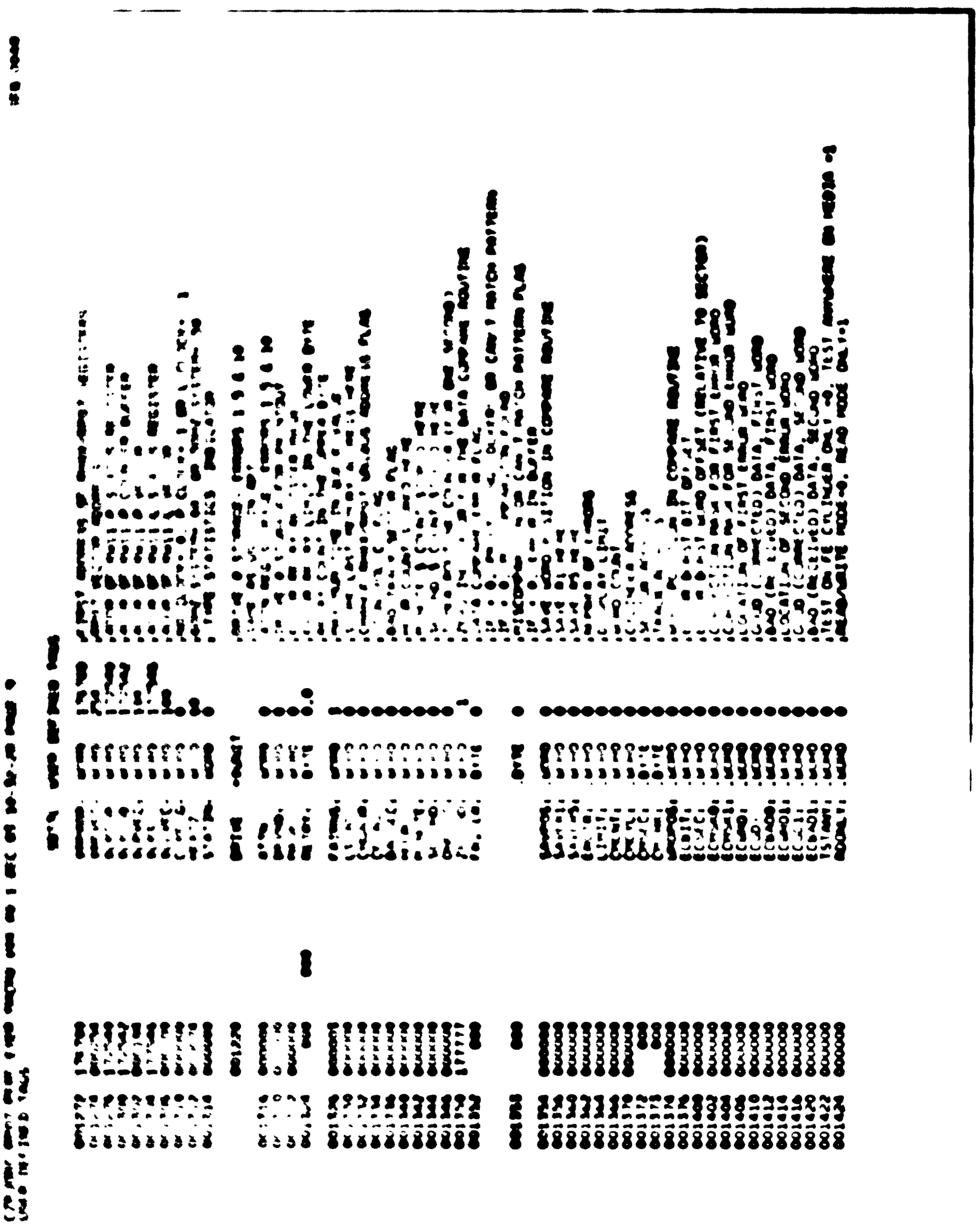
```
	.SBTTL	[illegible] COMMON TAGS

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

	001114			.=1114
001114				$CMTAG:			;;START OF COMMON TAGS
001114	000000				.WORD	0
001116	000			$TSTNM:	.BYTE	0	;;CONTAINS THE TEST NUMBER
001117	000			$ERFLG:	.BYTE	0	;;CONTAINS ERROR FLAG
001120	000000			$ICNT:	.WORD	0	;;CONTAINS SUBTEST ITERATION COUNT
001122	000000			$LPADR:	.WORD	0	;;CONTAINS SCOPE LOOP ADDRESS
001124	000000			$LPERR:	.WORD	0	;;CONTAINS SCOPE RETURN FOR ERRORS
001126	000000			$ERTTL:	.WORD	0	;;CONTAINS TOTAL ERRORS DETECTED
001130	000			$ITEMB:	.BYTE	0	;;CONTAINS ITEM CONTROL BYTE
001131	001			$ERMAX:	.BYTE	1	;;CONTAINS MAX. ERRORS PER TEST
001132	000000			$ERRPC:	.WORD	0	;;CONTAINS PC OF LAST ERROR INSTRUCTION
001134	000000			$GDADR:	.WORD	0	;;CONTAINS ADDRESS OF 'GOOD' DATA
001136	000000			$BDADR:	.WORD	0	;;CONTAINS ADDRESS OF 'BAD' DATA
001140	000000			$GDDAT:	.WORD	0	;;CONTAINS 'GOOD' DATA
001142	000000			$BDDAT:	.WORD	0	;;CONTAINS 'BAD' DATA
001144	000000				.WORD	0	;;RESERVED--NOT TO BE USED
001146	000000				.WORD	0
001150	000			$AUTOB:	.BYTE	0	;;AUTOMATIC MODE INDICATOR
001151	000			$INTAG:	.BYTE	0	;;INTERRUPT MODE INDICATOR
001152	000000				.WORD	0
001154	177570			SWR:	.WORD	DSWR	;;ADDRESS OF SWITCH REGISTER
001156	177570			DISPLAY: .WORD	DDISP	;;ADDRESS OF DISPLAY REGISTER
001160	177560			$TKS:	177560		;;TTY KBD STATUS
001162	177562			$TKB:	177562		;;TTY KBD BUFFER
001164	177564			$TPS:	177564		;;TTY PRINTER STATUS REG. ADDRESS
001166	177566			$TPB:	177566		;;TTY PRINTER BUFFER REG. ADDRESS
001170	000			$NULL:	.BYTE	0	;;CONTAINS NULL CHARACTER FOR FILLS
001171	002			$FILLS:	.BYTE	2	;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001172	012			$FILLC:	.BYTE	12	;;INSERT FILL CHARS. AFTER A "LINE FEED"
001173	000			$TPFLG:	.BYTE	0	;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
001174	000000			$TMP0:	.WORD	0	;;USER DEFINED
001176	207	377	377	$BELL:	.ASCIZ	<207><377><377>	;;CODE FOR BELL
001202	077			$QUES:	.ASCII	/?/	;;Q[illegible]STION MARK
001203	015			$CRLF:	.ASCII	<15>	;;CARRIAGE RETURN
001204	012	000		$LF:	.ASCIZ	<12>	;;LINE FEED
				;;*************************************************************
				.SBTTL	APT MAILBOX-ETABLE

				;;*************************************************************
				.EVEN
001206				$MAIL:			;;APT MAILBOX
001206	000000			$MSGTY:	.WORD	AMSGTY	;;MESSAGE TYPE CODE
001210	000000			$FATAL:	.WORD	AFATAL	;;FATAL ERROR NUMBER
001212	000000			$TESTN:	.WORD	ATESTN	;;TEST NUMBER
001214	000000			$PASS:	.WORD	APASS	;;PASS COUNT
001216	000000			$DEVCT:	.WORD	ADEVCT	;;DEVICE COUNT
001220	000000			$UNIT:	.WORD	AUNIT	;;I/O UNIT NUMBER
001222	000000			$MSGAD:	.WORD	AMSGAD	;;MESSAGE ADDRESS
001224	000000			$MSGLG:	.WORD	AMSGLG	;;MESSAGE LENGTH
001226				$ETABLE:			;;APT ENVIRONMENT TABLE
```





```
( [illegible] )                                                                    [illegible]
  [illegible]

        001476  000000                    [illegible]  .WORD   0        ;[illegible]
                                                                       ;[illegible]
                                                                       ;[illegible]
                                                                       ;[illegible]
                                                                       ;[illegible]
                                                                       ;[illegible]
                                                                       ;[illegible]
                                                                       ;[illegible]
                                                                       ;[illegible]
                                            .EVEN
        001502  000                 [illegible]:  .BYTE   [illegible]
        001503  [illegible]         [illegible]   .BYTE   [illegible]  ;[illegible]
        001504  [illegible]         [illegible]   .BYTE   [illegible]  ;[illegible]
        001505  [illegible]         [illegible]   .BYTE   [illegible]  ;[illegible]
        001506  [illegible]         [illegible]   .BYTE   [illegible]  ;[illegible]
        001507  000                 [illegible]   .BYTE   [illegible]  ;[illegible]
                                            .EVEN
                                            [illegible]

                                            [illegible]
                                            ;[illegible]
                                            ;[illegible]
                                            ;[illegible]
                                            ;[illegible]
                                            ;[illegible]
                                            ;[illegible]
                                            ;[illegible]
                                            ;[illegible]
                                            ;[illegible]

        001510  161700                      [illegible]  .WORD   161700  ;(2.50 X 10**8 WORDS) OR (4.120 X 10**9 BITS) READ
        001512  007540                                   .WORD   007540

                                            ;[illegible]
                                            ;[illegible]
                                            ;[illegible]
                                            ;[illegible]
                                            ;[illegible]
                                            ;[illegible]
                                            ;[illegible]
                                            ;[illegible]
                                            ;[illegible]

        001514  041100                      ENDSER:  .WORD   041100  ;(1 X 10**6 SECS)
        001516  000017                               .WORD   000017

        001520  000002                      MAXERR   .WORD   50.     ;MAXIMUM ERRORS ALLOWED PER DRIVE
        001522  000004                      [illegible]  .WORD   4       ;NUMBER OF COMPARE ERRORS TYPED OUT
        001524  000017  000017              CMPTIM.  .WORD   15..15. ;FIRST WORD IS THE INTERVAL BETWEEN DATA COMPARES
                                                                       ;(IN MINUTES). SECOND WORD IS THE INTERVAL COUNTER.
                                                                       ;IF FIRST WORD IS ZERO, THEN DATA COMPARE IS ALWAYS ON.
                                                                       ;THIS TIMER HAS NO AFFECT IF SWR1=1.
        001530  012000                      MAXWNT.  .WORD   12000.  ;MAXIMUM WORD COUNT (6-12000.)
        001532  000000  000000              INTVAL:  .WORD   0.0     ;FIRST WORD IS THE PERFORMANCE TYPEOUT INTERVAL
                                                                       ;(IN MINUTES). SECOND WORD IS THE INTERVAL COUNTER.
                                                                       ;IF FIRST WORD IS ZERO, NO REPORT IS TYPED.
        001534  000001                      PASSES:  .WORD   1       ;NUMBER OF PASSES TO END OF TEST (THIS PARAMETER IS
                                                                       ;NOT USED WHEN PROGRAM IS OPERATING IN AUTO RUN(CHAIN)
                                                                       ;MODE).
        001536  000000                      PATTERN: .WORD   0       ;IF EQ 0,RANDOMLY SELECT DATA PATTERN
                                                                       ;IF NOT EQ 0,SELECT ONE SET OF PATTERN
                                                                       ;POINTED BY THE "PATTERN".
        001540  000000                      RANDWC:  .WORD   0       ;IF EQ TO 0, GENERATE A RANDOM WORD COUNT
                                                                       ;FOR THE OPERATION.
                                                                       ;IF NOT EQ TO 0, USE THE VALUE IN 'WRDCNT' FOR
```

```
                                        ;THE WORD COUNT
001502  000008          RATIO:  .WORD   8       ;READ/WRITE RATIO (RANGE 0   7)
                                        ;0   15/1       (READ/WRITE)
                                        ;1 -  7/1
                                        ;2 -  6/2
                                        ;3 -  5/3
                                        ;4 -  4/4
                                        ;5 -  3/5
                                        ;6 -  2/6
                                        ;7 -  1/7
001504  000001          ENDING: .WORD   1       ;IF NOT EQ 0, END OF PASS DETERMINED
                                        ;BY THE 'WORDS READ' COUNT. (2 50 X 10**6 WORDS)
                                        ;IF EQ 0, END OF PASS DETERMINED
                                        ;BY THE SEEK COUNT. (1 X 10**6 SEEKS)
001506  000001          WRTCHK: .WORD   1       ;IF NOT EQ 0, DO AN APPROPRIATE WRITE
                                        ;CHECK AFTER EACH WRITE COMMAND
                                        ;IF EQ 0, SELECT WRITE CHECK COMMANDS
                                        ;RANDOMLY.
001510  000000          RANDOM: .WORD   0       ;IF EQ TO 0, RANDOMLY SELECT DATA BLOCK
                                        ;ADDRESS. IF NOT EQU 0,SEQUENTIALLY
                                        ;SELECT DATA BLOCK ADDRESS

                        .SBTTL  VALUES FOR FIRST OPERATION

001512  000010          BEGPAT: .WORD   8.      ;STARTING PATTERN CODE (RANGE 1 - 15.)
001514  000004          BEGCOD: .WORD   4       ;STARTING COMMAND CODE (RANGE 0 - 5)
                                        ;0 = WRITE CHECK DATA ('WCHD')
                                        ;1 = WRITE CHECK HEADER & DATA ('WCHMD' - NOT USED)
                                        ;2 = WRITE DATA ('WRTDAT')
                                        ;3 = FORMAT TRACK ('FMTRK' - NOT USED)
                                        ;4 = READ DATA ('RDDAT')
                                        ;5 = READ HEADER & DATA ('RDHD')
001516  000400          BEGWC:  .WORD   256.    ;STARTING  WRD CNT (RANGE 6 - WRDCNT)

                         SBTTL  TABLES, CONSTANTS, AND VARIABLE LOCATIONS

                        ;LIST OF DRIVES PERFORMING COMMANDS
001520  000000          ORDERQ: .WORD   0
001522  000000                  .WORD   0
001524  000000                  .WORD   0
001526  000000                  .WORD   0
001530  000000                  .WORD   0
001532  000000                  .WORD   0
001534  000000                  .WORD   0
001536  000000                  .WORD   0
001540  000000                  .WORD   0

001542  000000          ASNLST: .WORD   0               ;A BIT SET IS AN ASSIGNED DRIVE

                        ;ADDRESSES OF DRIVES TO BE DROPPED
001544  000000          DDRVS:  .WORD   0
001546  000000                  .WORD   0
001550  000000                  .WORD   0
001552  000000                  .WORD   0
001554  000000                  .WORD   0
001556  000000                  .WORD   0
001560  000000                  .WORD   0
```

```
001562  000000                          .WORD   0
001564  000000                          .WORD   0

                                ;ADDRESSES OF NEWLY ASSIGNED DRIVES
001566  000000                  NEWUNT: .WORD   0
001570  000000                          .WORD   0
001572  000000                          .WORD   0
001574  000000                          .WORD   0
001576  000000                          .WORD   0
001600  000000                          .WORD   0
001602  000000                          .WORD   0
001604  000000                          .WORD   0
001606  000000                          .WORD   0

                                ;LIST OF DRIVES WAITING FOR BUFFERS/PARAMETERS
001610  000000                  AVAIL:  .WORD   0
001612  000000                          .WORD   0
001614  000000                          .WORD   0
001616  000000                          .WORD   0
001620  000000                          .WORD   0
001622  000000                          .WORD   0
001624  000000                          .WORD   0
001626  000000                          .WORD   0
001630  000000                          .WORD   0

                                ;LIST OF DRIVES WAITING FOR BUFFERS
001632  000000                  WAIT:   .WORD   0
001634  000000                          .WORD   0
001636  000000                          .WORD   0
001640  000000                          .WORD   0
001642  000000                          .WORD   0
001644  000000                          .WORD   0
001646  000000                          .WORD   0
001650  000000                          .WORD   0
001652  000000                          .WORD   0

                                ;BUFFER ALLOCATION TABLE ENTRY COUNT
001654  000000                  BUFTBL: .WORD   0
001656  000000  000000                  .WORD   0.0
001662  000000  000000                  .WORD   0.0
001666  000000  000000                  .WORD   0.0
001672  000000  000000                  .WORD   0.0
001676  000000  000000                  .WORD   0.0
001702  000000  000000                  .WORD   0.0
001706  000000  000000                  .WORD   0.0
001712  000000  000000                  .WORD   0.0
001716  000000  000000                  .WORD   0.0
001722  000000  000000                  .WORD   0.0
001726  000000  000000                  .WORD   0.0
001732  000000  000000                  .WORD   0.0
001736  000000  000000                  .WORD   0.0
001742  000000  000000                  .WORD   0.0
001746  000000  000000                  .WORD   0.0
001752  000000  000000                  .WORD   0.0
001756  000000  000000                  .WORD   0.0
001762  000000  000000                  .WORD   0.0
```

```
001766  000000  000000                  .WORD   0,0
001772  000000  000000                  .WORD   0,0
001776  000000  000000                  .WORD   0,0
002002  000000  000000                  .WORD   0,0
002006  000000  000000                  .WORD   0,0
002012  000000  000000                  .WORD   0,0
002016  000000  000000                  .WORD   0,0
002022  000000  000000                  .WORD   0,0
002026  000000  000000                  .WORD   0,0
002032  000000  000000                  .WORD   0,0
002036  000000  000000                  .WORD   0,0
002042  000000  000000                  .WORD   0,0
002046  000000  000000                  .WORD   0,0
002052  000000  000000                  .WORD   0,0

                                ;DRIVE PARAMETER BLOCK(DPB) POINTER TABLE
002056  046314                  BLKADR: .WORD   DRIVE0  ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 0
002060  046556                          .WORD   DRIVE1  ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 1
002062  047020                          .WORD   DRIVE2  ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 2
002064  047262                          .WORD   DRIVE3  ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 3
002066  047524                          .WORD   DRIVE4  ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 4
002070  047766                          .WORD   DRIVE5  ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 5
002072  050230                          .WORD   DRIVE6  ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 6
002074  050472                          .WORD   DRIVE7  ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 7

                                ;DRIVER COMMAND CONTROL TABLE (USED IN RP DRIVER)
002076      151                 COMTBL: .BYTE   WCKD    ;WRITE CHECK DATA
002077      377                         .BYTE   -1      ;WRITE CHECK HEADER AND DATA (NOT USED)
002100      161                         .BYTE   WRTDAT  ;WRITE DATA
002101      377                         .BYTE   -1      ;FORMAT TRACK (NOT USED)
002102      171                         .BYTE   RDDAT   ;READ DATA
002103      173                         .BYTE   RDHD    ;READ HEADER AND DATA

                                ;FUNCTION(COMMAND) CODE CONTROL TABLE
002104      004                 OPTBL:  .BYTE   4       ;SEEK
002105      006                         .BYTE   6       ;RECAL
002106      010                         .BYTE   10      ;DRIVE CLEAR
002107      012                         .BYTE   12      ;RELEASE
002110      016                         .BYTE   16      ;RETURN TO CENTERLINE
002111      020                         .BYTE   20      ;READIN PRESET
002112      022                         .BYTE   22      ;PACK ACKNOWLEDGE
002113      030                         .BYTE   30      ;SEARCH
002114      050                         .BYTE   50      ;WRITE CHECK DATA
002115      052                         .BYTE   52      ;WRITE CHECK HEADER AND DATA
002116      060                         .BYTE   60      ;WRITE DATA
002117      062                         .BYTE   62      ;FORMAT TRACK
002120      070                         .BYTE   70      ;READ DATA
002121      072                         .BYTE   72      ;READ HEADER AND DATA
002122      377                         .BYTE   -1      ;TERMINATOR
                                .EVEN

                                ;MESSAGE CONTROL TABLE FOR 'OPTBL' TABLE
002124  123     105     105     MNTBL:  .ASCIZ  /SEEK   /
002134  122     105     103             .ASCIZ  /RECAL  /
002144  104     122     126             .ASCIZ  /DRVCLR /
002154  122     105     114             .ASCIZ  /RELSE  /
002164  122     124     103             .ASCIZ  /RTC    /
```

```
002174  122  105  101          .ASCIZ  /READIN /
002204  120  101  103          .ASCIZ  /PACK   /
002214  123  105  101          .ASCIZ  /SEARCH
002224  127  103  113          .ASCIZ  /WCKD   /
002234  127  103  115          .ASCIZ  /WCKMD  /
002244  127  122  124          .ASCIZ  /WRTDAT /
002254  106  115  124          .ASCIZ  /FMTRK  /
002264  122  104  104          .ASCIZ  /RDDAT  /
002274  122  104  110          .ASCIZ  /RDMD   /
002304  116  117  116          .ASCIZ  /NONE   /
```

```
                                ;STANDARD DATA PATTERN POINTER TABLE

002314  002360          STNDAT: .WORD   DATA0   ;STANDARD DATA PATTERN 0
002316  002420                  .WORD   DATA1   ;STANDARD DATA PATTERN 1
002320  002460                  .WORD   DATA2   ;STANDARD DATA PATTERN 2
002322  002520                  .WORD   DATA3   ;STANDARD DATA PATTERN 3
002324  002560                  .WORD   DATA4   ;STANDARD DATA PATTERN 4
002326  002620                  .WORD   DATA5   ;STANDARD DATA PATTERN 5
002330  002660                  .WORD   DATA6   ;STANDARD DATA PATTERN 6
002332  002720                  .WORD   DATA7   ;STANDARD DATA PATTERN 7
002334  002760                  .WORD   DATA8   ;STANDARD DATA PATTERN 8
002336  003020                  .WORD   DATA9   ;STANDARD DATA PATTERN 9
002340  003060                  .WORD   DATA10  ;STANDARD DATA PATTERN 10
002342  003120                  .WORD   DATA11  ;STANDARD DATA PATTERN 11
002344  003160                  .WORD   DATA12  ;STANDARD DATA PATTERN 12
002346  003220                  .WORD   DATA13  ;STANDARD DATA PATTERN 13
002350  003260                  .WORD   DATA14  ;STANDARD DATA PATTERN 14
002352  003320                  .WORD   DATA15  ;STANDARD DATA PATTERN 15
002354  002360                  .WORD   ZEROS   ;ALL 0 S PATTERN
002356  003262                  .WORD   ONES    ;ALL 1'S PATTERN

002360                  ZEROS:
002360  000000          DATA0:  .WORD   0       ;ALL 0 S DATA PATTERN
002362  000000                  .WORD   0
002364  000000                  .WORD   0
002366  000000                  .WORD   0
002370  000000                  .WORD   0
002372  000000                  .WORD   0
002374  000000                  .WORD   0
002376  000000                  .WORD   0
002400  000000                  .WORD   0
002402  000000                  .WORD   0
002404  000000                  .WORD   0
002406  000000                  .WORD   0
002410  000000                  .WORD   0
002412  000000                  .WORD   0
002414  000000                  .WORD   0
002416  000000                  .WORD   0

002420  000001          DATA1:  .WORD   000001  ;STANDARD PATTERN 1
002422  000003                  .WORD   000003
002424  000007                  .WORD   000007
002426  000017                  .WORD   000017
002430  000037                  .WORD   000037
002432  000077                  .WORD   000077
002434  000177                  .WORD   000177
002436  000377                  .WORD   000377
002440  000777                  .WORD   000777
002442  001777                  .WORD   001777
002444  003777                  .WORD   003777
002446  007777                  .WORD   007777
002450  017777                  .WORD   017777
002452  037777                  .WORD   037777
002454  077777                  .WORD   077777
002456  177777                  .WORD   177777

002460  177776          DATA2:  .WORD   177776  ;STANDARD PATTERN 2
```

```
002462  177774                          .WORD   177774
002464  177770                          .WORD   177770
002466  177760                          .WORD   177760
002470  177740                          .WORD   177740
002472  177700                          .WORD   177700
002474  177600                          .WORD   177600
002476  177400                          .WORD   177400
002500  177000                          .WORD   177000
002502  176000                          .WORD   176000
002504  174000                          .WORD   174000
002506  170000                          .WORD   170000
002510  160000                          .WORD   160000
002512  140000                          .WORD   140000
002514  100000                          .WORD   100000
002516  000000                          .WORD   000000

002520  000000                  DATA3:  .WORD   000000  ;STANDARD PATTERN 3
002522  000000                          .WORD   000000
002524  000000                          .WORD   000000
002526  177777                          .WORD   177777
002530  177777                          .WORD   177777
002532  177777                          .WORD   177777
002534  000000                          .WORD   000000
002536  000000                          .WORD   000000
002540  177777                          .WORD   177777
002542  177777                          .WORD   177777
002544  000000                          .WORD   000000
002546  177777                          .WORD   177777
002550  000000                          .WORD   000000
002552  177777                          .WORD   177777
002554  000000                          .WORD   000000
002556  177777                          .WORD   177777

002560  133331                  DATA4:  .WORD   133331  ;STANDARD PATTERN 4
002562  133331                          .WORD   133331
002564  133331                          .WORD   133331
002566  133331                          .WORD   133331
002570  133331                          .WORD   133331
002572  133331                          .WORD   133331
002574  133331                          .WORD   133331
002576  133331                          .WORD   133331
002600  133331                          .WORD   133331
002602  133331                          .WORD   133331
002604  133331                          .WORD   133331
002606  133331                          .WORD   133331
002610  133331                          .WORD   133331
002612  133331                          .WORD   133331
002614  133331                          .WORD   133331
002616  133331                          .WORD   133331

002620  052525                  DATA5:  .WORD   052525  ;STANDARD PATTERN 5
002622  052525                          .WORD   052525
002624  052525                          .WORD   052525
002626  125252                          .WORD   125252
002630  125252                          .WORD   125252
002632  125252                          .WORD   125252
002634  052525                          .WORD   052525
```

```
002636  052525                          .WORD   052525
002640  125252                          .WORD   125252
002642  125252                          .WORD   125252
002644  052525                          .WORD   052525
002646  125252                          .WORD   125252
002650  052525                          .WORD   052525
002652  125252                          .WORD   125252
002654  052525                          .WORD   052525
002656  125252                          .WORD   125252

002660  147556                  DATA6:  .WORD   147556  ;STANDARD PATTERN 6 (NOT WORST CASE)
002662  147556                          .WORD   147556
002664  147556                          .WORD   147556
002666  147556                          .WORD   147556
002670  147556                          .WORD   147556
002672  147556                          .WORD   147556
002674  147556                          .WORD   147556
002676  147556                          .WORD   147556
002700  147556                          .WORD   147556
002702  147556                          .WORD   147556
002704  147556                          .WORD   147556
002706  147556                          .WORD   147556
002710  147556                          .WORD   147556
002712  147556                          .WORD   147556
002714  147556                          .WORD   147556
002716  147556                          .WORD   147556

002720  155555                  DATA7:  .WORD   155555  ;STANDARD PATTERN 7
002722  133333                          .WORD   133333
002724  155555                          .WORD   155555
002726  133333                          .WORD   133333
002730  155555                          .WORD   155555
002732  133333                          .WORD   133333
002734  155555                          .WORD   155555
002736  133333                          .WORD   133333
002740  155555                          .WORD   155555
002742  133333                          .WORD   133333
002744  155555                          .WORD   155555
002746  133333                          .WORD   133333
002750  155555                          .WORD   155555
002752  133333                          .WORD   133333
002754  155555                          .WORD   155555
002756  133333                          .WORD   133333

002760  030221                  DATA8:  .WORD   030221  ;STANDARD PATTERN 8 (WORST CASE)
002762  030221                          .WORD   030221
002764  030221                          .WORD   030221
002766  030221                          .WORD   030221
002770  030221                          .WORD   030221
002772  030221                          .WORD   030221
002774  030221                          .WORD   030221
002776  030221                          .WORD   030221
003000  030221                          .WORD   030221
003002  030221                          .WORD   030221
003004  030221                          .WORD   030221
003006  030221                          .WORD   030221
003010  030221                          .WORD   030221
```

```
003012  030221                          .WORD   030221
003014  030221                          .WORD   030221
003016  030221                          .WORD   030221

003020  000001          DATA9:  .WORD   000001  ;STANDARD PATTERN 9
003022  000002                  .WORD   000002
003024  000004                  .WORD   000004
003026  000010                  .WORD   000010
003030  000020                  .WORD   000020
003032  000040                  .WORD   000040
003034  000100                  .WORD   000100
003036  000200                  .WORD   000200
003040  000400                  .WORD   000400
003042  001000                  .WORD   001000
003044  002000                  .WORD   002000
003046  004000                  .WORD   004000
003050  010000                  .WORD   010000
003052  020000                  .WORD   020000
003054  040000                  .WORD   040000
003056  100000                  .WORD   100000

003060  177776          DATA10: .WORD   177776  ;STANDARD PATTERN 10
003062  177775                  .WORD   177775
003064  177773                  .WORD   177773
003066  177767                  .WORD   177767
003070  177757                  .WORD   177757
003072  177737                  .WORD   177737
003074  177677                  .WORD   177677
003076  177577                  .WORD   177577
003100  177377                  .WORD   177377
003102  176777                  .WORD   176777
003104  175777                  .WORD   175777
003106  173777                  .WORD   173777
003110  167777                  .WORD   167777
003112  157777                  .WORD   157777
003114  137777                  .WORD   137777
003116  077777                  .WORD   077777

003120  172666          DATA11: .WORD   172666  ;STANDARD PATTERN 11
003122  155555                  .WORD   155555
003124  172666                  .WORD   172666
003126  155555                  .WORD   155555
003130  172666                  .WORD   172666
003132  155555                  .WORD   155555
003134  172666                  .WORD   172666
003136  155555                  .WORD   155555
003140  172666                  .WORD   172666
003142  155555                  .WORD   155555
003144  172666                  .WORD   172666
003146  155555                  .WORD   155555
003150  172666                  .WORD   172666
003152  155555                  .WORD   155555
003154  172666                  .WORD   172666
003156  155555                  .WORD   155555

003160  077777          DATA12: .WORD   077777  ;STANDARD PATTERN 12
003162  137777                  .WORD   137777
```

```
003166  157777                          .WORD   157777
003166  167777                          .WORD   167777
003170  173777                          .WORD   173777
003172  175777                          .WORD   175777
003174  176777                          .WORD   176777
003176  177377                          .WORD   177377
003200  177577                          .WORD   177577
003202  177677                          .WORD   177677
003204  177737                          .WORD   177737
003206  177757                          .WORD   177757
003210  177767                          .WORD   177767
003212  177773                          .WORD   177773
003214  177775                          .WORD   177775
003216  177776                          .WORD   177776

003220  125525                  DATA13: .WORD   125525  ;STANDARD PATTERN 13
003222  052252                          .WORD   052252
003224  125525                          .WORD   125525
003226  052252                          .WORD   052252
003230  125525                          .WORD   125525
003232  052252                          .WORD   052252
003234  125525                          .WORD   125525
003236  052252                          .WORD   052252
003240  125525                          .WORD   125525
003242  052252                          .WORD   052252
003244  125525                          .WORD   125525
003246  052252                          .WORD   052252
003250  125525                          .WORD   125525
003252  052252                          .WORD   052252
003254  125525                          .WORD   125525
003256  052252                          .WORD   052252

003260  000000                  DATA14: .WORD   000000  ;STANDARD PATTERN 14
003262  177777                  ONES:   .WORD   177777  ;ALL 1 S DATA PATTERN
003264  177777                          .WORD   177777
003266  177777                          .WORD   177777
003270  177777                          .WORD   177777
003272  177777                          .WORD   177777
003274  177777                          .WORD   177777
003276  177777                          .WORD   177777
003300  177777                          .WORD   177777
003302  177777                          .WORD   177777
003304  177777                          .WORD   177777
003306  177777                          .WORD   177777
003310  177777                          .WORD   177777
003312  177777                          .WORD   177777
003314  177777                          .WORD   177777
003316  177777                          .WORD   177777

003320  177777                  DATA15: .WORD   177777  ;STANDARD PATTERN 15
003322  000000                          .WORD   000000
003324  000000                          .WORD   000000
003326  000000                          .WORD   000000
003330  000000                          .WORD   000000
003332  000000                          .WORD   000000
003334  000000                          .WORD   000000
003336  000000                          .WORD   000000
```

```
                                    ;[illegible] TABLE
                                    ;[illegible] TABLE CONTAINS THE INFORMATION FOR EACH DEVICE THAT CAN OCCUR.
                                    ;[illegible] INFORMATION GIVEN IS [illegible] OF THE ENTRY WHERE DATA IS FOUND IN
                                    ;[illegible] POSITION GIVEN IS [illegible] OF THE ENTRY IN THE TABLE IS PERTINENT.
                                    ;[illegible]:     IF BIT IS 0 THE ALL ONES DATA IS (NONE)
                                    ;[illegible]:     EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

                                    ;[illegible]     [illegible]     ;[illegible] TO THE [illegible]
                                    ;[illegible]     [illegible]     ;[illegible] TO THE [illegible]
                                    ;[illegible]     [illegible]     ;[illegible] TO [illegible]
                                    ;[illegible]     [illegible]     ;[illegible] TO THE DATA FORMAT

  001560                            [illegible]:
1                                   ;[illegible] 1
2
3
4 001560  051020                            [illegible]     ;ON CONTROLLED INTERRUPT OCCURRED (NONE = 0)
5 001562  054034                            [illegible]
6 001564  054160                            [illegible]
7 001566  000000                            0
8
9                                   ;[illegible] 2
10
11 001570  051077                           [illegible]     ;UNEXPECTED ATTENTION OCCURRED
12 001572  054034                           [illegible]
13 001574  054510                           [illegible]
14 001576  000000                           0
15
16                                  ;[illegible] 3
17
18 001600  051130                           [illegible]     ;PARITY ERROR (NONE=1)
19 001602  054114                           [illegible]
20 001604  054510                           [illegible]
21 001606  000000                           0
22
23                                  ;[illegible] 4
24
25 001610  051170                           [illegible]     ;PARITY ERROR (PRO=1)
26 001612  054101                           [illegible]
27 001614  054520                           [illegible]
28 001616  000000                           0
29
30                                  ;[illegible] 5
31
32 001620  051230                           [illegible]     ;NOT USED
33 001622  054034                           [illegible]
34 001624  054510                           [illegible]
35 001626  000000                           0
36
37                                  ;[illegible] 6
38
39 001630  051264                           [illegible]     ;[illegible] DIDN'T RESPOND TO ADDRESSING
40 001632  054200                           [illegible]
41 001634  054524                           [illegible]
42 001636  000000                           0
```

```
(20 MAR 80)  (KEB MACRO V38.1 [KEB MACRO V38.1 SRC] 30-SEP-80 09:02  [KEB.SRC]
KEB001 OUTSTAT.TABLE                                                    VECTOR TABLE

                                        ;THIS MODULE HANDLES UNEXPECTED INTERRUPTS
                                        ;
000000  012400                          UNEXPTD: MOV     (SP)+,R0       ;GET PC WHERE THE TRAP OR ACCURED
000002  000400                                   BR      UNEXPTD.1      ;GO SAVE PC & R0
000004  010046                                   MOV     R0,-(SP)       ;SAVE R0 ON STACK
000006  012700  000000                          MOV     #0,R0          ;GET PSW OF TRAP
000012  000001                                   WAIT                   ;WAIT FOR THE ERROR
                                        ;UNEXPTD.1:  ;SAVE PSW WHERE IT WAS WHEN INTERRUPT OCCURED
000014                                  UNEXPTD.1:
000014  010046                                   MOV     R0,-(SP)       ;SAVE PSW WHERE IT WAS
000016  010600                                   MOV     SP,R0          ;
000020  005760  000000                          TST     PC_SAVE(R0)    ;ADJUST PC TO POINT TO TRAP
000024  000000                                   HALT                   ;HALT THE PROCESSOR
000026  000776                                   BR      .-2            ;IN CASE OF CONTINUE
                                        ;
                                        ;
                                        ;
000030  012737  177777  000150          START:  MOV     #-1,@#KWVECT   ;ALLOW INTERRUPT REQUESTS TO BE CHANGED
000036  000407                                   BR      START1         ;START THE PROGRAM
                                        ;
000040  012737  000000  000152          START1: MOV     #0,@#KWVECT+2  ;ALLOW INTERRUPT REQUESTS CHANGE PLAS
                                        ;-----------------------------------------------------------------
000046  005000                          TEST:   CLR     R0
000050  012737  000004  000114                  MOV     #4,@#114       ;SET TRAP VECTOR TO ODT TRAP ADR
                                        ;
000056  005777  000000                  START2: TST     @0             ;TRY LOOP, WAIT FOR INTERRUPT
000062  005200                                  INC     R0             ;BUMP R0
000064  000776                                  BR      .-2            ;CLEAR THE WORLD
                                        ;
                                        ;       INITIALIZE THE CONTROL AREA
                                        ;CLEAR THE CONTROL AREA (R0=0, R1=#WORDS)
000066  012700  000156                  CLEAR:  MOV     #156,R0        ;START LOCATION TO BE CLEARED
000072  005020                          CLEAR1: CLR     (R0)+          ;CLEAR MEMORY LOCATION
000074  020027  000156                          CMP     R0,#156+156    ;DONE?
000100  001374                          BNE     CLEAR1                 ;BACK IF NO
000102  012706  001000                  MOV     #1000,SP               ;RESET THE STACK POINTER
                                        ;INITIALIZE A FEW OF THE VECTORS
000106  012737  000340  000010          MOV     #340,@#10              ;SET VECTOR FOR ERROR ROUTINE
000114  012737  000340  000014          MOV     #340,@#14              ;(LEVEL 7)
000122  012737  000340  000034          MOV     #340,@#34              ;SET VECTOR FOR TRAP CALLS
000130  012737  000340  000020          MOV     #340,@#20              ;(LEVEL 7)
000136  012737  000340  000004          MOV     #340,@#4               ;SET BUS FAILURE VECTOR
000144  012737  000340  000250          MOV     #340,@#250             ;(LEVEL 7)
000152  012737  177545  001300          MOV     #177545,@#300          ;SET UP THE RANDOM NUMBER GENERATOR
000160  012737  125436  001302          MOV     #125436,@#302          ;(A 16 BIT AND A 16 BIT)
                                        ;SIZE FOR A MEMORY SWITCH REGISTER.  IF NOT FOUND OR IF 35
                                        ;CLEAR TO A -1.  SET UP FOR A SUBROUTINE SWITCH OR DISPLAY.
000170  013706  000006                  MOV     @#SYSTAB+4(SP),SP      ;SAVE THE R0 VALUE
000174  012737  000340  000000          MOV     #340,@#0               ;SET UP TRAP TO ZERO
000202  012737  177570  001156          MOV     #177570,@#SWREG        ;SET UP FOR A MEMORY SWITCH REGISTER
000210  012737  177570  001154          MOV     #177570,@#DISPLAY      ;AND A MEMORY DISPLAY REGISTER
000216  022777  177777  175240          CMP     @#-1,@#SWREG           ;TRY TO REFERENCE NONEXISTENT MEM
000224  001012                          BNE     660                    ;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                       ;AND THE NONEXISTENT MEM IS NOT = -1
000226  000001                          WAIT                           ;BRANCH IF NO TIMEOUT
000230  012716  000336                  MOV     #336,(SP)              ;SET UP FOR TRAP RETURN
000234  000002                  660:    RTI
```

```
        [illegible]                                                        ;[illegible] TO SOFTWARE SWR
        [illegible]
        [illegible]                                              ;RESTORE ERROR VECTOR
        [illegible]
        [illegible]
        [illegible]
        [illegible]                                                        ;[illegible] SWITCH REGISTER
        [illegible]                              67$:
                                                 ;SET UP [illegible] VECTOR FOR [illegible] TIMEOUT
        004770  012737  [illegible]  [illegible]          MOV    [illegible]   ;[illegible] TIMEOUT
        004776  012737  [illegible]  [illegible]          MOV    [illegible]   ;LEVEL 0

        000000  104401  001450                  TYPE   ,[illegible]  ;TYPE PROGRAM ID, REV LEVEL & PATCH LEVEL
                                         .SBTTL  TYPE PROGRAM NAME
                                         ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
        000010  [illegible]  177777             INC    #-1           ;;FIRST TIME?
        [illegible]                             BNE    60$           ;;BRANCH IF NO
        [illegible]  104401  000000             TYPE   ,60$          ;;TYPE ASCIZ STRING
        [illegible]  000[illegible]             BR     60$           ;;GET OVER THE ASCIZ
                                         ;;60$:  .ASCIZ  <CRLF>RP07 PERFORMANCE EXERCISER<CRLF>
        000062                           60$:
                                         .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
        000062  005737  000042                  TST    @#42          ;;ARE WE RUNNING UNDER XXDP/ACT?
        [illegible]                             BNE    70$           ;;BRANCH IF YES
        [illegible]  001226  000001             CMPB   #APTE,#1      ;;ARE WE RUNNING UNDER APT?
        [illegible]                             BEQ    70$           ;;BRANCH IF YES
        004100  [illegible]  001154  000176     CMP    SWR,#SWREG    ;;SOFTWARE SWITCH REG SELECTED?
        004106  [illegible]                     BNE    71$           ;;BRANCH IF NO
        004110  104406                          GTSWR                ;;GET SOFT-SWR SETTINGS
        004112  [illegible]                     BR     71$
        004114  112737  000001  001150  70$:    MOVB   #1,AUTOB      ;;GET AUTO-MODE INDICATOR
        004122                           71$:

                                         ;THE FOLLOWING FINDS OUT THE PROGRAM CONTROL MODE;
                                         ;PAPER TAPE (MANUAL), ACT11, XXDP CHAIN OR DUMP

        004122  005037  001430                  CLR    XXDP          ;CLEAR 'XXDP' LOAD DEVICE STORAGE
        004126  1[illegible]737  177777  000041  CMPB   @#1,#41      ;LOADED FROM AN RP07 ? (UNKNOWN)
        004134  0[illegible]1121                BNE    30            ;BR IF NOT
        004136  01[illegible]37  000040  001430 MOV    @#40,XXDP     ;GET DEVICE INDICATOR AND NUMBER
        004144  1[illegible]737  000007  001430 CMPB   #7,XXDP       ;IS IT A VALID NUMBER ?
        004152  1[illegible]002                 BHIS   10            ;YES
        004154  1[illegible]37  001430          CLRB   XXDP          ;NO, DEFAULT TO DRIVE 0
        004160  005737  000042           10:    TST    @#42          ;CHAIN MODE OR ACT11 AUTO ACCEPT ?
        004164  001425                          BEQ    20            ;BR IF NEITHER
        004166  104401  004176                  TYPE   ,73$          ;;TYPE ASCIZ STRING
        004172  000412                          BR     72$           ;;GET OVER THE ASCIZ
                                         ;;73$:  .ASCIZ  <CRLF>NOT TESTING DRIVE /
        004220                           72$:
        004220  005046                          CLR    -(SP)         ;CLEAR WORD ON STACK
        004222  113716  001430                  MOVB   XXDP,(SP)     ;GET DRIVE ADDRESS
        004226  104405                          TYPOS                ;TYPE THE ADDRESS
        004230     001                          .BYTE  1             ;ONLY 1 CHARACTER
        004231     000                          .BYTE  0             ;SUPRESS LEADING ZEROS
        004232  104401  001203                  TYPE   ,$CRLF        ;CR-LF
        004236  000460                          BR     30            ;GET NUMBER OF DRIVES
```

M5


```
004700  005227  177777          2$:     INC     @-1             ;FIRST TIME THRU HERE ?
004704  001005                          BNE     3$              ;NO
004706  104401  004734                  TYPE    ,75$            ;;TYPE ASCIZ STRING
004712  000410                          BR      76$             ;;GET OVER THE ASCIZ
                                ;;75$:  .ASCIZ  <CRLF>/TO TEST DRIVE /
004734
                                76$:
004734  005046                          CLR     -(SP)           ;CLEAR WORD ON STACK
004736  113716  001430                  MOVB    RMDP,(SP)       ;GET DRIVE ADDRESS
004742  104403                          TYPOS                   ;TYPE DRIVE ADDRESS
004744     001                          .BYTE   1               ;ONLY 1 CHARACTER
004745     000                          .BYTE   0               ;SUPRESS LEADING ZEROS
004746  104401  004314                  TYPE    ,76$            ;;TYPE ASCIZ STRING
004312  000432                          BR      3$              ;;GET OVER THE ASCIZ
                                ;;76$:  .ASCIZ  /, HALT PROGRAM, CLEAR LOC. 40 AND RESTART PROGRAM./<CRLF>
004400
                                3$:
004400  004737  033510                  JSR     PC,$TKINT       ;TURN ON THE KEYBOARD INTERRUPT
004404  105737  001226                  TSTB    $ENV            ;RUN UNDER APT MODE
004410  001415                          BEQ     5$              ;NO,DO NOT BOTHER
004412  105737  001256                  TSTB    $VECT1          ;NEW VECTOR ?
004416  001403                          BEQ     4$              ;NOT LOAD IF = 0
004420  113737  001256  001274          MOVB    $VECT1,@RPVEC   ;NEW VECTOR
004426  005737  001262          4$:     TST     $BASE           ;NEW BASE ADDRESS ?
004432  001411                          BEQ     6$              ;NO
004434  013737  001262  001272          MOV     $BASE,@RPADR    ;NEW BASE ADDRESS
004442  000405                          BR      6$

004444  105737  001150          5$:     TSTB    $AUTOB          ;RUNNING IN AUTO MODE ?
004450  001002                          BNE     6$              ;YES
004452  004737  062476                  JSR     PC,BUSADR       ;CHECK RMXX/RP07 BUS ADDRESS
004456  013737  001272  040640  6$:     MOV     @RPADR,RPADR    ;LOAD ADDRESS INTO DRIVER
004464  013737  001274  040642          MOV     @RPVEC,RPVEC    ;LOAD VECTOR INTO DRIVER
004472  005037  001314                  CLR     STATIN          ;CLEAR PERFORMANCE SUMMARY TYPEOUT FLAG
004476  012705  001520                  MOV     #ORDER0,R5      ;START OF AREA TO CLEAR
004502  005025                  7$:     CLR     (R5)+
004504  022705  002056                  CMP     #BLKADR,R5      ;LOOK FOR END OF CLEAR AREA
004510  001374                          BNE     7$              ;BR IF NOT FINISHED
004512  012706  001100                  MOV     #STACK,SP       ;SETUP THE STACK POINTER
004516  005037  177776                  CLR     PS              ;CLEAR THE PROCESSOR STATUS WORD
004522  013737  001312  001346          MOV     HERTZ,ONESEC    ;RESTORE ONE SECOND COUNTER VALUE
004530  005037  001340                  CLR     HOUR            ;CLEAR THE HOUR'S COUNTER
004534  005037  001342                  CLR     MINUTE          ;CLEAR THE MINUTE'S COUNTER
004540  005037  001344                  CLR     SECOND          ;CLEAR THE SECOND'S COUNTER
004544  005037  001472                  CLR     INTRVL+2        ;CLEAR INTERVAL COUNTER
004550  013737  001462  001464          MOV     CMPTIM,CMPTIM+2 ;INIT COMPARE TIME COUNTER
004556  005037  001334                  CLR     CFLAG           ;CLEAR THE 'CONTROL C' FLAG
004562  005037  046446                  CLR     DRIVE0+$FIRST   ;RESET $FIRST FLAG FOR DRIVE 0
004566  005037  046710                  CLR     DRIVE1+$FIRST   ;RESET $FIRST FLAG FOR DRIVE 1
004572  005037  047152                  CLR     DRIVE2+$FIRST   ;RESET $FIRST FLAG FOR DRIVE 2
004576  005037  047414                  CLR     DRIVE3+$FIRST   ;RESET $FIRST FLAG FOR DRIVE 3
004602  005037  047656                  CLR     DRIVE4+$FIRST   ;RESET $FIRST FLAG FOR DRIVE 4
004606  005037  050120                  CLR     DRIVE5+$FIRST   ;RESET $FIRST FLAG FOR DRIVE 5
004612  005037  050362                  CLR     DRIVE6+$FIRST   ;RESET $FIRST FLAG FOR DRIVE 6
004616  005037  050624                  CLR     DRIVE7+$FIRST   ;RESET $FIRST FLAG FOR DRIVE 7
004622  005037  001424                  CLR     RDONLY          ;ASSUME READ/WRITE CONDITION
004626  032777  000001  174320          BIT     #SW0,@SWR       ;IS EXERCISER IN 'READ ONLY' MODE ?
004634  001402                          BEQ     8$              ;BR IF NO
```

GET VALUE FOR SOFTWARE SWITCH REGISTER

```
106 004636  005237  001424                    INC     RDONLY          ;LOCK PROGRAM IN 'READ ONLY' MODE
107 004642                              8$:
108
110                                     ;SEE IF OPERATOR WANTS HELP TEXT PRINTED
111
112 004642  105737  001150                    TSTB    $AUTOB          ;AUTO MODE ?
113 004646  001035                            BNE     13$             ;BR IF YES
114 004650  005227  177777                    INC     #1              ;FIRST TIME THRU HERE ?
115 004654  001032                            BNE     13$             ;BR IF NO
116
117 004656  104401  064370            9$:     TYPE    ,MSHELP         ;TYPE 'TYPE HELP MESSAGE (L) N ? '
118 004662  104411                            RDLIN                   ;READ THE ENTRY
119 004664  012600                            MOV     (SP)+,R0        ;SAVE ADDRESS OF RESPONSE
120 004666  005737  001334                    TST     CFLAG           ;WAS (^C) TYPED?
121 004672  001013                            BNE     10$             ;BR IF YES
122 004674  105710                            TSTB    (R0)            ;WAS RESPONSE A CARRIAGE RETURN ?
123 004676  001414                            BEQ     11$             ;BR IF YES (DEFAULT)
124 004700  105760  000001                    TSTB    1(R0)           ;WAS IT TERMINATED WITH CARRIAGE RETURN ?
125 004704  001006                            BNE     10$             ;BR IF NO
126 004706  122710  000131                    CMPB    #'Y,(R0)        ;WAS IT A 'Y' RESPONSE ?
127 004712  001411                            BEQ     12$             ;BR IF YES
128 004714  122710  000116                    CMPB    #'N,(R0)        ;WAS IT A 'N' RESPONSE ?
129 004720  001410                            BEQ     13$             ;BR IF YES
130 004722  104401  060163            10$:    TYPE    ,BADENT         ;TYPE BAD ENTRY MESSAGE
131 004726  000753                            BR      9$              ;TRY AGAIN
132 004730  104401  057077            11$:    TYPE    ,NODFLT         ;TYPE 'NO DEFAULT'
133 004734  000750                            BR      9$              ;TRY AGAIN
134 004736  104401  064500            12$:    TYPE    ,HELPTX         ;TYPE HELP TEXT MESSAGE
135 004742                              13$:
```

```
                                        ;AUTO SIZE FOR RH70 CONTROLLER AND DETERMINE IF IT IS
                                        ;JUMPERED FOR 22 OR 32 REGISTERS

004742  005037  040646          SIZE70: CLR     RHEXT           ;CLEAR RPBAE OFFSET
004746  042737  174000  001234          BIC     #174000,$CPUOP  ;CLEAR CPU TYPE REGISTER
004754  013746  000004                  MOV     ERRVEC,(SP)     ;SAVE CONTENTS OF ERROR VECTOR
004760  012737  005032  000004          MOV     #2$,ERRVEC      ;SETUP 'TRAP' RETURN ADDRESS
004766  013700  001272                  MOV     $RPADR,R0       ;GET RPCS1 ADDRESS
004772  062700  000050                  ADD     #50,R0          ;GET REGISTER OFFSET FOR RH70
004776  012701  000012                  MOV     #10.,R1         ;GET NUMBER OF REGISTERS TO CHECK
005002  005720                          TST     (R0)+           ;TRAP IF NOT A VALID RPBAE
005004  005720                          TST     (R0)+           ;TRAP IF NOT A VALID RPCS3
005006  012737  000050  040646          MOV     #50,RHEXT       ;LOAD OFFSET FOR RPBAE (22 REGISTER RH)
005014  005720                  1$:     TST     (R0)+           ;TRAP IF NOT A VALID REGISTER
005016  005301                          DEC     R1              ;DONE WITH ALL 32 REGISTERS ?
005020  001375                          BNE     1$              ;BR IF NO
005022  012737  000074  040646          MOV     #74,RHEXT       ;LOAD OFFSET FOR RPBAE (32 REGISTER RH)
005030  000403                          BR      3$
005032  012716  005040          2$:     MOV     #3$,(SP)        ;SETUP RETURN ADDRESS
005036  000002                          RTI

005040  013700  001272          3$:     MOV     $RPADR,R0       ;GET RPCS1 REGISTER
005044  013701  040646                  MOV     RHEXT,R1        ;GET RPBAE REGISTER OFFSET
005050  001416                          BEQ     4$              ;BR IF NONE
005052  060001                          ADD     R0,R1           ;GET RPBAE REGISTER
005054  052710  001400                  BIS     #A17!A16,(R0)   ;SET EXTENDED ADDRESS BITS IN RPCS1
005060  022711  000003                  CMP     #3,(R1)         ;ARE THE EXTENDED BITS SET IN RPBAE ?
005064  001010                          BNE     4$              ;BR IF NO
005066  005011                          CLR     (R1)            ;CLEAR EXTENDED ADDRESS BITS IN RPBAE
005070  011046                          MOV     (R0),-(SP)      ;SAVE RPCS1 REG CONTENTS
005072  042726  176377                  BIC     #^C<A17!A16>,(SP)+      ;ARE THE EXTEND BITS CLEAR IN RPCS1 ?
005076  001003                          BNE     4$              ;BR IF NO
005100  052737  030000  001234          BIS     #BIT13!BIT12,$CPUOP     ;SET THE 11/70 CPU TYPE CODE
005106  012637  000004          4$:     MOV     (SP)+,ERRVEC    ;RESTORE CONTENTS OF ERROR VECTOR
```

```
                                        ;ROUTINE TO DETERMINE BUFFER MAX WORD COUNT
                                        ;
005112  005227  177777        SIZMEM: INC     @#1             ;FIRST TIME THRU HERE ?
005116  001005                        BNE     1$              ;BR IF NO
005120  004737  062344                JSR     PC,$SIZE        ;SEE HOW MUCH MEMORY ON SYSTEM
005124  013737  062474  001330        MOV     $LSTAD,LSTAD    ;SAVE THE LAST ADDRESS
005132  012737  000001  001654  1$:   MOV     #1,BUFTBL       ;LOAD NUMBER OF BUFFERS
005140  012737  064370  001656        MOV     #ENDPGM,BUFTBL+2        ;STARTING ADDRESS OF BUFFER
005146  013737  001330  001660        MOV     LSTAD,BUFTBL+4  ;LAST ADDR TO BUFFER ALLOCATION TABLE
005154  023727  001330  160000        CMP     LSTAD,#160000   ;OVER 28K ?
005162  101403                        BLOS    2$              ;NO
005164  012737  160000  001660        MOV     #160000,BUFTBL+4        ;XXDP MAX MEMORY 28K
005172  162737  064370  001660  2$:   SUB     #ENDPGM,BUFTBL+4        ;SUBTRACT PROGRAM SPACE
005200  000241                        CLC                     ;CLEAR THE 'C' BIT
005202  006037  001660                ROR     BUFTBL+4        ;CONVERT TO WORD COUNT
005206  105737  001150                TSTB    $AUTOB          ;RUNNING IN AUTO MODE ?
005212  001403                        BEQ     3$              ;BR IF NO
005214  162737  003000  001660        SUB     #1536.,BUFTBL+4 ;SUBTRACT 'XXDP' LOADER SIZE (1.5K WORDS)
005222  023737  001466  001660  3$:   CMP     WRDCNT,BUFTBL+4 ;IS MAX WORD COUNT TOO LARGE ?
005230  003406                        BLE     4$              ;BR IF NO
005232  013737  001660  001466        MOV     BUFTBL+4,WRDCNT ;USE MAX AVAIL MEMORY AS MAX WRD CNT
005240  013737  001466  060646        MOV     WRDCNT,PARLST+2 ;VALUE FOR THE PARAMETER TABLE
005246                          4$:
```

```
                                        ;SEE IF THE OPERATOR WANTS TO CHANGE ANY PARAMETERS
                                
 005246  005737  040100        LKPAR:  TST     PWRFLG          ;RETURNING FROM POWER FAIL ?
 005252  001154                        BNE     SETVEC          ;BRANCH IF YES
 005260  105737  001150                TSTB    @AUTOB          ;RUNNING IN AUTO MODE ?
 005264  001404                        BEQ     1$              ;BR IF NO
 005266  032777  000004  173660        BIT     #SW02,@SWR      ;DOES USER WANT MANUAL INTERVENTION ?
 005274  001467                        BEQ     8$              ;BR IF YES
                                
 005276  005037  001334        1$:     CLR     CFLAG           ;CLEAR CONTROL C FLAG
 005302  104401  060306                TYPE    .MESFE          ;TYPE 'DO YOU WANT TO WRITE ANYWHERE ON MEDIA (L) N ?'
 005306  104411                        RDLIN                   ;READ THE ENTRY
 005310  012600                        MOV     (SP)+,R0        ;SAVE ADDRESS OF RESPONSE
 005312  005737  001334                TST     CFLAG           ;WAS IT CONTROL C ?
 005316  001353                        BNE     LKPAR           ;BR IF YES
 005320  105710                        TSTB    (R0)            ;WAS RESPONSE A CARRIAGE RETURN ?
 005322  001454                        BEQ     8$              ;BR IF YES
 005324  105760  000001                TSTB    1(R0)           ;WAS IT TERMINATED WITH CARRIAGE RETURN ?
 005330  001006                        BNE     2$              ;BR IF NO
 005332  122710  000131                CMPB    @'Y,(R0)        ;WAS IT A 'Y' RESPONSE ?
 005336  001406                        BEQ     3$              ;BR IF YES
 005340  122710  000116                CMPB    @'N,(R0)        ;WAS IT A 'N' RESPONSE ?
 005344  001443                        JEQ     8$              ;BR IF YES
 005346  104401  060165        2$:     TYPE    .BADENT         ;TYPE BAD ENTRY MESSAGE
 005352  000735                        BR      LKPAR           ;TRY AGAIN
 005354  005737  001424        3$:     TST     RDONLY          ;PROGRAM RUNNING IN READ ONLY MODE ?
 005360  001032                        BNE     7$              ;BR IF YES
                                
 005362  005037  001334        4$:     CLR     CFLAG           ;CLEAR CONTROL C FLAG
 005366  104401  060367                TYPE    .OVRWRT         ;TYPE '   ! CUSTOMER DATA WILL BE OVERWRITTEN !
                                                                ;       -------- - -       - --    ---- -
                                                                ;CONTINUE (L)  ?'
 005372  104411                        RDLIN                   ;READ THE ENTRY
 005374  012600                        MOV     (SP)+,R0        ;SAVE ADDRESS OF RESPONSE
 005376  005737  001334                TST     CFLAG           ;WAS IT CONTROL C ?
 005402  001321                        BNE     LKPAR           ;BR IF YES
 005404  105710                        TSTB    (R0)            ;WAS RESPONSE A CARRIAGE RETURN ?
 005406  001414                        BEQ     6$              ;BR IF YES
 005410  105760  000001                TSTB    1(R0)           ;WAS IT TERMINATED WITH CARRIAGE RETURN ?
 005414  001006                        BNE     5$              ;BR IF NO
 005416  122710  000131                CMPB    @'Y,(R0)        ;WAS IT A 'Y  RESPONSE ?
 005422  001411                        BEQ     7$              ;BR IF YES
 005424  122710  000116                CMPB    @'N,(R0)        ;WAS IT A 'N' RESPONSE ?
 005430  001411                        BEQ     8$              ;BR IF YES
 005432  104401  060163        5$:     TYPE    .BADENT         ;TYPE BAD ENTRY MESSAGE
 005436  000751                        BR      4$              ;TRY AGAIN
 005440  104401  057077        6$:     TYPE    .NODFLT         ;TYPE 'NO DEFAULT'
 005444  000746                        BR      4$              ;TRY AGAIN
 005446  005237  001422        7$:     INC     TSTANY          ;ENABLE TEST ANYWHERE OPTION
 005452  000402                        BR      9$
 005454  104401  060531        8$:     TYPE    .FEONLY         ;TYPE '* TESTING WILL OCCUR ON FE CYLINDER ONLY *
                                
 005460  005737  001424        9$:     TST     RDONLY          ;IS PROGRAM LOCKED IN "READ ONLY" MODE ?
 005464  001402                        BEQ     13$             ;BR IF NO
 005466  104401  060570                TYPE    .MREAD          ;TYPE '* PROGRAM IS LOCKED IN 'READ ONLY' MODE *'
                                
 005472  005037  001334        13$:    CLR     CFLAG           ;CLEAR CONTROL C FLAG
```

GET VALUE FOR SOFTWARE SWITCH REGISTER

```
 88 005476  105737  001150            TSTB    $AUTOB          ;RUNNING IN AUTO MODE ?
 89 005502  001040                    BNE     SETVEC          ;BR IF YES
 90 005504  104401  060746            TYPE    ,ASKPAR         ;TYPE 'CHANGE PARAMETERS ?
 91 005510  104411                    RDLIN                   ;READ THE ENTRY
 92 005512  012600                    MOV     (SP)+,R0        ;SAVE ADDRESS OF RESPONSE
 93 005514  005737  001334            TST     CFLAG           ;WAS (^C) TYPED?
 94 005520  001364                    BNE     13$             ;BR IF YES
 95 005522  105710                    TSTB    (R0)            ;WAS RESPONSE A CARRIAGE RETURN (DEFAULT 'N')?
 96 005524  001427                    BEQ     SETVEC          ;BR IF YES
 97 005526  105760  000001            TSTB    1(R0)           ;WAS IT TERMINATED WITH CARRIAGE RETURN ?
 98 005532  001006                    BNE     14$             ;BR IF NO
 99 005534  122710  000131            CMPB    #'Y,(R0)        ;WAS IT A 'Y' RESPONSE ?
100 005540  001406                    BEQ     ENTPR           ;BR IF YES
101 005542  122710  000116            CMPB    #'N,(R0)        ;WAS IT A 'N' RESPONSE ?
102 005546  001416                    BEQ     SETVEC          ;BR IF YES
103 005550  104401  060163    14$:    TYPE    ,BADENT         ;TYPE BAD ENTRY MESSAGE
104 005554  000746                    BR      13$             ;TRY AGAIN
105 005556            15$:
106
107 005556  012703  060644    ENTPR:  MOV     #PARLST,R3      ;PARAMETER TABLE ADDRESS
108 005562  004737  031170            JSR     PC,PARENT       ;GET THE PARAMETER ENTRY
109 005566  023727  001466  000006    CMP     WRDCNT,#6       ;IS THE 'WRDCNT' VALUE OK ?
110 005574  103003                    BHIS    SETVEC          ;BR IF IT IS
111 005576  012737  000006  001466    MOV     #6,WRDCNT       ;SET 'WRDCNT' TO THE MINIMUM VALUE
```

```
                                        ;DISPLAY DRIVE STATUS AND SET UP THE OTHER SYSTEM DEVICES THAT
                                        ;THE PROGRAM WILL USE. PROGRAM RETURN HERE ON POWER FAIL
                                        ;
 4 005604  004737  024574       SETVEC: JSR     PC,CKCLK        ;START THE CLOCK
 5 005610  004737  040652               JSR     PC,RPINIT       ;INITIALIZE THE RP07 DRIVER
 9 005614  012737  177777  040602       MOV     #-1,SAVEFG      ;SET THE SAVE REGISTERS FLAG
10 005622  005227  177777               INC     #-1             ;FIRST TIME THRU ?
11 005626  001407                       BEQ     1$              ;BR IF YES
12 005630  005737  040100               TST     PWRFLG          ;RETURNING FROM POWER FAIL ?
13 005634  001004                       BNE     1$              ;BRANCH IF YES
14 005636  032777  000004  173310       BIT     #SW02,@SWR      ;TYPEOUT THE DRIVE STATUS TABLE ?
15 005644  001113                       BNE     12$             ;BR IF NOT
16 005646  005004               1$:     CLR     R4              ;DRIVE TABLE POINTER
17 005650  104401  057113               TYPE    ,DRSTAT         ;TYPE 'DRIVE STATUS'
18 005654  104401  001203       2$:     TYPE    ,$CRLF          ;CR-LF
19 005660  010446                       MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
                                                                 ;;TYPE DRIVE NUMBER
   005662  104403                       TYPOS                   ;;GO TYPE--OCTAL ASCII
   005664     002                       .BYTE   2               ;;TYPE 2 DIGIT(S)
   005665     000                       .BYTE   0               ;;SUPPRESS LEADING ZEROS
20 005666  104401  056606               TYPE    ,BLNKS4         ;TYPE 4 BLANKS
21 005672  105764  040534               TSTB    DRVSTA(R4)      ;CHECK DRIVE'S STATUS
22 005676  100416                       BMI     5$              ;BR IF UNSAFE
23 005700  001020                       BNE     6$              ;BR IF ONLINE
24
25 005702  105764  040544               TSTB    DRVTYP(R4)      ;SEE IF OFFLINE OR NONEXISTENT
26 005706  001404                       BEQ     3$              ;BR IF NONEXISTENT
27 005710  100006                       BPL     4$              ;BR IF OFFLINE
28 005712  104401  056760               TYPE    ,NOTRP          ;DRIVE NOT AN RP07
29 005716  000460                       BR      11$             ;CHECK NEXT DRIVE
30
31 005720  104401  056775       3$:     TYPE    ,NOTPRS         ;DRIVE NOT PRESENT
32 005724  000455                       BR      11$             ;CHECK NEXT DRIVE
33
34 005726  104401  056667       4$:     TYPE    ,UNTOFF         ;DRIVE OFFLINE
35 005732  000416                       BR      8$              ;PRINT DRIVE TYPE
36
37 005734  104401  057031       5$:     TYPE    ,NOTSAF         ;DRIVE UNSAFE
38 005740  000413                       BR      8$              ;PRINT DRIVE TYPE
39
40 005742  005737  001430       6$:     TST     XXDP            ;LOADED FROM THIS DEVICE ?
41 005746  001406                       BEQ     7$              ;BR IF NO
42 005750  123704  001430               CMPB    XXDP,R4         ;LOADED FROM THIS DRIVE ?
43 005754  001003                       BNE     7$              ;BR IF NO
44 005756  104401  057041               TYPE    ,LODEV          ;TYPE 'LOAD DEVICE'
45 005762  000436                       BR      11$
46 005764  104401  056700       7$:     TYPE    ,UNTON          ;DRIVE ONLINE
47 005770  104401  056610       8$:     TYPE    ,BLNKS2         ;TYPE 2 BLANKS
48 005774  005000                       CLR     R0
49 005776  116400  040544               MOVB    DRVTYP(R4),R0   ;GET DRIVE TYPE
50 006002  012737  057155  006026       MOV     #9$,RP07,10$    ;ASSUME ADDRESS OF RP07 MESSAGE
51 006010  122700  000005               CMPB    #5,R0           ;IS DEVICE AN RP07 ?
52 006014  001403                       BEQ     9$              ;TYPE IT IF YES
53 006016  012737  057162  006026       MOV     #9$,RP07P,10$   ;ASSUME ADDRESS OF RP07+ MESSAGE
54
55 006024  104401               9$:     TYPE                    ;TYPE THE DRIVE TYPE MESSAGE
56 006026  000000               10$:    .WORD   0               ;MESSAGE ADDRESS HERE
```

GET VALUE FOR SOFTWARE SWITCH REGISTER

```
                57
                58 006050  006304                       ASL     R4                      ;CHANGE TO WORD INDEX
                59 006052  016402  002056                MOV     BLKADR(R4),R2           ;GET BASE ADDRESS
                60 006056  006204                       ASR     R4                      ;CHANGE TO BYTE INDEX
                61 006060  004737  065342                JSR     PC,SVREGS               ;SAVE ALL REGISTERS
                62 006064  032762  000004  000200       BIT     #ILV,HRPOS(R2)          ;INTERLEAVE SECTOR SET ?
                63 006072  001002                       BNE     11$                     ;BR IF YES
                64 006074  104401  057056                TYPE    ,NINLEV                 ;TYPE NON-INTERLEAVED MESSAGE
                65
                66 006100  005204                 11$:  INC     R4                      ;INCREMENT DRIVE NUMBER/TABLE POINTER
                67 006102  020427  000010                CMP     R4,#8.                  ;FINISHED ?
                68 006106  001272                       BNE     2$                      ;BR IF NOT
                69 006110  104401  001203                TYPE    ,$CRLF                  ;CR-LF
                70 006114                          12$:
```

```
                        ;INITIALIZE PROGRAM PARAMETERS FOR STARTUP
                        ;
006076  004737  035510          STA:    JSR     PC,$TWINT       ;INITIALIZE THE KEYBOARD INTERRUPT HANDLER
006100  012737  142200  001446          MOV     #142200,@#ENDCDN ;INITIALIZE NORMAL XFER COUNT(LSB)
006106  012737  007540  001450          MOV     #7540,@#ENDCDN+2 ;(MSB)
006114  032777  001000  175052          BIT     #SW9,@#SWR      ;DO YOU WANT SHORT RUN TIME (SW9=1) ?
006122  001406                          BEQ     1$              ;BR IF NO
006124  012737  030440  001446          MOV     #30440,@#ENDCDN ;INITIALIZE (1/4 OF NORMAL) XFER COUNT(LSB)
006132  012737  001730  001450          MOV     #1730,@#ENDCDN+2 ;(MSB)
                        ;
006140  105737  001226          1$:     TSTB    @#KBW           ;APT SCRIPT MODE ?
006144  001406                          BEQ     2$              ;NO
006146  012737  006110  001446          MOV     #6110,@#ENDCDN  ;INITIALIZE (1/16 OF NORMAL) XFER COUNT(LSB)
006154  012737  000366  001450          MOV     #366,@#ENDCDN+2 ;(MSB)
                        ;
006162  105737  001150          2$:     TSTB    @#AUTOB         ;RUNNING IN AUTO MODE ?
006166  001003                          BNE     3$              ;BR IF YES
006170  005737  001332                  TST     @#CHGADR        ;START AT 200 ?
006174  003456                          BLE     8$              ;NO
                        ;
006176  005001          3$:     CLR     R1                      ;DRIVE 0
006200  005002                  CLR     R2                      ;AVAIL TABLE INDEX
006202  005003                  CLR     R3                      ;DRIVE# * 2
006204  005737  001430          4$:     TST     @#XIDP          ;LOADED FROM THIS DEVICE ?
006210  001405                          BEQ     5$              ;BR IF NO
006212  123701  001430                  CMPB    @#XIDP,R1       ;LOADED FROM THIS DRIVE ?
006216  001433                          BEQ     7$              ;BR IF YES
006220  105761  000534          5$:     TSTB    DRVSTA(R1)      ;DRIVE ON LINE ?
006224  003430                          BLE     7$              ;NO
006226  016500  002056                  MOV     @LUNADR(R5),R0  ;LOAD DPB ADDRESS
006232  004737  030164                  JSR     PC,CLRDPB       ;CLEAR DPB BLOCK
006236  004737  031070                  JSR     PC,GETID        ;GET DRIVE SERIAL NUMBER
006242  032760  000004  000200          BIT     #ILV,XDPOS(R0)  ;INTERLEAVE SECTOR SET ?
006250  001402                          BEQ     6$              ;BR IF NO
006252  010062  001566                  MOV     R0,NEWUNT(R2)   ;LOAD DPB ADDRESS TO AVAIL QUEUE
006256  004737  030402          6$:     JSR     PC,DRVPRM       ;SETUP DRIVE PARAMETER LIMITS
006262  005737  040100                  TST     PWRFLG          ;RETURNING FROM POWER FAIL ?
006266  001007                          BNE     7$              ;BRANCH IF YES
006270  005060  000132                  CLR     @FIRST(R0)      ;RESET @FIRST FLAG FOR FIRST 200 START
006274  112760  177776  000026          MOVB    #-2,@PACK(R0)   ;SETUP COMMAND 'WT' (WRITE DATA AND TEST)
006302  004737  020330                  JSR     PC,SEQPAR       ;SETUP SEQUENTIAL PARAMETERS FOR WRITE
                        ;
006306  022322          7$:     CMP     (R3)+,(R2)+     ;INCREMENT INDEX
006310  005201                  INC     R1              ;NEXT DRIVE
006312  020127  000007          CMP     R1,#7           ;ALL DRIVES ASSIGNED ?
006316  003752                  BLE     4$              ;NO
006320  005037  001332          8$:     CLR     @#CHGADR        ;CLEAR START FLAG
006324  000403                  BR      9$
                        ;
006326  012737  000001  001334  8$:     MOV     #1,@#CFLAG      ;DUMMY 'CONTROL C' FLAG
006334  005037  040100          9$:     CLR     @#PWRFLG        ;CLEAR POWER FAIL FLAG
```

```
                                        .SBTTL  MAIN PROGRAM
                                ;
                                ;
000500  005737* 001530          MAIN:   TST     CFLAG           ;WAS (^C) TYPED?
000504  001407                          BEQ     50              ;BR IF NO
000506  005737  001520          10:     TST     ORDERS          ;ANY DRIVES IN ORDER QUE ?
000512  001402                          BEQ     20              ;BR IF NO, EXIT
000514  000137  007256                  JMP     IDLE            ;LET ALL DRIVES FINISH ORDER
000520  004737  026530          20:     JSR     PC,$KB          ;SERVICE THE KEYBOARD
000524                          30:
                                ;
                                ;CHECK FOR DRIVES TO BE DROPPED
                                ;
000524  012701  000010          MAIN0A: MOV     #8.,R1          ;DRIVE COUNTER
000530  012705  001544                  MOV     #DROPS,R5       ;ADDRESS OF  DROPPED DRIVE  TABLE
000534  005715                  40:     TST     (R5)            ;SEE IF ENTRY AT PRESENT POSITION
000536  001406                          BEQ     50              ;BR IF THERE IS ONE
000540  005725                  45:     TST     (R5)+           ;INCREMENT TO NEXT TABLE POSITION
000542  005301                          DEC     R1              ;DECREMENT DRIVE COUNTER
000544  003373                          BGT     40              ;BR IF MORE TO CHECK
000546  000137  000737                  JMP     MAIN1           ;GO CHECK FOR NEW ASSIGNED DRIVES
                                ;
000552  012701  001610          50:     MOV     #AVAIL,R1       ;ADDRESS OF 'AVAILABLE DRIVES' TABLE
000556  005711                  60:     TST     (R1)            ;IF AT END OF 'AVAIL' TABLE ?
000560  001406                          BEQ     70              ;BR IF YES
000562  021115                          CMP     (R1),(R5)       ;IS DRIVE IN 'AVAIL' THE TABLE ?
000564  001412                          BEQ     75              ;BR IF YES
000566  005721                          TST     (R1)+           ;NO, INCREMENT 'AVAIL' TABLE ADDRESS
000570  000772                          BR      60              ;AND CONTINUE LOOKING
                                ;
000572  012701  001632          50:     MOV     #WAIT,R1        ;ADDRESS OF THE 'WAIT' BUFFER TABLE
000576  005711                  60:     TST     (R1)            ;AT THE END OF 'WAIT' TABLE ?
000600  001760                          BEQ     20              ;BR IF YES
000602  021115                          CMP     (R1),(R5)       ;IS DRIVE IN THE 'WAIT' TABLE ?
000604  001402                          BEQ     70              ;BR IF YES
000606  005721                          TST     (R1)+           ;NO, INCREMENT 'WAIT' TABLE ADDRESS
000610  000772                          BR      60              ;AND CONTINUE LOOKING
                                ;
000612  011100                  70:     MOV     (R1),R0         ;PUT THE DRIVE'S BLOCK ADDRESS IN R0
000614  104401  001703                  TYPE    ,SCRLF          ;CR LF
000620  004737  025426                  JSR     PC,$TIME        ;TYPE ELAPSED TIME
000624  104401  057401                  TYPE    ,DASHS          ;TYPE '-----'
000630  104401  057437                  TYPE    ,MSGDRP         ;TYPE 'DROPPED, '
000634  004737  025122                  JSR     PC,ONESUM       ;TYPE ONE DRIVE SUMMARY
000640  005015                          CLR     (R5)            ;CLEAR THE 'DROP DRIVE' TABLE ENTRY
000642  004737  022260                  JSR     PC,COMPRES      ;COMPRESS THE RESPECTIVE TABLE
000646  000735                          BR      20              ;SEE IF ANY MORE DRIVES
```

```
                                   ;LOOP FOR DRIVES TO BE ASSIGNED

[illegible] 012700 000010          [illegible]:  MOV   [illegible]         ;DRIVE COUNT
[illegible] [illegible]                         CLR   R1                  ;ASSIGN LIST INDEX
[illegible] [illegible]                         CLR   R2                  ; AVAIL  INDEX
[illegible] [illegible]                         CLR   R5                  ;NEW DRIVE INDEX
[illegible] [illegible] [illegible]     10$:  TST   [illegible](R5)     ;NEW DRIVE IN THIS POSITION
[illegible] [illegible]                         BNE   3$                  ;GO IF THERE IS
[illegible] [illegible]                 20$:  TST   (R5)+               ;INCREMENT R5
[illegible] [illegible]                         INC   R1                  ;INCREMENT ASSIGN INDEX
[illegible] [illegible]                         DEC   R0                  ;INCREMENT DRIVE COUNT
[illegible] [illegible]                         BNE   1$                  ;GO IF MORE DRIVES
[illegible] [illegible]                         BR    [illegible]         ;START OPERATIONS FOR THE AVAILABLE DRIVES

[illegible] 104401 [illegible]         3$:   TYPE  ,[illegible]        ;CR LF
[illegible] 104401 [illegible]                TYPE  ,[illegible]        ; DRIVE:
[illegible] 010146                              MOV   R1,-(SP)            ;;SAVE R1 FOR TYPEOUT
                                                                           ;;TYPE DRIVE NUMBER
[illegible] 104405                              TYPOS                     ;;GO TYPE--OCTAL ASCII
[illegible]    [illegible]                      .BYTE 2                   ;;TYPE 2 DIGIT(S)
[illegible]    [illegible]                      .BYTE 0                   ;;SUPPRESS LEADING ZEROS
[illegible] 104401 [illegible]                  TYPE  ,[illegible]        ;TYPE :
[illegible] 104401 [illegible]                  TYPE  ,[illegible]        ;TYPE 1 BLANK
[illegible] [illegible] [illegible]             MOV   [illegible](R5),R0  ;PUT ADDRESS OF [illegible] IN R0
[illegible] [illegible] [illegible]             JSR   PC,[illegible]      ;TYPE DRIVE SERIAL NUMBER
[illegible] 104401 [illegible]                  TYPE  ,[illegible]        ; STARTED
[illegible] [illegible] [illegible]     40$:  TST   AVAIL(R2)           ;AT END OF AVAILABLE TABLE
[illegible] [illegible]                         BEQ   5$                  ;GO IF YES
[illegible] [illegible]                         TST   (R2)+               ;INCREMENT AVAILABLE TABLE INDEX
[illegible] [illegible]                         BR    4$                  ;CONTINUE LOOKING FOR END OF TABLE
[illegible] [illegible] [illegible] [illegible] 5$:  MOV [illegible](R5),AVAIL(R2)  ;MOVE ADDR OF DRIVE INTO AVAIL LST
[illegible] [illegible] [illegible]             CLR   [illegible](R5)     ;TAKE DRIVE OUT OF NEW DRIVE TABLE
[illegible] [illegible] [illegible] [illegible] BIS [illegible](R1),[illegible]  ;SET DRIVE ASSIGNED INDICATOR
[illegible] [illegible] [illegible]             CLR   [illegible]         ;CLEAR AUTO ASSIGN
[illegible] [illegible]                         TST   (R2)+               ;INCREMENT AVAILABLE TABLE POINTER
[illegible] [illegible]                         BR    20$                 ;LOOP FOR MORE DRIVES
```

[illegible]

```
                            ; [illegible] BUFFER SPACE, AND START ORDERS FOR DRIVES [illegible]
                            ; [illegible]

[illegible]                                         ;START FROM THE FIRST LOCATION
[illegible]                                         ;ANY [illegible] ACTIVE ?
[illegible]                                         ;[illegible] IF [illegible]
[illegible]                                         ;[illegible] AUTO MODE ?
[illegible]                                         ;[illegible] IF [illegible]
[illegible]
[illegible]                                         ;[illegible] VALUE
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]
007[illegible] 00[illegible]37 0[illegible]  10:  JMP  [illegible]   ;GIVE CONTROL TO [illegible]

00[illegible] 00[illegible] 001[illegible]  20:  TST  WAIT(R2)   ;ANY DRIVES WAITING FOR THE BUFFER ?
00[illegible] 00[illegible]              BEQ  50          ;[illegible] IF NO
00[illegible] 00[illegible] 001[illegible]       MOV  WAIT(R2),R0 ;[illegible] THE [illegible] ADDRESS
00[illegible] 00[illegible]              CLR  [illegible]  ;[illegible] THE START [illegible]
00[illegible] 00[illegible] 01[illegible]       JSR  PC,[illegible] ;[illegible] THE BUFFER [illegible]
00[illegible] 00[illegible] 00[illegible]       MOV  [illegible] ;IF A BUFFER IS [illegible] AVAILABLE
00[illegible] 00[illegible]              BEQ  40          ;BRANCH IF NO BUFFER AVAILABLE
00[illegible] 00[illegible] 00[illegible]       CLR  [illegible](R0) ;[illegible] PARAMETERS [illegible] FLAG
00[illegible] 00[illegible] 00[illegible]       CLR  [illegible](R0) ;CLEAR THE [illegible] FLAG

00[illegible] 00[illegible]37 00[illegible]       JSR  PC,[illegible] ;FILL THE BUFFER
00[illegible] 00[illegible] [illegible]       JSR  PC,[illegible] ;[illegible] AND [illegible]
00[illegible] 00[illegible] 00[illegible]       MOV  [illegible],R5 ;[illegible] QUE INTO ORDER QUE
00[illegible] 00[illegible]         30:  TST  (R5)+       ;QUE AVAILABLE ?
00[illegible] 00[illegible]              BNE  30          ;[illegible] IF NO
00[illegible] 00[illegible]              MOV  R0,-(R5)    ;[illegible] THE [illegible] ADDRESS INTO THE ORDER QUE
00[illegible] 00[illegible] 001[illegible]       MOV  WAIT(R2),R1 ;[illegible] THE DRIVE FROM THE WAIT QUE
00[illegible] 00[illegible]              ADD  R2,R1       ;[illegible] THE QUE POSITION
00[illegible] 00[illegible] 00[illegible]       JSR  PC,[illegible] ;[illegible] THE QUE
00[illegible] 00[illegible]              BR   20          ;BRANCH IF DONE
00[illegible] 00[illegible]         40:  TST  (R2)+       ;[illegible] THE NEXT QUE
007032 00[illegible]              BR   20          ;[illegible] BACK

007034 005737 001530  50:  TST  ORDERS      ;ANY OUTSTANDING ORDERS ?
007040 001075              BNE  IDLE        ;[illegible] IF YES

007042 005002              CLR  R2          ;CLEAR DRIVE TABLE POINTER
00[illegible] 00[illegible] 001610  60:  TST  AVAIL(R2)   ;ANY DRIVES WAITING FOR PARAMETERS
00[illegible] 00[illegible]              BNE  70          ;BRANCH IF ANY
00[illegible] 00[illegible]37 007256       JMP  IDLE        ;BRANCH IF NONE
00[illegible] 0[illegible] 00[illegible]610  70:  MOV  AVAIL(R2),R0 ;[illegible] BLOCK ADDR IN R0
00[illegible] 00[illegible] 00[illegible]130       TST  [illegible](R0) ;PARAMETERS BEEN SELECTED ?
00[illegible] 00[illegible]10              BNE  90          ;[illegible] IF THEY HAVE
00[illegible] 10[illegible] 000026       TSTB [illegible](R0) ;[illegible] COMMAND FOR THIS DRIVE ?
00[illegible]70 00[illegible]03              BEQ  80          ;[illegible] IF YES
00[illegible]7076 00[illegible]37 020530       JSR  PC,[illegible] ;GET SEQUENTIAL PARAMETERS FOR READ OR WRITE
007102 000[illegible]              BR   100         ;GET THE BUFFER

007104 004737 020672  80:  JSR  PC,[illegible] ;GO GENERATE THE PARAMETERS
```

```
[illegible]                                                                                    [illegible]
[illegible]

[illegible]  [illegible]  [illegible]          [illegible]  [illegible]  [illegible]  [illegible]
[illegible]  [illegible]                       100.         [illegible]  [illegible]  [illegible]
[illegible]  [illegible]  [illegible]                       [illegible]  [illegible]  [illegible]
[illegible]  [illegible]  [illegible]                       [illegible]  [illegible]  [illegible]
[illegible]  [illegible]  [illegible]                       [illegible]  [illegible]  [illegible]
[illegible]  [illegible]  [illegible]                       [illegible]  [illegible]  [illegible]
[illegible]  [illegible]  [illegible]                       [illegible]  [illegible]  [illegible]

[illegible]  [illegible]  [illegible]                       [illegible]  [illegible]  [illegible]
[illegible]  [illegible]  [illegible]                       [illegible]  [illegible]  [illegible]
[illegible]  [illegible]                       110.         [illegible]  [illegible]  [illegible]
[illegible]  [illegible]                                    [illegible]  [illegible]  [illegible]
[illegible]  [illegible]                                    [illegible]  [illegible]  [illegible]
[illegible]  [illegible]                                    [illegible]  [illegible]  [illegible]

[illegible]  [illegible]  [illegible]          120.         [illegible]  [illegible]  [illegible]
[illegible]  [illegible]  [illegible]  [illegible]          [illegible]  [illegible]  [illegible]
[illegible]  [illegible]                                    [illegible]  [illegible]  [illegible]
[illegible]  [illegible]  [illegible]          130.         [illegible]  [illegible]  [illegible]
[illegible]  [illegible]                                    [illegible]  [illegible]  [illegible]
[illegible]  [illegible]                                    [illegible]  [illegible]  [illegible]
[illegible]  [illegible]                                    [illegible]  [illegible]  [illegible]

[illegible]  [illegible]                       140.         [illegible]  [illegible]  [illegible]
[illegible]  [illegible]                                    [illegible]  [illegible]  [illegible]
[illegible]  [illegible]  [illegible]          150.         [illegible]  [illegible]  [illegible]
[illegible]  [illegible]                                    [illegible]  [illegible]  [illegible]
[illegible]  [illegible]  [illegible]                       [illegible]  [illegible]  [illegible]
[illegible]  [illegible]                                    [illegible]  [illegible]  [illegible]
```

```
                                ;WAIT FOR A COMMAND TO FINISH

007266  012701  001520          TIME:   MOV     #ORDERS,R1      ;ADDRESS OF THE ORDER QUE IN R1
007272  012105                  10$:    MOV     (R1)+,R5        ;GET QUEUE ADDRESS INTO R5
007274  001447                          BEQ     7$              ;BR IF END OF QUE
007276  005760  000016          20$:    TST     STATUS(R5)      ;SEE IF DRIVE FINISHED
007302  100375                          BPL     20$             ;BR IF DRIVE NOT FINISHED
007304  005741                          TST     -(R1)           ;BACKUP THE QUE POINTER
007306  010146                          MOV     R1,-(SP)        ;SAVE THE QUE ADDRESS
007310  004737  017270                  JSR     PC,SAVSTA       ;SEPARATE STATISTICS FOR DRIVE IN R5
007314  004737  007472                  JSR     PC,PROCES       ;PROCESS END OF COMMAND
007320  005037  001336                  CLR     BADSEC          ;CLEAR THE BAD TRK/SEC ERROR INDICATOR
007324  004737  031452                  JSR     PC,ABORTS       ;SEE IF ANY DRIVES HAVE TOO MANY ERRORS
                                        ;NOW SEE IF WE ARE AT THE END OF A PASS
007330  122760  177777  000026          CMPB    #-1,EPOCH(R0)   ;ANY COMMAND FOR THIS DRIVE ?
007336  001434                          BEQ     5$              ;BR IF YES
007340  122760  000001  000026          CMPB    #1,EPOCH(R0)    ;ANY COMMAND FOR THIS DRIVE ?
007346  001430                          BEQ     5$              ;BR IF YES
007350  005737  001500                  TST     ENDING          ;WILL THE EOP BE DETERMINED BY # OF SEEKS ?
007354  001412                          BEQ     3$              ;BR IF YES
007356  026037  000040  001450          CMP     @#OPAS+2(R0),ENDCOM+2   ;IS IT THE EOP BY DATA READ ?
007364  101017                          BHI     4$              ;BR IF YES
007366  103420                          BLO     5$              ;BR IF NO
007370  026037  000036  001446          CMP     @#OPAS(R0),ENDCOM       ;IS IT THE EOP BY DATA READ ?
007376  103414                          BLO     5$              ;BR IF NO
007400  000411                          BR      4$              ;THIS IS THE END OF PASS

007402  026037  000050  001454  3$:     CMP     @#SEEKS+2(R0),ENDSEK+2  ;IS IT THE EOP BY SEEKS ?
007410  101005                          BHI     4$              ;BR IF YES
007412  103406                          BLO     5$              ;BR IF NO
007414  026037  000046  001452          CMP     @#SEEKS(R0),ENDSEK      ;IS IT THE EOP BY SEEKS ?
007422  103402                          BLO     5$              ;BR IF NO
007424  004737  031500          4$:     JSR     PC,@#EOP        ;THIS IS THE END OF PASS

007430  012601                  5$:     MOV     (SP)+,R1        ;RESTORE THE ORDER TABLE INDEX
007432  012705  001610                  MOV     #AVAIL,R5       ;ADDRESS OF THE 'AVAIL' TABLE
007436  005725                  6$:     TST     (R5)+           ;IS IT THE END OF THE 'AVAIL' TABLE ?
007440  001376                          BNE     6$              ;BR IF NO
007442  011145                          MOV     (R1),-(R5)      ;MOVE THE DPB INTO THE 'AVAIL' TABLE
007444  004737  022260                  JSR     PC,CMPRES       ;COMPRESS THE ORDER QUE
007450  004737  017726                  JSR     PC,RELBUF       ;RESTORE BUFFER

007454  032777  000004  171524  7$:     BIT     #SW02,@#SWR     ;TYPE PERFORMANCE SUMMARY
007462  001016                          BNE     8$              ;BR IF NOT
007464  005737  001314                  TST     STATIN          ;TIME TO TYPE THE PERFORMANCE SUMMARY ?
007470  001413                          BEQ     8$              ;BR IF NOT
007472  005037  001314                  CLR     STATIN          ;CLEAR THE INDICATOR
007476  005737  001542                  TST     ASMLST          ;ANY DRIVES ASSIGNED ?
007502  001406                          BEQ     8$              ;BR IF NO
007504  104401  057167                  TYPE    ,MSGREP         ;TYPE '\/\/\/\/\/-PERFORMANCE REPORT-\/\/\/\/\/'
007510  004737  025020                  JSR     PC,STATPR       ;TYPE THE SUMMARY
007514  104401  057274                  TYPE    ,TRMREP         ;TYPE '/\/\/\/\/\...ETC'

007520  000137  006340          8$:     JMP     MAIN            ;CONTINUE THE LOOP
```

```
                                        ;PROCESS THE COMMAND TERMINATION
                                        ;
007472  111037  001320          PROCES: MOVB    (R0),DRVNO      ;DRIVE NUMBER FOR ANY ERROR MESSAGES
007476  005760  000016                  TST     $TATUS(R0)      ;SEE IF DRIVER SIGNALED AN ERROR
007502  100447                          BMI     ERPROC          ;BR IF ERROR
007504  032760  100000  000166          BIT     @BIT15,$RPCS1(R0)       ;SEE IF 'SC' SET
007512  001410                          BEQ     1$              ;BR IF NOT SET
007514  032760  040000  000166          BIT     @BIT14,$RPCS1(R0)       ;SEE IF 'TRE' SET
007522  001037                          BNE     ERPROC          ;BR IF SET
007524  032760  040000  000200          BIT     @BIT14,$RPDS(R0)        ;SEE IF 'ERR' SET
007532  001033                          BNE     ERPROC          ;BR IF SET
                                        ;
                                        ;NO ERRORS DETECTED IN REGISTERS, DO SOME CHECKING ANYWAY
                                        ;
007534  004737  014010          1$:     JSR     PC,CKERR        ;CHECK ERROR BITS
007540  004737  014110                  JSR     PC,CKBUS        ;CHECK BUS ADDRESS & WORD COUNT
007544  032777  000002  171402          BIT     @SW01,@SWR      ;INHIBIT DATA COMPARE (SW01=1) ?
007552  001022                          BNE     3$              ;BR IF YES
007554  005737  001462                  TST     CMPTIM          ;IS COMPARE DATA ALWAYS ON ?
007560  001415                          BEQ     2$              ;BR IF YES
007562  023737  001464  001462          CMP     CMPTIM+2,CMPTIM ;TIME TO COMPARE DATA ?
007570  002413                          BLT     3$              ;BR IF NO
007572  013746  001462                  MOV     CMPTIM,-(SP)    ;GET CURRENT INTERVAL TIME AND
007576  062716  000001                  ADD     #1,(SP)         ;ADD ONE MINUTE
007602  023726  001464                  CMP     CMPTIM+2,(SP)+  ;STILL COMPARING DATA ?
007606  002402                          BLT     2$              ;BR IF YES
007610  005037  001464                  CLR     CMPTIM+2        ;CLEAR INTERVAL TIMER FOR NEXT COMPARE
007614  004737  014174          2$:     JSR     PC,CMPAR        ;NO 'DCK' ERROR, BUT COMPARE DATA ANYWAY
007620  000207                  3$:     RTS     PC              ;RETURN
                                        ;
                                        ;COMMAND TERMINATED WITH AN ERROR - PROCESS THE ERROR
                                        ;
007622  032760  000200  000016  ERPROC: BIT     @BIT07,$TATUS(R0)       ;DONE BIT SET ?
007630  001402                          BEQ     ERPRC1          ;BR IF NO, ORDER DIDN'T COMPLETE NORMALLY
007632  000137  010206                  JMP     DONE            ;PROCESS ERROR WITH 'DONE' BIT SET
                                        ;
                                        ;PROCESS ORDER COMPLETION WITH 'ERROR' & 'DONE' NOT SET
                                        ;
007636                          ERPRC1:
007636  032760  010000  000016          BIT     @BIT12,$TATUS(R0)       ;SEE IF DRIVE WAS UNSAFE
007644  001025                          BNE     PUNSAF          ;BR IF YES
007646  032760  004000  000016          BIT     @BIT11,$TATUS(R0)       ;PARITY ERROR OCCURRED
007654  001041                          BNE     UCPAR           ;BR IF IT DID
007656  032760  002000  000016          BIT     @BIT10,$TATUS(R0)       ;FATAL PARITY ERROR?
007664  001044                          BNE     FALPAR          ;BR IF THERE IS ONE
007666  032760  001000  000016          BIT     @BIT09,$TATUS(R0)       ;TIMEOUT?
007674  001066                          BNE     SWTIM           ;BR IF YES
007676  032760  040002  000016          BIT     @BIT14!BIT01,$TATUS(R0) ;DRIVE WENT OFFLINE ?
007704  001101                          BNE     OFLIN           ;BR IF IT DID
007706  032760  000004  000016          BIT     @BIT2,$TATUS(R0)        ;PORT REQUEST TIME OUT ?
007714  001115                          BNE     PRTIM           ;BR IF IT DID
007716  000207                          RTS     PC              ;ERROR. RETURN
```

```
                                        ;DRIVE IS PERSISTENTLY  UNSAFE

007720  004737  022274        PUNSAF: JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
007724  104414  051435                DISPLY  .EM12           ;PERSISTENT DEVICE UNSAFE MESSAGE
007730  004737  022342                JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
007734  004737  022776                JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
007740  004737  023446                JSR     PC,LINE4        ;PRINT LINE 4 OF THE ERROR MESSAGE
007744  004737  026202                JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
007750  004737  024136                JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
007754  000137  031362                JMP     DROP            ;DROP THE DRIVE

                                        ;UNCORRECTABLE MASSBUS PARITY ERROR OCCURRED

007760  104401  001203        UCPAR:  TYPE    .$CRLF          ;CR-LF
007764  104401  001203                TYPE    .$CRLF          ;CR-LF
007770  104401  051337                TYPE    .EM10           ; UNCORRECTABLE PARITY ERROR' MESSAGE
007774  000406                        BR      FALPR1          ;FINISH PROCESSING THE ERROR

                                        ;'FATAL' MASSBUS PARITY ERROR OCCURRED

007776  104401  001203        FALPAR: TYPE    .$CRLF          ;CR-LF
010002  104401  001203                TYPE    .$CRLF          ;CR-LF
010006  104401  051402                TYPE    .EM11           ;'FATAL PARITY ERROR' MESSAGE

010012  104401  057676        FALPR1: TYPE    .MSGON          ;TYPE 'ON'
010016  104401  056661                TYPE    .DRVMSG         ;TYPE  DRIVE'
010022  013746  001320                MOV     DRVNO,-(SP)     ;;SAVE DRVNO FOR TYPEOUT
                                                              ;;TYPE DRIVE NUMBER
010026  104403                        TYPOS                   ;;GO TYPE--OCTAL ASCII
010030     002                        .BYTE   2               ;;TYPE 2 DIGIT(S)
010031     000                        .BYTE   0               ;;SUPPRESS LEADING ZEROS
010032  004737  026202                JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
010036  032777  100000  171110        BIT     @SW15,@SWR      ;HALT ON ERROR ?
010044  001401                        BEQ     1$              ;BR IF NOT
010046  000000                        HALT                    ;ERROR HALT
010050  000207                1$:     RTS     PC

                                        ;SOFTWARE TIMEOUT OCCURRED

010052  004737  022274        SWTIM:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
010056  104414  051466                DISPLY  .EM13           ;PRINT THE TIME OUT MESSAGE
010062  004737  022342                JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
010066  004737  022776                JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
010072  004737  023446                JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
010076  004737  026202                JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
010102  004737  024136                JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
010106  000207                        RTS     PC              ;RETURN
```

```
                                ;DRIVE WENT OFFLINE

010110  004737  022274        OFLIN:  JSR     PC,LINE1        ;PRINT LINE 1 OF THE ERROR MESSAGE
010114  104414  051540                DISPLY  .EM14           ;PRINT OFFLINE MESSAGE
010120  004737  022342                JSR     PC,LINE2        ;PRINT LINE 2 OF THE ERROR MESSAGE
010124  004737  022776                JSR     PC,LINE3        ;PRINT LINE 3 OF THE ERROR MESSAGE
010130  004737  023446                JSR     PC,LINE4        ;PRINT LINE 4 OF THE ERROR MESSAGE
010134  004737  026202                JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
010140  004737  024136                JSR     PC,LINE7        ;PRINT LINE 7 OF THE ERROR MESSAGE
010144  000137  031362                JMP     DROP            ;DROP THE DRIVE

                                ;PORT REQUEST TIMEOUT ERROR

010150  004737  022274        PRTIM:  JSR     PC,LINE1        ;TYPE LINE 1 OF THE ERROR MESSAGE
010154  104414  051563                DISPLY  .EM15           ;PRINT PORT TIME OUT MESSAGE
010160  004737  022342                JSR     PC,LINE2        ;TYPE LINE 2 OF THE ERROR MESSAGE
010164  004737  022776                JSR     PC,LINE3        ;TYPE LINE 3 OF THE ERROR MESSAGE
010170  004737  023446                JSR     PC,LINE4        ;TYPE LINE 4 OF THE ERROR MESSAGE
010174  004737  026202                JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
010200  004737  024136                JSR     PC,LINE7        ;TYPE LINE 7 OF THE ERROR MESSAGE
010204  000207                        RTS     PC              ;RETURN

                                ;PROCESS ORDER COMPLETION WITH 'ERROR' & 'DONE' BIT SET

010206  000240                DONE:   NOP
010210  032760  000030  000016  1$:     BIT     @#BIT04!BIT03,$STATUS(R0) ;UNSAFE OCCURRED
010216  001402                        BEQ     2$              ;BR IF NOT
010220  000137  013402                JMP     UNSAF           ;REPORT UNSAFE

010224  032760  040000  000202  2$:     BIT     @#BIT14,$RPER1(R0)      ;SEE IF 'UNS' SET
010232  001402                        BEQ     3$              ;BR IF NO
010234  000137  013402                JMP     UNSAF           ;REPORT UNSAFE

010240  032760  000002  000200  3$:     BIT     @#BIT1,$RPDS(R0) ;SEE IF 'EWN' SET
010246  001402                        BEQ     4$              ;BR IF NOT
010250  000137  013474                JMP     EWNERR          ;REPORT EARLY WARNING

010254  032760  040000  000176  4$:     BIT     @#BIT14,$RPCS2(R0)      ;IS 'WCE' SET ?
010262  001402                        BEQ     5$              ;BRANCH IF NOT SET
010264  000137  011242                JMP     WCKER           ;WRITE CHECK ERROR

010270  032760  040000  000166  5$:     BIT     @#BIT14,$RPCS1(R0)      ;CHECK 'TRE'
010276  001002                        BNE     6$              ;BR IF SET
010300  000137  013146                JMP     TRFER           ;PROCESS 'TRE'

010304  032760  000400  000202  6$:     BIT     @#BIT08,$RPER1(R0)      ;'HCRC  SET?
010312  001402                        BEQ     7$              ;BR IF NOT
010314  000137  011630                JMP     HCRCER          ;PROCESS 'HCRC'

010320  032760  000020  000202  7$:     BIT     @#BIT04,$RPER1(R0)      ;'FMT' SET?
010326  001402                        BEQ     8$              ;BR IF NOT SET
010330  000137  012022                JMP     CKFMT           ;CHECK FORMAT ERROR

010334  032760  000200  000202  8$:     BIT     @#BIT07,$RPER1(R0)      ;'HCE' SET?
010342  001402                        BEQ     9$              ;BR IF NOT SET
010344  000137  012212                JMP     CKHCE           ;CHECK 'HCE' ERROR
```

```
 65 010350  032760  020000  000202  9$:     BIT     #BIT13,$RPER1(R0)       ;'OPI' SET?
 66 010356  001402                          BEQ     10$                     ;BR IF NOT SET
 67 010360  000137  012506                  JMP     OPIER                   ;REPORT 'OPI'
 68
 69 010364  032760  000010  000202  10$:    BIT     #BIT3,$RPER1(R0)        ;'PAR' SET?
 70 010372  001402                          BEQ     11$                     ;BR IF NOT SET
 71 010374  000137  012640                  JMP     PARER                   ;REPORT 'PAR'
 72
 73 010400  032760  000040  000202  11$:    BIT     #BIT5,$RPER1(R0)        ;'WCF' SET?
 74 010406  001402                          BEQ     12$                     ;BR IF NOT SET
 75 010410  000137  013304                  JMP     WCFER                   ;REPORT 'WCF'
 76
 77 010414  032760  002000  000202  12$:    BIT     #BIT10,$RPER1(R0)       ;'IAE' SET?
 78 010422  001402                          BEQ     13$                     ;BR IF NOT SET
 79 010424  000137  012732                  JMP     IAEER                   ;REPORT 'IAE'
 80
 81 010430  032760  004000  000202  13$:    BIT     #BIT11,$RPER1(R0)       ;'WLE' SET?
 82 010436  001402                          BEQ     14$                     ;BR IF NOT SET
 83 010440  000137  012764                  JMP     WLEER                   ;REPORT 'WLF'
 84
 85 010444  032760  001000  000202  14$:    BIT     #BIT9,$RPER1(R0)        ;'AOE' SET?
 86 010452  001405                          BEQ     15$                     ;BR IF NOT SET
 87 010454  032760  002000  000200          BIT     #BIT10,$RPDS(R0)        ;'LBT' SET?
 88 010462  001401                          BEQ     15$                     ;BR IF NOT SET
 89 010464  000207                          RTS     PC                      ;'AOE' & 'LBT' SET, EXIT
 90
 91 010466  032760  010000  000202  15$:    BIT     #BIT12,$RPER1(R0)       ;SEE IF 'DTE' SET
 92 010474  001402                          BEQ     16$                     ;BR IF NOT
 93 010476  000137  012616                  JMP     DTEER                   ;REPORT 'DTE' ERROR
 94
 95 010502  005760  000202          16$:    TST     $RPER1(R0)              ;SEE IF 'DCK' SET
 96 010506  100002                          BPL     17$                     ;BR IF NOT
 97 010510  000137  010546                  JMP     DCKER                   ;PROCESS  DCK'
 98
 99 010514  032760  040000  000230  17$:    BIT     #BIT14,$RPER3(R0)       ;'SKI' SET
100 010522  001006                          BNE     18$                     ;BRANCH IF SKI, OCYL SET
101 010524  032760  100000  000230          BIT     #BIT15,$RPER3(R0)       ;BAD SPOT ?
102 010532  001004                          BNE     19$                     ;BRANCH IF SO
103 010534  000137  011770                  JMP     DRVER                   ;REPORT ERROR
104
105 010540  000137  013246          18$:    JMP     SKIER                   ;REPORT SKI ERROR
106 010544  000207                  19$:    RTS     PC                      ;EXIT FROM ERROR ANALYSIS  ROUT.
```

```
                                        ;PROCESS DATA ('DCK') CHECK ERROR
                                        
010546  004737  016420        DCKER:  JSR     PC,SPOTCK       ;SEE IF ERROR AT A BAD SPOT
010552  000207                  RTS     PC              ;IT IS, DON'T REPORT IT
010554  032760  000100  000202        BIT     #ECM,$RPER1(R0) ;ECM ERROR SET ?
010562  001411                  BEQ     1$              ;BR IF NO
010564  022760  010040  000232        CMP     #10040,$RPEC1(R0)       ;OTHERWISE RPEC1=10040
010572  001024                  BNE     2$              ;REPORT ECC LOGICAL FAILURE
010574  004737  022274                JSR     PC,LINE1        ;FIRST LINE OF ERROR MESSAGE
010600  104414  053536                DISPLY  .EM52           ;ECM ERROR - ECC UNCORRECTABLE
010604  000451                  BR      7$
                                        
010606  026027  000232  010040  1$:     CMP     $RPEC1(R0),#10040       ;IS POSITION COUNT OVER MAXIMUM ?
010614  101013                  BHI     2$              ;BR IF YES
010616  005760  000232                TST     $RPEC1(R0)      ;POSITION COUNT 0 ?
010622  001410                  BEQ     2$              ;BR IF YES
010624  005760  000234                TST     $RPEC2(R0)      ;VALUE IN PATTERN REGISTER ?
010630  001033                  BNE     6$              ;BR IF YES
010632  004737  022274                JSR     PC,LINE1        ;TYPE FIRST LINE OF ERROR MESSAGE
010636  104414  053360                DISPLY  .EM47           ;TYPE 'ECC LOGIC ERROR'
010642  000404                  BR      3$
010644  004737  022274        2$:     JSR     PC,LINE1        ;TYPE FIRST LINE OF ERROR MESSAGE
010650  104414  053220                DISPLY  .EM45           ;TYPE 'ECC LOGIC ERROR'
010654  004737  022342        3$:     JSR     PC,LINE2        ;TYPE LINE 2 OF ERROR MESSAGE
010660  004737  026202                JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
010664  012737  000003  001324        MOV     #3,RETRY        ;RETRY COUNT
010672  004737  017120                JSR     PC,$RETRY       ;RETRY THE ORDER
010676  000403                  BR      4$              ;RETRY WAS NOT SUCCESSFUL
010700  004737  024076                JSR     PC,LINE6C       ;PRINT 'CORRECTED ON N RETRY(S)'
010704  000402                  BR      5$              ;FINISH THE ERROR REPORT

010706  004737  024104        4$:     JSR     PC,LINE6D       ;PRINT 'UNCORRECTABLE AFTER N RETRY(S)'
010712  004737  024136        5$:     JSR     PC,LINE7        ;TYPE LINE 7 OF ERROR MESSAGE
010716  000207                  RTS     PC              ;RETURN

                                        ;THE VALUES IN THE ECC REGISTERS ARE CORRECT, REPORT 'DCK' ERROR

010720  004737  022274        6$:     JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
010724  104414  051640                DISPLY  .EM21           ;DATA CHECK ERROR
010730                          7$:
```

```
                ;
010730  004737  022342          DCKER1: JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
010734  004737  022776                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
010740  004737  023446                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
010744  004737  016436                  JSR     PC,PRTBAD       ;SEE IF BAD SECTOR TO BE PRINTED
010750  012737  110100  001322          MOV     #BIT15!BIT12!BIT06,MASK ;LOAD ERROR MASK
010756  032760  010000  000202          BIT     #BIT12,$RPER1(R0)       ;IS IT 'DTE' ?
010764  001012                          BNE     2$              ;BR IF YES
010766  032760  000100  000202          BIT     #BIT06,$RPER1(R0)       ;IS IT 'ECH' ?
010774  001403                          BEQ     1$              ;BR IF NO
010776  004737  011202                  JSR     PC,13$          ;COMPARE BAD DATA
011002  000403                          BR      2$
011004  004737  024056          1$:     JSR     PC,LINE6        ;PRINT 'SECTOR IS ECC CORRECTABLE'
011010  000460                          BR      10$             ;FINISH THE ERROR REPORT
011012  012737  000012  001324  2$:     MOV     #10.,RETRY      ;RETRY COUNT
                ;
011020  004737  020240          3$:     JSR     PC,GODRIV       ;RETRY
011024  005760  000016          4$:     TST     $TATUS(R0)      ;TEST FOR DONE
011030  001775                          BEQ     4$              ;BR IF NOT DONE
011032  100057                          BPL     12$             ;BR IF NOT ERROR
011034  032760  000200  000016          BIT     #BIT7,$TATUS(R0)  ;SEE IF ORDER TERMINIATED NORMALLY
011042  001006                          BNE     5$              ;BR IF NOT
011044  004737  026202                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
011050  104414  055650                  DISPLY  .LIN8M          ;'DIFFERENT ERROR DURING RETRY'
011054  000137  007636                  JMP     ERPRC1          ;SEE WHICH ERROR
                ;
011060  033760  001322  000202  5$:     BIT     MASK,$RPER1(R0) ;LOOK AT CURRENT ERROR
011066  001412                          BEQ     7$              ;BR IF DIFFERENT ERROR
011070  032760  010100  000202          BIT     #BIT12!BIT6,$RPER1(R0)  ;'ECH' OR 'DTE' STILL SET ?
011076  001421                          BEQ     9$              ;BR IF NEITHER SET
011100  105237  001325                  INCB    RETRY+1         ;INCREMENT RETRY COUNT
011104  123737  001324  001325          CMPB    RETRY,RETRY+1   ;DONE ?
011112  001342                          BNE     3$              ;BR IF NOT
011114  004737  024354          7$:     JSR     PC,LINE8        ;PRINT LINE 8 OF ERROR MESSAGE
011120  004737  026106          8$:     JSR     PC,INCHRD       ;INCREMENT 'HARD' ERROR COUNT
011124  004737  026202                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
011130  004737  024136                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
011134  004737  016436                  JSR     PC,PRTBAD       ;PRINT THE BAD SECTOR
011140  000436                          BR      15$             ;CLEAN UP AND RETURN
                ;
011142  004737  024056          9$:     JSR     PC,LINE6        ;PRINT 'SECTOR IS ECC CORRECTABLE'
011146  004737  024014                  JSR     PC,LINE5B       ;PRINT LINE 5B OF THE ERROR MESSAGE
011152  004737  026062          10$:    JSR     PC,INCSOF       ;INCREMENT 'SOFT' ERROR COUNT
011156  004737  015576                  JSR     PC,ECC          ;CORRECT THE ERROR USING ECC AND CHECK IT
011162  000407                          BR      13$             ;COMPARE THE BUFFER
                ;
011164  004737  024104          11$:    JSR     PC,LINE6D       ;PRINT 'UNCORRECTABLE AFTER N RETRY(S)'
011170  000753                          BR      8$              ;INCREMENT ERROR COUNT
                ;
011172  004737  024070          12$:    JSR     PC,LINE6A       ;PRINT 'SECTOR READ CORRECTLY AFTER N RETRY(S)'
011176  004737  026062                  JSR     PC,INCSOF       ;INCREMENT 'SOFT' ERROR COUNT
011202  012737  000001  001352  13$:    MOV     #1,FRSTER       ;SET PROCESSING 'DCKER' INDICATOR
011210  004737  014200                  JSR     PC,CMPARD       ;COMPARE THE BUFFER
011214  105737  001353                  TSTB    FRSTER+1        ;ERROR IN COMPARE ?
011220  100406                          BMI     15$             ;BRANCH IF ERROR
011222  004737  026202                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
011226  104414  056117                  DISPLY  .LIN9G          ;'DATA COMPARE OK' MESSAGE
```

```
*1 011252  000737  020136          140:    JSR     PC,ERR7         ;PRINT LINE # OF ERROR MESSAGE
*2 011256  000240                  150:    NOP
*3 011260  000207                          RTS     PC              ;RETURN
```

```
                                ;WRITE CHECK ERROR PROCESSING
                                ;
011242  005760  000202          WCHER:  TST     @WPER1(R0)      ;SEE IF 'DCK' SET ALSO
011246  100436                  BMI     2$                      ;BR IF IT IS
011250  004737  022276                  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
011254  104414  051744                  DISPLY  .EM23           ;PRINT WCE & DCK NOT SET
011260  005037  001322                  CLR     MASK            ;CLEAR ERROR MASK
011264  004737  022342                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
011270  004737  022776                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
011274  004737  023446                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
011300  004737  023536                  JSR     PC,LINE5        ;PRINT LINE 5 OF ERROR MESSAGE
011304  004737  026202                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
011310  012737  000003  001324          MOV     #3,RETRY        ;RETRY LIMIT
011316  004737  017120                  JSR     PC,$RETRY       ;RETRY THE OPERATION
011322  000403                  BR      1$                      ;RETRY UNSUCESSFUL
                                ;
011324  004737  024076                  JSR     PC,LINE6C       ;PRINT 'CORRECTED ON N RETRY(S)'
011330  000532                  BR      15$                     ;FINISH PROCESSING THE ERROR
                                ;
011332  004737  024104          1$:     JSR     PC,LINE6D       ;PRINT 'UNCORRECTABLE AFTER N RETRY(S)'
011336  000527                  BR      15$                     ;FINISH PROCESSING THE ERROR
                                ;
011340  012737  000012  001324  2$:      MOV     #10.,RETRY      ;RETRY LIMIT
011346  004737  022276                  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
011352  012737  051671  011500          MOV     #EM22,$ERMSG    ;ASSUME THAT EM22 WILL BE PRINTED
011360  032760  040000  000176          BIT     #BIT14,$RPCS2(R0)  ;DID 'WCK' ALSO SET ?
011366  001003                  BNE     3$                      ;BR IF IT DID
011370  012737  052572  011500          MOV     #EM37,$ERMSG    ;MESSAGE FOR 'DCK' AND 'WCK' NOT DURING
                                ;                               ;WRITE CHECK
011376  032760  000100  000202  3$:      BIT     #ECH,@WPER1(R0) ;WRITE CHECK ERROR WITH . ECH SET ?
011404  001410                  BEQ     4$                      ;BRANCH IF NOT
011406  012737  053536  011500          MOV     #EM32,$ERMSG    ;REPORT 'ECH' ERROR
011414  022760  010040  000232          CMP     #10040,$RPEC1(R0)       ;OTHERWISE RPEC1=10040
011422  001017                  BNE     5$                      ;REPORT ECC LOGICAL FAILURE
011424  000424                  BR      7$
                                ;
011426  026027  000232  010040  4$:      CMP     $RPEC1(R0),#10040       ;IS POSITION COUNT OVER MAXIMUM ?
011434  101012                  BHI     5$                      ;BR IF YES
011436  005760  000232                  TST     $RPEC1(R0)      ;POSITION COUNT 0 ?
011442  001407                  BEQ     5$                      ;BR IF YES
011444  005760  000234                  TST     $RPEC2(R0)      ;VALUE IN PATTERN REGISTER ?
011450  001007                  BNE     6$                      ;BR IF YES
011452  012737  053360  011500          MOV     #EM47,$ERMSG    ;ELSE, REPORT THE ECC LOGIC FAILURE
011460  000403                  BR      6$
011462  012737  053220  011500  5$:      MOV     #EM45,$ERMSG    ;ELSE, REPORT THE ECC LOGIC FAILURE
011470  012737  000003  001324  6$:      MOV     #3,RETRY        ;RETRY LIMIT
                                ;
011476  104414                  7$:     DISPLY                  ;TYPE THE ERROR MESSAGE
011500  000000                  8$:     .WORD   0               ;MESSAGE ADDRESS GOES HERE
011502  004737  022342                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
011506  004737  022776                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
011512  004737  023446                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
011516  004737  023536                  JSR     PC,LINE5        ;PRINT LINE 5 OF ERROR MESSAGE
                                ;
011522  004737  020240          9$:     JSR     PC,$DORIV       ;RETRY THE ORDER
011526  005760  000016          10$:    TST     $STATUS(R0)     ;ORDER FINISHED ?
011532  001775                  BEQ     10$                     ;BR IF NOT
```

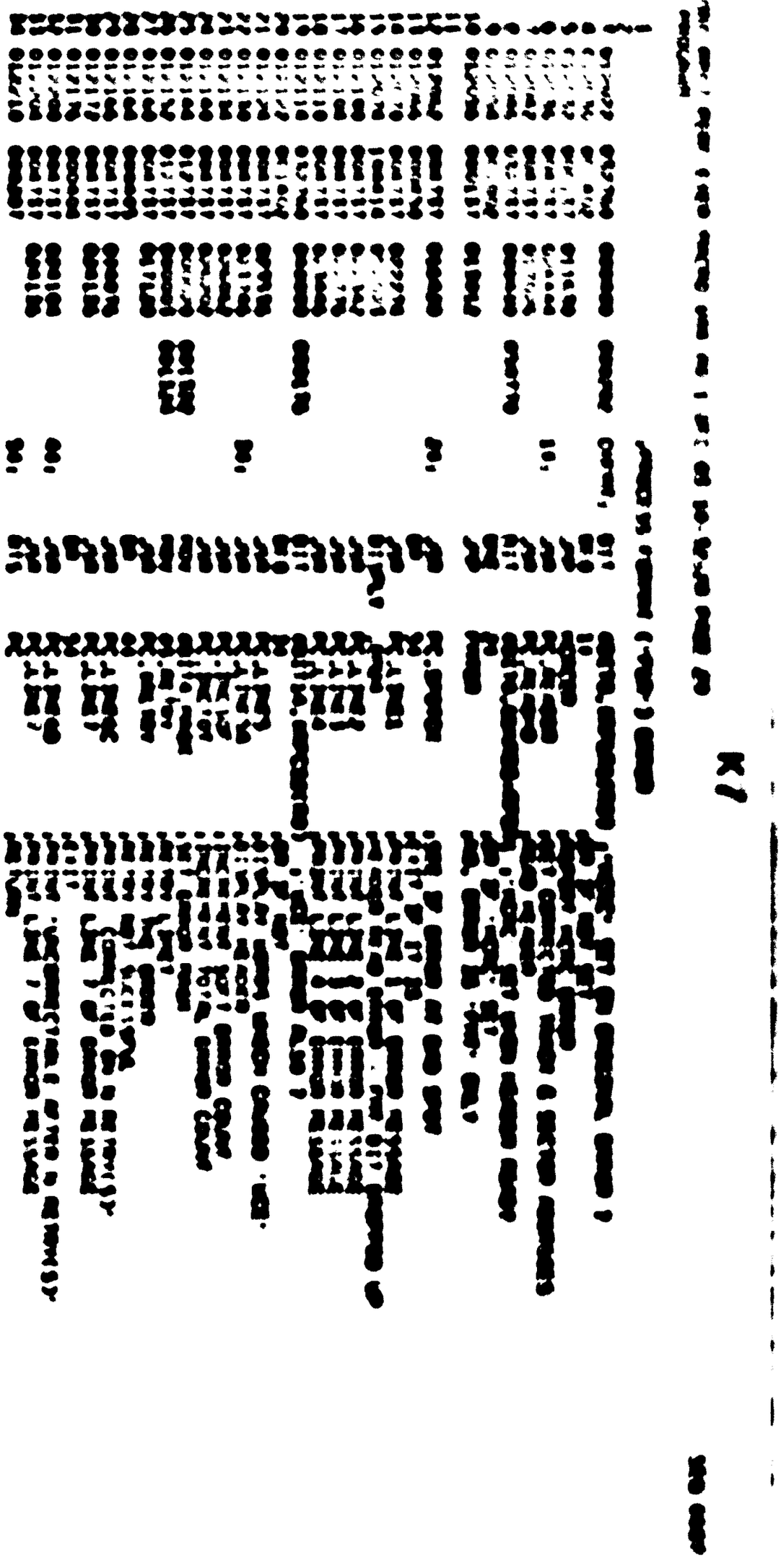
```
60 011536 100005                    BPT     110             ;GO IF ERROR ON ERROR
61 011540 105237 001525             INCB    RETRY+1         ;INCREMENT RETRY COUNT
62 011544 004737 004076             JSR     PC,LINE4C       ;PRINT 'CORRECTED ON N RETRY(S)'
63 011550 000416                    BR      130             ;FINISH ERROR PROCESSING
64
65 011552 105237 001525         110:    INCB    RETRY+1         ;INCREMENT RETRY COUNT
66 011556 121737 001526 001525          CMPB    RETRY,RETRY+1   ;DONE ?
67 011564 001003                    BNE     10              ;BR IF AT RETRY LIMIT
68 011566 032760 100000 000000          BIT     #BIT15,ERRPC01(R0)  ; DCH: SET
69 011574 001401                    BEQ     120             ;GO IF NOT - DIFFERENT ERROR
70 011576 000752                    BR      90              ;CONTINUE RETRY
77
78 011600 004737 004356         120:    JSR     PC,LINE8        ;PRINT LINE 8 - 'DIFFERENT ERROR'
79 011604 000405                    BR      150
80 011606 004737 000042         130:    JSR     PC,INCSOF       ;COUNT AS A SOFT ERROR
81 011612 000402                    BR      150             ;EXIT
82
83 011614 004737 026106         140:    JSR     PC,INCHRD       ;COUNT AS A HARD ERROR
84 011620 004737 026202         150:    JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
85 011624 004737 024136                 JSR     PC,LINE7        ;FINISH THE ERROR MESSAGE
86 011630 000207                    RTS     PC              ;RETURN
```

```
                                        ;REPORT  MODE  ERROR

011690  006737  010620          MODERR: JSR     PC,SPNTCH       ;SET UP ERROR AT END SPOT
011694  000136                  BIS     #36                     ;SET UP IF IT IS
011698  006737  020366          JSR     PC,.NEXERR              ;GO TO ERROR SECTOR HANDLER
011702  006737  017726          JSR     PC,.NEXERR              ;GET THE HEAD INFORMATION
011706  006737  022276          JSR     PC,.LINE1               ;PRINT LINE 1 OF ERROR MESSAGE
011712  104416  000417          BYTPNT  ERRPC                   ;PRINT ERROR
011716  006737  022342          JSR     PC,.LINE2               ;PRINT LINE 2 OF ERROR MESSAGE
011722  006737  022376          JSR     PC,.LINE3               ;PRINT LINE 3 OF ERROR MESSAGE
011726  006737  023006          JSR     PC,.LINE4               ;PRINT LINE 4 OF ERROR MESSAGE
011732  006737  023106          JSR     PC,.LINE5A              ;PRINT THE HEADER INFORMATION
011736  012760  000000  000176          MOV     #0,@SAVPSW(R0)  ; WAS  ERROR ALSO ?
011744  001402                  BEQ     10$                     ;BR IF NOT
011746  006737  023136          JSR     PC,.LINE9               ;0:WAS BY HEADER SWITCH CAUSED  WAS
011752  006737  023142          JSR     PC,.INCSOF              ;INCREMENT  SOFT  ERROR COUNT
011756  006737  023152          JSR     PC,.INCTOT              ;INCREMENT TOTAL ERROR COUNT
011762  012737  000010  001522  10$:    MOV     #10,@#ERRCNT    ;SET ERROR COUNT
011770  012737  000003  001520          MOV     #3,@#RETRY      ;RESTORE RETRY
011776  006737  017620          JSR     PC,@#ERRTBV             ;RESTORE ERROR
012002  000005                  RESET                           ;RESTORE NOT SUCCESSFUL
012004  006737  020076          JSR     PC,.LINEC               ;PRINT 'CORRECTED ON N RETRY(S)'
012010  006737  023136          JSR     PC,.LINE7               ;PRINT LINE 7 OF ERROR MESSAGE
012014  000406                  BR      30$                     ;EXIT
012016  006737  020106          JSR     PC,.LINE6               ;PRINT 'UNCORRECTABLE AFTER N RETRY(S)'
012022  006737  023136          JSR     PC,.LINE7               ;PRINT LINE 7 OF ERROR MESSAGE
012026  000207          30$:    RTS     PC                      ;RETURN


                                        ;REPORT DRIVE ERROR

011770  006737  022276  DRVERR: JSR     PC,.LINE1               ;PRINT LINE 1 OF ERROR MESSAGE
011774  104416  000417          BYTPNT  DRVPC                   ;REPORT DRIVE ERROR
012000  006737  022342          JSR     PC,.LINE2               ;PRINT LINE 2 OF ERROR MESSAGE
012004  006737  022376          JSR     PC,.LINE3               ;PRINT LINE 3 OF ERROR MESSAGE
012010  006737  023052          JSR     PC,.INCTOT              ;INCREMENT TOTAL ERROR COUNT
012014  006737  023136          JSR     PC,.LINE7               ;PRINT LINE 7 OF ERROR MESSAGE
012020  000207                  RTS     PC                      ;RETURN
```



```
                                        ;PRINT LINE 1 OF ERROR MESSAGE
                                        ;PRINT LINE 2 OF ERROR MESSAGE
                                        ;PRINT LINE 3C OF ERROR MESSAGE
                                        ;POP STACK INTO CYLNDR
                                        ;PRINT LINE 5A OF THE ERROR MESSAGE
                                        ;INCREMENT MISPOSITIONING COUNT
                                        ;INCREMENT TOTAL ERROR COUNT
                                        ;PRINT LINE 7A OF ERROR MESSAGE
                                        ;RECALIBRATE
                                        ;EXIT
```

```
57
58 012466  004737  016420          SPERR:  JSR     PC,SPOTCK       ;SEE IF ERROR AT BAD SPOT
59 012472  000207                          RTS     PC              ;RETURN IF IT IS
60 012474  004737  022270                  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
61 012500  104414  000151                  DISPLY  ,EMS2           ;SPOT ERROR
62 012504  004737  022342                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
63 012510  004737  022376                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
64 012514  004737  023000                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
65 012520  004737  020662                  JSR     PC,INCSOF       ;COUNT 'SOFT' AS A 'HARD' DATA ERROR
66                                                                 ;SPOT AS BOTH FOR NOW
67 012524  004737  020700                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
68 012530  012737  000000  001322          MOV     #0,ITEMP        ;ERROR FLAG
69 012536  012737  000001  001320  SPERR1: MOV     #1,RETRY        ;RETRY LIMIT
70 012544  004737  017120                  JSR     PC,SETDEV       ;SET THE DRIVE
71 012550  000405                          BR      10              ;SET UNIT # A
72 012552  004737  020076                  JSR     PC,LINE6C       ;PRINT 'CORRECTION # ATTEMPTED RETRY(S)'
73 012556  004737  024136                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
74 012562  000207                          RTS     PC              ;EXIT
75 012564  004737  024106          10:     JSR     PC,LINE6D       ;PRINT 'UNCORRECTABLE # ATTEMPTED RETRY(S)'
76 012570  004737  024136                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
77 012574  000207                          RTS     PC              ;RETURN
78
79                                 ;REPORT 'DTE' ERROR
80
81 012576  004737  016420          DTERR:  JSR     PC,SPOTCK       ;SEE IF ERROR AT BAD SPOT
82 012602  000207                          RTS     PC              ;RETURN IF IT IS
83 012604  004737  022270                  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
84 012610  104414  000151                  DISPLY  ,EMS2           ;'DTE' ERROR
85 012614  000137  010750                  JMP     DCKER1          ;FINISH PROCESSING THE 'DTE' ERROR
```

```
                                        ;REPORT 'PAR' ERROR
                                        
012640  004737  022274          PARER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
012644  104414  052364                  DISPLY  .EM33           ;REPORT 'PAR'
012650  004737  022342                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
012654  004737  023102                  JSR     PC,LINE3E       ;PRINT LINE 3E OF ERROR MESSAGE
012660  004737  023446                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
012664  004737  026202                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
012670  012737  000010  001322          MOV     #BIT03,MASK     ;ERROR MASK
012676  012737  000003  001324          MOV     #3,RETRY        ;RETRY LIMIT
012704  004737  017120                  JSR     PC,$RETRY       ;RETRY ORDER
012710  000405                          BR      2$              ;RETRY UNSUCESSFUL
012712  004737  024076                  JSR     PC,LINE6C       ;PRINT 'CORRECTED ON N RETRY(S)'
012716  004737  024136          1$:     JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
012722  000207                          RTS     PC              ;EXIT
012724  004737  024104          2$:     JSR     PC,LINE6D       ;PRINT 'UNCORRECTABLE AFTER N RETRY(S)'
012730  000772                          BR      1$              ;FINISH ERROR MESSAGE

                                        ;REPORT 'IAE' ERROR

012732  004737  022274          IAEER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
012736  104414  052503                  DISPLY  .EM35           ;REPORT 'IAE'
012742  004737  022342                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
012746  004737  023170                  JSR     PC,LINE3F       ;PRINT LINE 3F OF ERROR MESSAGE
012752  004737  026202                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
012756  004737  024136                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
012762  000207                          RTS     PC              ;RETURN

                                        ;REPORT 'WLE' ERROR

012764  004737  022274          WLEER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
012770  104414  052541                  DISPLY  .EM36           ;REPORT 'WLE'
012774  004737  022342                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
013000  004737  026202                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
013004  004737  024136                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
013010  000207                          RTS     PC              ;RETURN

                                        ;REPORT FORMAT ERROR

013012  004737  022274          FMTER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
013016  104414  052152                  DISPLY  .EM26           ;FORMAT ERROR
013022  004737  022342                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
013026  004737  022776                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
013032  004737  023446                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
013036  032760  040000  000176          BIT     #BIT14,$RPCS2(R0)  ;'WCE' ERROR ALSO ?
013044  001402                          BEQ     1$              ;BR IF NOT
013046  004737  023536                  JSR     PC,LINE5        ;DISPLAY WORDS WHICH CAUSED 'WCE'
013052  004737  023746          1$:     JSR     PC,LINE5A       ;PRINT LINE 5A OF ERROR MESSAGE
013056  004737  026202                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
013062  004737  024136                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
013066  000207                          RTS     PC
```

```
                                        ;REPORT HEADER COMPARE ERROR
                                        
013070  004737  022274          HCEER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
013074  104414  052177                  DISPLY  ,EM27           ;HEADER COMPARE ERROR
013100  004737  022342                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
013104  004737  022776                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
013110  004737  023446                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
013114  032760  040000  000176          BIT     #BIT14,$RPCS2(R0)  ;'WCE' ERROR ALSO ?
013122  001402                          BEQ     1$              ;BR IF NOT
013124  004737  023536                  JSR     PC,LINE5        ;DISPLAY WORDS WHICH CAUSED 'WCE'
013130  004737  023746          1$:     JSR     PC,LINE5A       ;PRINT LINE 5A OF ERROR MESSAGE
013134  004737  026202                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
013140  004737  024136                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
013144  000207                          RTS     PC              ;RETURN
                                        
                                        ;PROCESS CONTROL/INTERFACE TRANSFER ERROR
                                        
013146  004737  022274          TRFER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
013152  104414  052644                  DISPLY  ,EM40           ;RHXX OR UNIBUS TRANSFER ERROR
013156  004737  022342                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
013162  004737  022776                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
013166  004737  023446                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
013172  004737  026202                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
013176  032760  121400  000176          BIT     #BIT15!BIT13!BIT9!BIT8,$RPCS2(R0)  ;'DLT','UPE','MXF', 'MDPE' SET ?
013204  001415                          BEQ     2$              ;BR IF NONE SET
013206  012737  000003  001324          MOV     #3,RETRY        ;RETRY LIMIT
013214  005037  001322                  CLR     MASK            ;CLEAR ERROR MASK
013220  004737  017120                  JSR     PC,$RETRY       ;RETRY THE OPERATION
013224  000403                          BR      1$              ;RETURN HERE IF RETRY UNSUCESSFUL
013226  004737  024076                  JSR     PC,LINE6C       ;PRINT  CORRECTED ON N RETRY(S)'
013232  000402                          BR      2$              ;FINISH THE ERROR REPORT
013234  004737  024104          1$:     JSR     PC,LINE6D       ;PRINT 'UNCORRECTABLE AFTER N RETRY(S)'
013240  004737  024136          2$:     JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
013244  000207                          RTS     PC
                                        
                                        ;PROCESS 'SKI' ERRORS
                                        
013246  004737  022274          SKIER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
013252  104414  053435                  DISPLY  ,EM50           ;'SKI' ERROR
013256  004737  022342                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
013262  004737  023012                  JSR     PC,LINE3B       ;PRINT LINE 3B OF ERROR MESSAGE
013266  004737  026202                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
013272  004737  026132                  JSR     PC,INCSKI       ;INCREMENT 'SKI' OR 'OCYL' ERROR COUNT
013276  004737  024256                  JSR     PC,LINE7A       ;PRINT LINE 7A OF ERROR MESSAGE
013302  000207                          RTS     PC
```

```
                                        ;REPORT WRITE CLOCK FAILURE ('WCF')
                                        ;
013304  004737  022274          WCFER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
013310  104414  052441                  DISPLY  .EM34           ;REPORT WRITE CLOCK FAILURE
013314  004737  022342                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
013320  004737  023004                  JSR     PC,LINE3A       ;PRINT LINE 3A OF ERROR MESSAGE
013324  004737  023446                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
013330  004737  026202                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
013334  004737  016436                  JSR     PC,PRTBAD       ;SEE IF BAD SECTOR TO BE PRINTED
013340  012737  000003  001324          MOV     #3,RETRY        ;RETRY COUNT
013346  012737  000040  001322          MOV     #BIT05,MASK     ;ERROR MASK
013354  004737  017120                  JSR     PC,$RETRY       ;RETRY THE ORDER
013360  000405                          BR      2$              ;RETURN HERE IF RETRY UNSUCESSFUL
013362  004737  024076                  JSR     PC,LINE6C       ;PRINT 'CORRECTED ON N RETRY(S)'
013366  004737  024136          1$:     JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
013372  000207                          RTS     PC
013374  004737  024104          2$:     JSR     PC,LINE6D       ;PRINT 'UNCORRECTABLE AFTER N RETRY(S)'
013400  000772                          BR      1$

                                        ;PROCESS DRIVE UNSAFE ERROR

013402  004737  022274          UNSAF:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
013406  104414  053602                  DISPLY  .EM60           ;REPORT DRIVE UNSAFE
013412  004737  022342                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
013416  004737  022776                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
013422  004737  026202                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
013426  012737  040000  001322          MOV     #BIT14,MASK     ;RETRY MASK
013434  012737  000003  001324          MOV     #3,RETRY        ;RETRY COUNT
013442  004737  017120                  JSR     PC,$RETRY       ;RETRY THE ORDER
013446  000403                          BR      1$              ;RETRY WAS UNSUCESSFUL
013450  004737  024076                  JSR     PC,LINE6C       ;PRINT 'CORRECTED ON N RETRY(S)
013454  000404                          BR      2$              ;CONTINUE WITH ERROR REPORT
013456  004737  024104          1$:     JSR     PC,LINE6D       ;PRINT 'UNCORRECTABLE AFTER N RETRY(S)'
013462  000137  031362                  JMP     DROP            ;DROP UNSAFE DRIVE

013466  004737  024136          2$:     JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
013472  000207                          RTS     PC              ;RETURN

                                        ;REPORT EARLY WARNING ERROR

013474  004737  022274          EWNERR: JSR     PC,LINE1        ;SKIP LINES
013500  000240                          NOP
013502  032760  000002  000230          BIT     #BIT1,$RPER3(R0)        ;'TPE' SET?
013510  001403                          BEQ     1$              ;BR IF NO
013512  104414  053625                  DISPLY  .EM70           ;PRINT TEMPERATURE WARNING ERROR
013516  000411                          BR      3$              ;TYPE REMAINING ERROR MESSAGE
013520  032760  000004  000230  1$:     BIT     #BIT2,$RPER3(R0)        ;'AIR' SET?
013526  001403                          BEQ     2$              ;BR IF NO
013530  104414  053704                  DISPLY  .EM71           ;'AIR SYSTEM WARNING ERROR'
013534  000402                          BR      3$
013536  104414  053762          2$:     DISPLY  .EM72           ;'EARLY WARNING ERROR,AIR TPE NOT SET
013542  004737  022342          3$:     JSR     PC,LINE2        ;ERROR MESSAGE
013546  004737  022776                  JSR     PC,LINE3        ;PRINT OUT
013552  004737  026202                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
013556  012737  000007  001322          MOV     #7,MASK         ;SET UP ERROR MASK
013564  012737  000003  001324          MOV     #3,RETRY        ;RETRY COUNT
013572  004737  017120                  JSR     PC,$RETRY       ;RETRY THE ORDER
```

```
58 013576  000403                    BR      4$              ;RETRY UNSUCESSFUL RETURN
59 013600  004737  024076             JSR     PC,LINE6C       ;PRINT 'CORRECTED ON N RETRY(S)'
60 013604  000207                    RTS     PC              ;RETURN
61 013606  004737  024104     4$:     JSR     PC,LINE6D       ;PRINT 'UNCORRECTABLE AFTER N RETRY(S)'
62 013612  000137  031362             JMP     DROP            ;DROP DRIVE
```

```
                                        ;REPORT AN  UNKNOWN' DATA PATTERN
                                        ;
                                        ;R1 = POINTER FIRST WORD OF DATA BUFFER

013616  005737  001334        NOMTCH: TST     CFLAG           ;WAS (^C) TYPED?
013622  001060                        BNE     5$              ;BR IF YES
013624  105737  001352        1$:     TSTB    FRSTER          ;FIRST ERROR IN THE SECTOR ?
013630  001013                        BNE     2$              ;BR IF NOT OR IF PROCESSING 'DCKER'
013632  004737  022274                JSR     PC,LINE1        ;TYPE LINE 1 OF ERROR MESSAGE
013636  104414  053027                DISPLY  ,EM43           ;'CAN'T MATCH DATA WITH PATTERN'
013642  004737  022342                JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
013646  004737  023004                JSR     PC,LINE3A       ;PRINT LINE 3A OF ERROR MESSAGE
013652  004737  023446                JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
013656  000404                        BR      3$              ;CONTINUE PROCESSING ERROR
013660  104414  053027        2$:     DISPLY  ,EM43           ;'CAN'T MATCH DATA WITH PATTERN'
013664  104414  001203                DISPLY  ,$CRLF          ;CR-LF
013670  104414  056015        3$:     DISPLY  ,LIN9I          ;HEADER FOR DATA PRINTOUT
013674  005737  001334        4$:     TST     CFLAG           ;WAS (^C) TYPED?
013700  001031                        BNE     5$              ;BR IF YES
013702  010146                        MOV     R1,-(SP)        ;ADDRESS OF WORD POSITION
013704  004737  024366                JSR     PC,LINOCT       ;TYPE WORD NUMBER
013710  005237  001374                INC     WRDPOS          ;INC WORD POSITION
013714  013746  001374                MOV     WRDPOS,-(SP)    ;PUT THE WORD POS ON THE STACK
013720  004737  033412                JSR     PC,$SB2D        ;CONVERT IT TO DECIMAL
013724  004537  032742                JSR     R5,$REPLZ       ;TYPE IT (REPLACE LEADING ZEROS)
013730  000006                        .WORD   6               ;TYPE 6 DIGITS
013732  104414  057771                DISPLY  ,PERIOD         ;TYPE '.'
013736  104414  056610                DISPLY  ,BLNKS2         ;TYPE 2 BLANKS
013742  012146                        MOV     (R1)+,-(SP)     ;ADDRESS OF DATA WORD
013744  004737  024366                JSR     PC,LINOCT       ;TYPE DATA
013750  104414  001203                DISPLY  ,$CRLF          ;CR-LF
013754  023727  001374  000003        CMP     WRDPOS,#3       ;DONE ALL WORDS ?
013762  001344                        BNE     4$              ;BR IF NO
013764  062701  000770        5$:     ADD     #<252.*2.>,R1   ;INCREMENT BUFFER POINTER - 4 MATCHING WORDS
013770  005002                        CLR     R2              ;CLEAR 'WORDS TO COMPARE' COUNT IN R2
013772  012737  177777  001352        MOV     #-1,FRSTER      ;SET ERROR FOUND INDICATOR
014000  013737  001460  001364        MOV     CMPLMT,LIMIT    ;RESET THE COMPARE ERROR TYPEOUT LIMIT
014006  000207                        RTS     PC              ;RETURN
```

```
                                        ;CHECK ERROR BITS IN THE RMXX & RPO7 REGISTERS
                                        ;
014010  032760  060000  000166  CKERR:  BIT     #60000,@RPCS1(R0)  ;SEE IF 'TRE' OR 'MCPE' SET
014016  001015                          BNE     1$                 ;BR IT EITHER SET
014020  032760  177400  000176          BIT     #177400,@RPCS2(R0) ;SEE IF ERROR BITS IN CS2 SET
014026  001011                          BNE     1$                 ;BR IF ANY SET
014030  005760  000202                  TST     @RPER1(R0)         ;ANY BITS SET IN ER1
014034  001006                          BNE     1$                 ;BR IF ANY SET
014036  005760  000230                  TST     @RPER3(R0)         ;ANY BITS SET IN ER3 ?
014042  001003                          BNE     1$                 ;BR IF YES
                                        ;
014044  005760  000226                  TST     @RPER2(R0)         ;ANY BIT SET IN ER2
014050  000416                          BR      2$                 ;BR IF NO
                                        ;
014052  004737  022274          1$:     JSR     PC,LINE1           ;PRINT LINE 1 OF ERROR MESSAGE
014056  104414  053123                  DISPLY  .EM44              ;ERROR BITS SET, BUT NO ERROR BITS SET
014062  004737  022342                  JSR     PC,LINE2           ;PRINT LINE 2 OF ERROR MESSAGE
014066  004737  022776                  JSR     PC,LINE3           ;PRINT LINE 3 OF ERROR MESSAGE
014072  004737  023446                  JSR     PC,LINE4           ;PRINT LINE 4 OF ERROR MESSAGE
014076  004737  026202                  JSR     PC,INCTOT          ;INCREMENT TOTAL ERROR COUNT
014102  004737  024136                  JSR     PC,LINE7           ;PRINT LINE 7 OF ERROR MESSAGE
014106  000207                  2$:     RTS     PC                 ;RETURN
                                        ;
                                        ;CHECK BUS ADDRESS REGISTER & WORD COUNT REGISTER
                                        ;
014110  005760  000170          CKBUS:  TST     @RPWC(R0)          ;CHECK WORD COUNT
014114  001010                          BNE     1$                 ;BR IF NOT ZERO
014116  016046  000020                  MOV     @WRDL(R0),-(SP)    ;WORD LENGTH
014122  006316                          ASL     (SP)               ;CHANGE INTO BYTE COUNT
014124  066016  000006                  ADD     @BUF(R0),(SP)      ;ADD THE STARTING LOCATION
014130  022660  000172                  CMP     (SP)+,@RPBA(R0)    ;BUFFER ADDRESS PROPER ?
014134  001416                          BEQ     2$                 ;BR IF OK
014136  004737  022274          1$:     JSR     PC,LINE1           ;PRINT LINE 1 OF ERROR MESSAGE
014142  104414  052702                  DISPLY  .EM41              ;BUS ADDRESS OR WORD COUNT INCORRECT
014146  004737  022342                  JSR     PC,LINE2           ;PRINT LINE 2 OF ERROR MESSAGE
014152  004737  023034                  JSR     PC,LINE3D          ;PRINT LINE 3D OF ERROR MESSAGE
014156  004737  023446                  JSR     PC,LINE4           ;PRINT LINE 4 OF ERROR MESSAGE
014162  004737  026202                  JSR     PC,INCTOT          ;INCREMENT TOTAL ERROR COUNT
014166  004737  024136                  JSR     PC,LINE7           ;PRINT LINE 7 OF ERROR MESSAGE
014172  000207                  2$:     RTS     PC
```

```
                                ;COMPARE THE BUFFER
                        ;
014174  005037  001552  CMPAR:  CLR     FRSTER          ;CLEAR 'FIRST ERROR' AND 'NO DCK' INDICATOR
                        ;
014200  132760  000004  000024 CMPARD: BITB    #4,@CODE(R0)    ;SEE IF READ COMMAND
014206  001001                 BNE     1$              ;BR IF IT IS
014210  000207                 RTS     PC              ;RETURN
                        ;
014212  005037  001562  1$:     CLR     ERCTR           ;CLEAR THE ERROR COUNTER
014216  016001  000006          MOV     @BUF(R0),R1     ;BUFFER ADDRESS
014222  016037  000020  001566 MOV     @WPOL(R0),CMCNT ;WORD COUNT TO WORKING LOCATION
014230  066037  000170  001566 ADD     @WPWC(R0),CMCNT ;CALCULATE WORDS TRANSFERED
014236  001001                 BNE     2$
014240  000207                 RTS     PC              ;EXIT--NO WORDS XFERED
                        ;
014242  016037  000012  001570 2$:     MOV     @CYL(R0),CMCYL  ;CYLINDER ADDRESS WORKING LOCATION
014250  052737  150000  001570 BIS     #150000,CMCYL   ;SET MFG, USER AND FMT BITS
014256  016037  000010  001572 MOV     @SEC(R0),CMSEC  ;SECTOR & TRACK ADDRESSES TO WORKING LOCNS
014264  013737  001460  001564 MOV     CMPLMT,LIMIT    ;DISPLAY LIMIT
014272  005237  001564          INC     LIMIT           ;CONVERT PARAMETER INTO LIMIT VALUE
014276  012737  177777  001550 CMSTR:  MOV     #-1,ZROIND      ;CLEAR THE 'ZERO'S' INDICATOR
014304  005037  001556          CLR     SAVER1          ;CLEAR THE R1 SAVE WORD
014310  005037  001560          CLR     SAVER5          ;CLEAR THE R5 SAVE WORD
014314  023760  001566  000022 CMP     CMCNT,@SSEC(R0) ;IS BUFFER SIZE GREATER THAN ONE SECTOR ?
014322  101005                 BHI     1$              ;BR IF IT IS
014324  013702  001566          MOV     CMCNT,R2        ;LESS THAN, USE REMAINING BUFFER
014330  005037  001566          CLR     CMCNT           ;SET COUNTER TO 0
014334  000405                 BR      2$
014336  016002  000022  1$:     MOV     @SSEC(R0),R2    ;COMPARE SECTOR
014342  166037  000022  001566 SUB     @SSEC(R0),CMCNT ;DECREMENT WORD COUNT
014350  126027  000024  000005 2$:     CMPB    @CODE(R0),#5    ;READ HEADER & DATA?
014356  001034                 BNE     CMDAT           ;BR IF NOT
                        ;
                                ;COMPARE HEADER WORDS
                        ;
014360  012705  001570 CMHED:  MOV     #CMCYL,R5       ;ADDRESS OF COMPARING CYLINDER
014364  052711  150000          BIS     #150000,(R1)    ;SET BITS INCASE BAD SECTOR ENCOUNTER
014370  022521                 CMP     (R5)+,(R1)+     ;CHECK CYLINDER
014372  001405                 BEQ     1$              ;BR IF COMPARE OK
014374  012737  177777  001574 MOV     #-1,WRDPOS      ;INDICATE WORD POSITION IS HEADER WRD 1
014402  004737  014436          JSR     PC,CMSTR2       ;REPORT ERROR
014406  022521                 CMP     (R5)+,(R1)+     ;COMPARE SECTOR/TRACK
014410  001405                 BEQ     2$              ;BR IF EQ
014412  012737  177776  001574 MOV     #-2,WRDPOS      ;INDICATE WORD POSITION IS HEADER WRD 2
014420  004737  014436          JSR     PC,CMSTR2       ;REPORT ERROR
014424  162702  000002 2$:     SUB     #2,R2           ;SUBTRACT HEADER LENGTH FROM SIZE
014430  003007                 BGT     CMDAT           ;BR IF NOT FINISHED
014432  000137  015052          JMP     CMPRX           ;COMPARE THE DATA PORTION
                        ;
014436  005237  001562 CMSTR2: INC     ERCTR           ;INCREMENT THE ERROR COUNT
014442  004737  015060          JSR     PC,CMPRT        ;REPORT THE COMPARISON ERROR
014446  000207                 RTS     PC              ;CHECK THE REST OF THE HEADER
```

```
                                ;COMPARE DATA FIELD
                ;
014450  012737  177777  001374  CMDAT:  MOV     #-1,WORDPOS     ;INITIALIZE WORD POSITION
014456  004737  015506          10:     JSR     PC,MATCH        ;FIND THE PATTERN
014462  000403                          BR      1$              ;FOUND A PATTERN
014464  004737  015616                  JSR     PC,NOMTCH       ;RETURN HERE IF NO MATCH WITH PATTERN MODE
014470  000463                          BR      7$              ;BYPASS COMPARE ROUTINE
                ;
014472  011405          20$:    MOV     (R4),R5         ;ADDRESS OF PATTERN ADDRESS IN R4
014474  012703  000020          30$:    MOV     #16.,R3         ;R3 IS PATTERN POS COUNTER
014500  005237  001374          25$:    INC     WRDPOS          ;INCREMENT TO CURRENT WORD POSITION
014504  022125                  40$:    CMP     (R1)+,(R5)+     ;COMPARE BUFFER WITH PATTERN
014506  001016                          BNE     4$              ;BR IF NOT EQUAL
014510  005737  001362                  TST     ERCTR           ;ERRORS DETECTED ?
014514  001406                          BEQ     3$              ;BR IF NO ERRORS
014516  032777  000010  164430          BIT     #SW3,@#SWR      ;SWITCH 3 SET ?
014524  001402                          BEQ     3$              ;BR IF NOT SET
014526  004737  015060                  JSR     PC,CMPRT        ;DISPLAY THE WORD
                ;
014532  005302          50$:    DEC     R2              ;DECREMENT SIZE COUNT
014534  003441                  BLE     7$              ;BR WHEN AT END
014536  005303                  DEC     R3              ;DECREMENT PATT POS COUNT
014540  001357                  BNE     2$              ;BR IF NOT AT END OF PATT
014542  000755                  BR      1$              ;RESTART THE PATTERN
                ;
014544  005761  177776          40$:    TST     -2(R1)          ;IS MISCOMPARED CHARACTER=0
014550  001410                          BEQ     5$              ;BR IF YES
014552  012737  177777  001350          MOV     #-1,ZROIND      ;SET NON-ZERO MISCOMPARED INDICATOR
014560  005237  001362                  INC     ERCTR           ;INCREMENT THE ERROR COUNTER
014564  004737  015060                  JSR     PC,CMPRT        ;REPORT ERROR
014570  000760                          BR      3$              ;CONTINUE COMPARE
                ;
014572  105737  001352          50$:    TSTB    FRSTER          ;FIRST ERROR?
014576  100412                          BMI     6$              ;BR IF NOT
014600  005037  001350                  CLR     ZROIND          ;SET THE ZERO INDICATOR
014604  010137  001356                  MOV     R1,SAVER1       ;SAVE CURRENT R1
014610  010537  001360                  MOV     R5,SAVER5       ;SAVE CURRENT R5
014614  013737  001374  001354          MOV     WRDPOS,SAVPOS   ;SAVE CURRENT WORD POSITION
014622  000743                          BR      3$              ;CONTINUE COMPARE
014624  005737  001350          60$:    TST     ZROIND          ;ANY MISCOMPARIONS NOT ZEROS ?
014630  001740                          BEQ     3$              ;BR IF NONE-ALL ERRORS=ZERO
014632  004737  015060                  JSR     PC,CMPRT        ;REPORT ERROR
014636  000735                          BR      3$              ;CONTINUE COMPARING
                ;
014640  126027  000024  000005  70$:    CMPB    @CODE(R0),#5    ;READ HEAD AND DATA ?
014646  001414                          BEQ     9$              ;YES
014650  013702  001366          80$:    MOV     CMCNT,R2        ;SET COUNTER = REMAIN BUFFER LENGTH
014654  020227  000004                  CMP     R2,#4           ;IS THERE AT LEAST 4 WORDS TO MATCH PATTERN ?
014660  002474                          BLT     CMPRX           ;BR IF NO
014662  162737  000400  001366          SUB     #256.,CMCNT     ;GREATER THAN A SECTOR ?
014670  003667                          BLE     CMDAT           ;NO,RETURN TO COMPARE LOOP
014672  012702  000400                  MOV     #256.,R2        ;SET COUNTER =SECTOR SIZE
014676  000664                          BR      CMDAT           ;RETURN TO COMPARE LOOP
                ;
014700  023727  001366  000002  90$:    CMP     CMCNT,#2        ;IS THERE AT LEAST 2 WORDS TO COMPARE HEADER ?
014706  002461                          BLT     CMPRX           ;BR IF NO
014710  105237  001372                  INCB    CMSEC           ;INCREMENT COUNTER
```

```
58 014716  121760  001572  000152          CMPB    CYREC.SECLIM(R0)        ;RUN SECTOR 0 ?
59 014722  101426                          BLOS    10$                     ;NO
60 014724  105037  001572                  CLRB    CYREC                   ;RESET SECTOR 0
61 014730  105237  001573                  INCB    CYTRK                   ;INCREMENT TRACK 0
62 014734  121760  001573  000154          CMPB    CYTRK,TRKLIM(R0)        ;RUN TRACK 0 ?
63 014742  101414                          BLOS    10$                     ;NO
64 014744  105037  001573                  CLRB    CYTRK                   ;RESET TRACK 0
65 014750  005237  001570                  INC     CYCYL                   ;INCREMENT CYLINDER NUMBER
66 014754  013746  001570                  MOV     CYCYL,-(SP)             ;GET COMPARING CYLINDER
67 014760  042716  150000                  BIC     #150000,(SP)            ;SAVE ONLY THE CYLINDER BITS
68 014764  021660  000150                  CMP     (SP)+,CYLLIM(R0)        ;LAST CYLINDER ?
69 014770  101401                          BLOS    10$                     ;NO
70 014772  000427                          BR      CYRTN                   ;NORMAL RETURN,NOT WRAP AROUND
71
72 014776  012705  001570          10$:    MOV     #CYCYL,R5               ;ADDRESS OF COMPARING CYLINDER
73 015000  052711  150000                  BIS     #150000,(R1)            ;SET BITS INCASE BAD SECTOR ENCOUNTER
74 015004  020521                          CMP     (R5)+,(R1)+             ;COMPARE 1ST HEADER WORD
75 015006  001405                          BEQ     11$                     ;MATCH
76 015010  012737  177777  001574          MOV     #-1,WRDPOS              ;INDICATE WORD POSITION IS HEADER WRD 1
77 015016  004737  014436                  JSR     PC,CMSTR2               ;NOT MATCH
78 015022  020521                  11$:    CMP     (R5)+,(R1)+             ;SECOND WORD OF HEADER
79 015024  001405                          BEQ     12$                     ;MATCH
80 015026  012737  177776  001574          MOV     #-2,WRDPOS              ;INDICATE WORD POSITION IS HEADER WRD 2
81 015034  004737  014436                  JSR     PC,CMSTR2               ;NOT MATCH
82
83 015040  162737  000002  001566  12$:    SUB     #2,CMCNT                ;ADJUST WORD COUNT
84 015046  003401                          BLE     CMPRX                   ;COMPARE IS DONE
85 015050  000677                          BR      8$                      ;RETURN TO COMPARE LOOP
86
93 015052  004737  015440          CMPRX:  JSR     PC,ENDCMP               ;PRINT LAST LINE OF ERRORS
94 015056  000207                          RTS     PC
```

```
                                        ;TYPE DATA COMPARE ERRORS
                                        ;
 0 015060  000250                       CMPDT,  NOP
10
11 015062  005737  001356               10:     TST     SAVED1          ;PRINT SAVED VALUES ?
12 015066  001005                               BNE     20              ;GO IF YES
13 015070  004737  015166                       JSR     PC,PRINT        ;PRINT INITIAL MESSAGE INFO
14 015074  004737  015250                       JSR     PC,70           ;PRINT REMAINDER OF MESSAGE
15 015100  000436                               BR      30              ;EXIT
16 015102                              20:
   015102  010146                               MOV     R1,-(SP)        ;SAVE R1 ON STACK
   015104  010246                               MOV     R2,-(SP)        ;SAVE R2 ON STACK
   015106  011746  001374                       MOV     @#177374,-(SP)  ;SAVE ERROR ON STACK
17 015112  011701  001356                       MOV     SAVED1,R1       ;DISPLAY SAVED R1
18 015116  011702  001360                       MOV     SAVED2,R2       ;DISPLAY SAVED R2
19 015122  011703  001354  001376               MOV     SAVED0,@#177376 ;DISPLAY SAVED ERROR POSITION
20 015130  004737  015166                       JSR     PC,PRINT        ;PRINT INITIAL MESSAGE INFO
21 015134  004737  015250                       JSR     PC,70           ;SAVED VALUES
22 015140  005037  001346                       CLR     SAVED1          ;CLEAR SAVED DISPLAY INDICATORS
23 015144  005037  001360                       CLR     SAVED2          ;AND OTHER ONE
24 015150  014137  001354                       MOV     (SP)+,SAVED0    ;LOAD STACK INTO SAVED0
   015154  014502                               MOV     (SP)+,R2        ;LOAD STACK INTO R2
   015156  014501                               MOV     (SP)+,R1        ;LOAD STACK INTO R1
25 015160  004737  015250                       JSR     PC,70           ;PRINT REMAINDER OF MESSAGE
26 015164  000207                       30:     RTS     PC              ;RETURN
27
28 015166  105737  001352               40:     TSTB    @#177560        ;PRINT ERROR ?
29 015172  100003                               BPL     50              ;GO IF NOT
30 015174  001013                               BNE     60              ;GO IF FIRST ERROR AND DOUBLE-DCH ERROR,
31                                                                      ;I.E. IF FIRST ERROR AND CHARACTER ERROR W/O DCH
32 015176  004737  027270               50:     JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
33 015202  104414                       DISPLY  (R4)+                   ;DATA COMPARE ERROR W/O DCH ERROR
34 015204  004737  027342                       JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
35 015210  004737  027504                       JSR     PC,LINE3A       ;PRINT LINE 3A OF ERROR MESSAGE
36 015214  004737  027446                       JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
37 015220  000404                               BR      60              ;GO TO TYPE HEADER
38
39 015222  104414  059670               56:     DISPLY  .LINE8          ;PRINT 'DATA COMPARISON ERRORS'
40 015226  104414  001205                       DISPLY  .BCRLF          ;CR LF
41 015232  104414  055722               60:     DISPLY  .LINE9          ;PRINT '        WORD EXPCTD RECEIVD
42                                                                      ;       ADDR    POS     DATA    DATA'
43 015236  012737  177777  001352               MOV     #-1,FIRSTER     ;SET FIRST ERROR FLAG
44 015244  000207                               RTS     PC              ;RETURN
45
46 015246  005737  001356               70:     TST     CFLAG           ;WAS (PC) TYPED?
47 015252  100412                               BMI     90              ;GO IF YES
48 015254  005737  001364                       TST     LIMIT           ;TYPEOUT LIMIT REACHED ?
49 015260  001403                               BEQ     80              ;GO IF IT WAS
50 015262  005337  001364                       DEC     LIMIT           ;DECREMENT LIMIT COUNTER
51 015266  001005                               BNE     100             ;GO IF NOT AT LIMIT
52 015270  032777  000200  163664       80:     BIT     #BIT7,@#SWR     ;PRINT ALL DATA COMPARE ERRORS ?
53 015276  001001                               BNE     100             ;GO IF YES
54 015300  000207                       90:     RTS     PC              ;RETURN
55
56 015302  010146                       100:    MOV     R1,-(SP)        ;BUFFER ADDRESS
57 015304  162716  000002                       SUB     #2,(SP)         ;ADJUST ADDRESS
58 015310  004737  024366                       JSR     PC,LINOCT       ;TYPE IT
```

```
017716  004737  017750                  JSR     PC,110          ;GET WORD POSITION
017722  016546  177776                  MOV     [illegible],(SP) ;PUT EXPECTED DATA ON THE STACK
017726  004737  [illegible]             JSR     PC,LINDCT       ;TYPE IT
017732  104414  [illegible]             DISPLY  [illegible]     ;TYPE 2 BLANKS
017736  016146  177776                  MOV     [illegible],(SP) ;[illegible] DATA
017742  004737  [illegible]             JSR     PC,LINDCT       ;TYPE IT
017746  104414  001205                  DISPLY  .CRLF           ;CR LF
017752  000207                          RTS     PC              ;RETURN

017754  022727  001570  177777  110:    CMP     [illegible],0 1 ;IS IT HEADER WORD 1 ?
017762  001005                          BNE     120             ;GO IF NO
017764  104414  [illegible]             DISPLY  [illegible]     ;TYPE        [illegible]
017770  000420                          BR      140
017772  022727  001570  177776  120:    CMP     [illegible],0 2 ;IS IT HEADER WORD 2 ?
020000  001005                          BNE     130             ;GO IF NO
020002  104414  [illegible]             DISPLY  [illegible]     ;TYPE        [illegible]
020006  000411                          BR      140
020010  011746  001570          130:    MOV     [illegible],(SP) ;PUT THE WORD POS ON THE STACK
020014  004737  [illegible]             JSR     PC,[illegible]  ;CONVERT IT TO DECIMAL
020020  004737  [illegible]             JSR     PC,[illegible]  ;TYPE IT (REPLACE LEADING ZEROS)
020024  000006                          .WORD   6               ;TYPE 6 DIGITS
020026  104414  [illegible]             DISPLY  [illegible]     ;TYPE [illegible]
020032  104414  [illegible]     140:    DISPLY  [illegible]     ;TYPE 2 BLANKS
020036  000207                          RTS     PC


                        ;LAST LINE OF COMPARE ERROR REPORTING

020040  105737  001556          ERROP:  TSTB    [illegible]-1   ;ANY COMPARE ERRORS FOUND ?
020044  001417                          BEQ     20              ;GO IF NOT
020046  005737  001562                  TST     ERCTR           ;SEE HOW MANY ERRORS
020052  001410                          BEQ     10              ;GO IF ONLY CAN'T MATCH PATTERN
020054  104414  [illegible]             DISPLY  [illegible]     ;NUMBER OF [illegible]
020060  011746  001562                  MOV     ERCTR,(SP)      ;NUMBER OF ERRORS
020064  004737  [illegible]             JSR     PC,LINDEC       ;TYPE IT
020070  104414  001205                  DISPLY  .CRLF           ;CR LF
020074  004737  [illegible]     10:     JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
020100  004737  [illegible]             JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
020104  000207                  20:     RTS     PC              ;RETURN
```

```
[illegible]
```

```
                                        ;ADD ECC TO CORRECT THE DATA ERROR
                                        ;
016000  016037  000234  001402  ECC:    MOV     @#REC2(R0),ECMSK0       ;GET THE ERROR BIT MASK
016004  016046  001402                  MOV     @#REC1(R0),-(SP)        ;GET WORD COUNT REGISTER (2'S COMP)
016010  060516  000000                  ADD     R5,@#REC(R0),(SP)       ;ADD WORDS REQUESTED
016014  005216                          INC     (SP)
016016  005726                          TST     (SP)+                   ;ADJUST STACK
016020  005267                          INC     ECWRD                   ;BUMP TO WORD WITHIN SECTOR
016024  005046                  10:     CLR     -(SP)                   ;CLEAR NEXT STACK LOCN
016026  016046  000002                  SUB     2(R0),-(SP)             ;SECTOR SIZE
016032  004737  012156                  JSR     PC,DIV                  ;DIVIDE WORDS WITHIN BY SECTOR SIZE
016036  005716                          TST     (SP)                    ;PARTIAL SECTOR WORKED ?
016040  001415                          BEQ     20                      ;BR IF NOT
016042  005216                          INC     (SP)                    ;COMPENSATE DATA NUMBER OF BYTES
016044  163617  001400                  SUB     (SP),ECSEC              ;SUBTRACT SECTOR IN ERROR
016050  122700  000005  000005          MOVB    #5,@CCOM(R0)            ;SET OPERATION, READ HEAD & DATA
016056  005007                          CLR     PC                      ;GO IF NOT
016060  062737  000004  001400          ADD     #4,ECSEC                ;ADD HEADER SIZE (IN BYTES) BACK IN
016066  000405                          BR      30                      ;GO ADJUST THE STACK POINTER
016070  162737  001000  001400  20:     SUB     #256.+2,ECSEC           ;SUBTRACT SECTOR DATA FIELD SIZE (IN BYTES)
016076  062706  000004          30:     ADD     #4,SP                   ;ADJUST THE STACK POINTER
016102  016037  000752  001376          MOV     @#C1(R0),ECBIT          ;ECC POSITION COUNT
016110  005337  001376                  DEC     ECBIT                   ;ADJUST BIT POSITION
016114  013737  001376  001406          MOV     ECBIT,ECWRD             ;LOAD THE WORD COUNT LOCATION
016122  042737  177760  001376          BIC     #^C17,ECBIT             ;SAVE THE BIT OFFSET COUNT
016130  042737  000017  001406          BIC     #17,ECWRD               ;CLEAR THE BIT OFFSET
016136  006237  001406                  ASR     ECWRD                   ;CONVERT TO BYTE COUNT(DIVIDE BY 2)
016142  006237  001406                  ASR     ECWRD                   ;CONVERT TO BYTE COUNT(DIVIDE BY 4)
016146  006237  001406                  ASR     ECWRD                   ;CONVERT TO BYTE COUNT(DIVIDE BY 8.)
016152  104414  056145                  DISPLY  ,LIN10A                 ;'ERROR BURST BEGINS AT '
016156  013746  001406                  MOV     ECWRD,-(SP)             ;PUT THE WORD COUNT ON THE STACK
016162  006216                          ASR     (SP)                    ;GET STARTING WORD FOR MESSAGE(DIVIDE BY 16.)
016164  011637  001374                  MOV     (SP),WRDPOS             ;GET CURRENT WORD POSITION
016170  004737  015412                  JSR     PC,$BIN                 ;CONVERT IT TO DECIMAL
016174  004737  015426                  JSR     PC,$OUTPR               ;TYPE IT (LEFT JUSTIFIED)
016200  104414  057771                  DISPLY  ,PERIOD                 ;TYPE '.'
016204  104414  056201                  DISPLY  ,LIN10B                 ;' IN DATA FIELD OF ERROR SECTOR'
016210  063737  001400  001406          ADD     ECSEC,ECWRD             ;FIND THE BEGINNING OF THE ERROR BURST
016216  026037  000172  001406          CMP     @#RPBA(R0),ECWRD        ;SEE IF BURST WAS IN DATA READ
016224  101002                          BHI     40                      ;BR IF IN DATA READ
016226  000137  016406                  JMP     ECC2                    ;NOT IN DATA READ - REPORT IT
                                        ;
016232  016037  000234  001402  40:     MOV     @#REC2(R0),ECMSK0       ;GET THE ERROR BIT MASK
016236  005037  001404                  CLR     ECMSK1                  ;CLEAR THE UPPER MASK WORD
016242  005337  001376          50:     DEC     ECBIT                   ;DECREMENT THE BIT OFFSET COUNT
016246  002405                          BLT     60                      ;BR IF DONE
016250  006337  001402                  ASL     ECMSK0                  ;SHIFT THE ERROR MASK
016254  006137  001404                  ROL     ECMSK1                  ;SHIFT THE LOWER INTO THE UPPER
016260  000770                          BR      50                      ;CONTINUE THE SHIFT
                                        ;
016262  017737  163320  001412  60:     MOV     @ECWRD,ECBADD           ;SAVE THE INCORRECT WORD
016270  013746  001402                  MOV     ECMSK0,-(SP)            ;PUT LOWER MASK ON STACK
016274  047716  163306                  BIC     @ECWRD,(SP)             ;CLEAR ERRONEOUS ONE BITS FROM MASK
016300  043777  001402  163300          BIC     ECMSK0,@ECWRD           ;CLEAR ERRONEOUS ONE BITS FROM BAD WORD
016306  052677  163274                  BIS     (SP)+,@ECWRD            ;SET DROPPED BITS
                                        ;
016312  005737  001404                  TST     ECMSK1                  ;DOES ERROR GO INTO NEXT WORD ?
```

```
     58 016116  001415                          BEQ     7$                      ;BR IF NO
     59 016120  013737  001406  001414          MOV     ECWRD,ECWRD1            ;DUPLICATE ADDRESS
     60 016126  062737  000002  001414          ADD     #2,ECWRD1               ;INCREMENT ERROR ADDRESS
     61 016134  026037  000172  001414          CMP     $RPBA(R0),ECWRD1                ;IS NEXT WORD IN THE BUFFER ?
     62 016142  101006                          BMI     8$                      ;BR IF YES, ELSE,
     63 016144  005737  001402                  TST     ECMSK0                  ;WAS ERROR IN FIRST WORD ?
     64 016150  001516                          BEQ     ECC2                    ;BR IF NO
     65 016152  005037  001414          7$:     CLR     ECWRD1                  ;CLEAR 2ND WORD ADDRESS
     66 016156  000414                          BR      ECC1                    ;PRINT WORD CORRECTED
     67
     68 016160  017737  163230  001420  8$:     MOV     @ECWRD1,ECBAD1          ;SAVE THE SECOND BAD WORD
     69 016166  013746  001404                  MOV     ECMSK1,-(SP)            ;PUT THE UPPER MASK ON THE STACK
     70 016172  047716  163216                  BIC     @ECWRD1,(SP)            ;CLEAR ERRONEOUS ONE BITS FROM UPPER MASK
     71 016176  043777  001404  163210          BIC     ECMSK1,@ECWRD1          ;CLEAR ERRONEOUS ONE BITS FROM DATA WORD
     72 016204  052677  163204                  BIS     (SP)+,@ECWRD1           ;SET DROPPED BITS
```

```
                1
 2 016210  104414  056345          ECC1:   DISPLY  .LIN10M         ;HEADER
 6 016214  013746  001406                  MOV     ECWRD,(SP)      ;PUT ECWRD ON THE STACK
   016220  004737  024366                  JSR     PC,LINOCT       ;TYPE ECWRD
 7 016224  013746  001374                  MOV     WRDPOS,(SP)     ;PUT THE WORD POS ON THE STACK
 8 016230  004737  033412                  JSR     PC,$B2D         ;CONVERT IT TO DECIMAL
 9 016234  004537  032742                  JSR     R5,$REPLZ       ;TYPE IT (REPLACE LEADING ZEROS)
10 016240  000006                          .WORD   6               ;TYPE 6 DIGITS
11 016242  104414  057771                  DISPLY  .PERIOD         ;TYPE '.'
12 016246  104414  056610                  DISPLY  .BLNKS2         ;TYPE 2 BLANKS
17 016252  013746  001412                  MOV     ECBADO,(SP)     ;PUT ECBADO ON THE STACK
   016256  004737  024366                  JSR     PC,LINOCT       ;TYPE ECBADO
   016262  104414  056610                  DISPLY  .BLNKS2         ;TYPE 2 BLANKS
   016266  017746  163114                  MOV     @ECWRD,-(SP)    ;PUT @ECWRD ON THE STACK
   016272  004737  024366                  JSR     PC,LINOCT       ;TYPE @ECWRD
   016276  104414  056610                  DISPLY  .BLNKS2         ;TYPE 2 BLANKS
18
19 016302  005737  001414                  TST     ECWRD1          ;PRINT THE NEXT WORD ?
20 016306  001441                          BEQ     ECCX            ;BR IF NOT
21 016310  104414  001203                  DISPLY  .$CRLF          ;CR-LF
25 016314  013746  001414                  MOV     ECWRD1,-(SP)    ;PUT ECWRD1 ON THE STACK
   016320  004737  024366                  JSR     PC,LINOCT       ;TYPE ECWRD1
26 016324  013746  001374                  MOV     WRDPOS,(SP)     ;PUT THE WORD POS ON THE STACK
27 016330  005216                          INC     (SP)            ;INC TO SECOND WORD OF BURST
28 016332  004737  033412                  JSR     PC,$B2D         ;CONVERT IT TO DECIMAL
29 016336  004537  032742                  JSR     R5,$REPLZ       ;TYPE IT (REPLACE LEADING ZEROS)
30 016342  000006                          .WORD   6               ;TYPE 6 DIGITS
31 016344  104414  057771                  DISPLY  .PERIOD         ;TYPE '.'
32 016350  104414  056610                  DISPLY  .BLNKS2         ;TYPE 2 BLANKS
37 016354  013746  001420                  MOV     ECBAD1,-(SP)    ;PUT ECBAD1 ON THE STACK
   016360  004737  024366                  JSR     PC,LINOCT       ;TYPE ECBAD1
   016364  104414  056610                  DISPLY  .BLNKS2         ;TYPE 2 BLANKS
   016370  017746  163020                  MOV     @ECWRD1,-(SP)   ;PUT @ECWRD1 ON THE STACK
   016374  004737  024366                  JSR     PC,LINOCT       ;TYPE @ECWRD1
   016400  104414  056610                  DISPLY  .BLNKS2         ;TYPE 2 BLANKS
38 016404  000402                          BR      ECCX            ;EXIT
39
40 016406  104414  056241          ECC2:   DISPLY  .LIN10C         ;ERROR BURST WAS NOT TRANSFERED TO MEMORY
41 016412  104414  001203          ECCX:   DISPLY  .$CRLF          ;CR LF
42 016416  000207                          RTS     PC              ;RETURN
```

```
                                        ;ROUTINE TO CHECK ERROR REGISTER #3 FOR THE BAD SECTOR BIT
                                        ;CALL
                                        ;       JSR     PC,SPOTCK       ;CALL ROUTINE
                                        ;       .                       ;RETURN HERE IF BAD SPOT IS FOUND, ELSE
                                        ;                               ;RETURN HERE

  16420  032760  100000  000230  SPOTCK: BIT     #BSE,$RPER3(R0) ;SEE IF BSE BIT IS SET.
016426  001002                          BNE     1$              ;BRANCH IF SO, ELSE
016430  062716  000002                  ADD     #2,(SP)         ;ADJUST RETURN ADDRESS.
016434  000207                  1$:     RTS     PC              ;EXIT

                                        ;ROUTINE TO DISPLAY THE SECTOR WHICH GAVE THE HARD ERROR
                                        ;CALL
                                        ;       JSR     PC,PRTBAD       ;CALL ROUTINE

016436  032777  000010  162510  PRTBAD: BIT     #SW3,@SWR       ;PRINT THE BAD SECTOR ?
016444  001520                          BEQ     8$              ;BR IF NOT
016446  016001  000172                  MOV     $RPBA(R0),R1    ;PUT THE END ADDRESS INTO R1
016452  016046  000020                  MOV     $WRDL(R0),-(SP) ;FIND THE BEGINNING OF THE SECTOR
016456  066016  000170                  ADD     $RPWC(R0),(SP)  ;SUBTRACT THE WORDS NOT TRANSFERED
016462  001002                          BNE     1$
016464  005726                          TST     (SP)+           ;RESTORE STACK
016466  000207                          RTS     PC              ;EXIT--NO WORDS XFERRED
016470  005046                  1$:     CLR     -(SP)           ;MAKE THE UPPER DIVIDEND 0
016472  016046  000022                  MOV     $SSEC(R0),-(SP) ;DIVIDE THE WORDS XFERED BY THE SECTOR SIZE
016476  004737  032126                  JSR     PC,$DIV         ;DIVIDE
016502  005716                          TST     (SP)            ;REMANDER = 0 ?
016504  001403                          BEQ     2$              ;BR IF IT IS - COMPLETE SECTOR TRANSFERED
016506  006316                          ASL     (SP)            ;CONVERT THE RESIDUAL SECTOR INTO BYTE COUNT
016510  161601                          SUB     (SP),R1         ;SUBTRACT IT FROM THE END ADDRESS
016512  000410                          BR      3$              ;FINISH THE SIZING
016514  162701  001000          2$:     SUB     #256.*2,R1      ;SUBTRACT FULL SECTOR FROM END ADDR (IN BYTES)
016520  122760  000005  000024          CMPB    #5,$CODE(R0)    ;WAS OPERATION READ HEADER & DATA ?
016525  001002                          BNE     3$              ;BR IF NOT
016530  162701  000004                  SUB     #4,R1           ;SUBTRACT HEADER SIZE FROM ADDR
016534  062706  000004          3$:     ADD     #4,SP           ;RESTORE THE STACK POINTER
016540  104414  001203                  DISPLY  ,$CRLF          ;CR-LF
016544  104414  056473                  DISPLY  ,LIN11H         ;PRINT THE HEADER
016550  122760  000005  000024          CMPB    #5,$CODE(R0)    ;WAS OPERATION READ HEADER & DATA ?
016556  001021                          BNE     4$              ;BR IF NOT
016560  104414  056546                  DISPLY  ,LIN11          ;TYPE 'ADDR    HEADER'
016564  010146                          MOV     R1,-(SP)        ;PUT THE ADDRESS ON THE STACK
016566  004737  024366                  JSR     PC,LINOCT       ;TYPE THE ADDRESS
016572  104414  056607                  DISPLY  ,BLNKS3         ;TYPE 3 BLANKS
016576  012146                          MOV     (R1)+,-(SP)     ;PUT WORD ON STACK
016600  004737  024366                  JSR     PC,LINOCT       ;TYPE THE 1ST HEADER WORD
016604  104414  056611                  DISPLY  ,BLNKS1         ;TYPE 1 BLANK
016610  012146                          MOV     (R1)+,-(SP)     ;PUT WORD ON STACK
016612  004737  024366                  JSR     PC,LINOCT       ;TYPE THE 2ND HEADER WORD
016616  104414  001203                  DISPLY  ,$CRLF          ;CR-LF

016622  104414  056567          4$:     DISPLY  ,LIN11A         ;TYPE 'ADDR    DATA'
016626  012702  000010          5$:     MOV     #8.,R2          ;8. DATA WORDS PER LINE
016632  010146                          MOV     R1,-(SP)        ;PUT THE ADDRESS ON THE STACK
016634  004737  024366                  JSR     PC,LINOCT       ;TYPE THE ADDRESS
016640  104414  056610                  DISPLY  ,BLNKS2         ;TYPE 2 BLANKS
016644  020160  000172          6$:     CMP     R1,$RPBA(R0)    ;PRINTED ALL THE SECTOR ?
```

```
58 016650  001412                    BEQ     7$              ;BR IF ALL PRINTED
59 016652  104414  056611            DISPLY  .BLNKS1         ;TYPE 1 BLANK
60 016656  012146                    MOV     (R1)+,(SP)      ;PUT THE DATA ON THE STACK
61 016660  004737  024366            JSR     PC,LINOCT       ;TYPE THE DATA
62 016664  005302                    DEC     R2              ;DECREMENT THE HORIZONTAL COUNT
63 016666  001366                    BNE     6$              ;BR IF NOT AT THE END OF THE LINE
64 016670  104414  001203            DISPLY  .$CRLF          ;CR LF
65 016674  000754                    BR      5$              ;RESTORE THE WORDS/LINE COUNT
66 016676  104414  001203      7$:   DISPLY  .$CR F          ;CR LF
67 016702  104414  001203            DISPLY  .$CRLF          ;CR LF
68 016706  000207            8$:     RTS     PC              ;RETURN
```

```
                                        ;ROUTINE TO DO A RECALIBRATE USING ACTIVE DPB
                                        ;CALL:
                                        ;       MOV     #DPB,R0         ;DPB ADDRESS
                                        ;       JSR     PC,RECALT
                                        ;       RETURN

016710  010037  016734          RECALT: MOV     R0,2$           ;LOAD THE DPB ADDRESS
016714  116060  000166  000027          MOVB    $RPCS1(R0),$PREVO(R0)   ;SAVE THE PREVIOUS COMMAND
016722  112760  000107  000002          MOVB    #RECAL,$COMND(R0)       ;LOAD THE NEW COMMAND
016730  004037  041364          1$:     JSR     R0,RP07         ;START THE RECALIBRATE
016734  000000                  2$:     .WORD   0               ;DPB ADDRESS
016736  000774                          BR      1$              ;DRIVER DIDN'T ACCEPT THE COMMAND

016740  005760  000016          3$:     TST     $TATUS(R0)      ;SEE IF FINISHED
016744  001775                          BEQ     3$              ;IF EQ NO
016746  004737  024444                  JSR     PC,READDR       ;DECREMENT THE ADDRESSES
016752  012660  000034                  MOV     (SP)+,$PREVA+2(R0)      ;MOVE THE CYLINDER ADDRESS
016756  112660  000033                  MOVB    (SP)+,$PREVA+1(R0)      ;MOVE THE TRACK ADDRESS
016762  112660  000032                  MOVB    (SP)+,$PREVA(R0)        ;MOVE THE SECTOR ADDRESS
016766  005060  000012                  CLR     $CYL(R0)        ;CLEAR THE CURRENT CYLINDER ADDRESS
016772  005060  000010                  CLR     $SEC(R0)        ;CLEAR THE CURRENT TRK/SEC ADDRESS
016776  000207                          RTS     PC              ;RETURN

                                        ;ROUTINE TO A RECAL WITH NO DPB ACTIVE
                                        ;CALL:
                                        ;       MOVB    #DRIVE,GENDPB   ;DRIVE ADDRESS
                                        ;       JSR     PC,RECALO
                                        ;       RETURN

017000  112737  000107  050736  RECALO: MOVB    #RECAL,GENDPB+$COMND    ;RELCALIBRATE COMMAND
017006  004037  041364          1$:     JSR     R0,RP07         ;DRIVER ENTRANCE
017012  050734                          GENDPB                  ;DPB ADDRESS FOR COMMAND
017014  000774                          BR      1$              ;DRIVER DIDN'T ACCEPT THE COMMAND
017016  005737  050752          2$:     TST     GENDPB+$TATUS   ;SEE IF FINISHED
017022  001775                          BEQ     2$              ;BR IF NOT FINISHED
017024  000207                          RTS     PC
```

```
                                        ;UTILITY READ HEADER ROUTINE
                                        ;CALL:
                                        ;       MOV     #DPB,R0         ;DPB ADDRESS
                                        ;       MOV     #SECTOR,-(SP)   ;SECTOR ADDRESS
                                        ;       MOV     #TRACK,-(SP)    ;TRACK ADDRESS
                                        ;       MOV     #CYLINDER,-(SP) ;CYLINDER ADDRESS
                                        ;       JSR     PC,READDR
                                        ;       RETURN

017026  116637  000004  050745 READHD: MOVB    4(SP),GENDPB+$TRK       ;TRACK ADDRESS
017034  116637  000006  050744         MOVB    6(SP),GENDPB+$SEC       ;SECTOR ADDRESS
017042  016637  000002  050746         MOV     2(SP),GENDPB+$CYL       ;CYLINDER ADDRESS
017050  111037  050734                 MOVB    (R0),GENDPB     ;DRIVE NUMBER
017054  112737  000173  050736         MOVB    #RDHD,GENDPB+$COMND     ;COMMAND
017062  012737  177776  050740         MOV     #-2,GENDPB+$WCNT        ;WORD CTR = 2
017070  004037  041364         1$:     JSR     R0,RP07         ;DRIVER ENTRANCE
017074  050734                         GENDPB                  ;DPB ADDRESS FOR COMMAND
017076  000774                         BR      1$              ;DRIVER DIDN'T ACCEPT COMMAND
017100  005737  050752         2$:     TST     GENDPB+$TATUS   ;FINISHED?
017104  001775                         BEQ     2$              ;BR IF NOT
017106  011666  000006                 MOV     (SP),6(SP)      ;ADJUST STACK FOR RETURN
017112  062706  000006                 ADD     #6,SP           ;ADJUST RETRUN POINTER
017116  000207                         RTS     PC              ;RETURN

                                        ;RETRY THE PRESENT OPERATION
                                        ;CALL:
                                        ;       MOV     #COUNT,RETRY    ;RETRY COUNT
                                        ;       JSR     PC,$RETRY
                                        ;       RETURN1                 ;RETRY UNSUCESSFUL
                                        ;       RETURN2                 ;SUCESSFUL RETRY
                                        ;                               ;NOTE: IF A DIFFERENT ERROR OCCURS DURING
                                        ;                               ;RETRY, THE ROUTINE EXITS TO 'ERPRC1'

017120  004737  020240         $RETRY: JSR     PC,GODRIV       ;RE-START ORDER
017124  005760  000016         1$:     TST     $TATUS(R0)      ;ORDER FINISHED?
017130  001775                         BEQ     1$              ;BR IF NOT
017132  100405                         BMI     2$              ;BR IF ERROR
017134  105237  001325                 INCB    RETRY+1         ;INCREMENT RETRY COUNT
017140  062716  000002                 ADD     #2,(SP)         ;INCREMENT RETURN
017144  000436                         BR      6$              ;GO TO EXIT

017146  032760  000200  000016 2$:     BIT     #BIT7,$TATUS(R0)  ;DID ORDER TERMINATE NORMALLY ?
017154  001441                         BEQ     8$              ;BR IF NOT
017156  005737  001322                 TST     MASK            ;IS ERROR MASK 0 ?
017162  001004                         BNE     3$              ;BR IF NOT
017164  005760  000202                 TST     $RPER1(R0)      ;MAKE SURE THAT THE DRIVE ERROR REG IS CLEAR
017170  001025                         BNE     7$              ;BR IF NOT
017172  000415                         BR      5$              ;CONTINUE RETRY

017174  022737  000007  001322 3$:     CMP     #7,MASK         ;EWN ERROR?
017202  001005                         BNE     4$              ;BR IF NO
017204  032760  000002  000200         BIT     #BIT1,$RPDS(R0) ;'EWN' STILL SET?
017212  001414                         BEQ     7$              ;NO, REPORT DIFFERENT ERROR
017214  000404                         BR      5$              ;SET, RETRY

017216  033760  001322  000202 4$:     BIT     MASK,$RPER1(R0) ;SAME ERROR?
017224  001407                         BEQ     7$              ;BR IF NOT
```

```
58 017226  105237  001325          58:     INCB    RETRY+1         ;INCREMENT RETRY COUNT
59 017232  123737  001324  001325          CMPB    RETRY,RETRY+1   ;DONE ?
60 017240  001327                          BNE     $RETRY          ;BR IF NOT DONE
61 017242  000207                  68:     RTS     PC              ;RETURN
62
63 017244  004737  024354          78:     JSR     PC,LINE8        ;REPORT DIFFERENT ERROR
64 017250  004737  024136                  JSR     PC,LINE7        ;PRINT LINE 7
65 017254  005726                          TST     (SP)+           ;ADJUST STACK POINTER FOR DIRECT RETURN
66 017256  000207                          RTS     PC              ;RETURN
67
68 017260  104414  055650          88:     DISPLY  #LINE8          ;'DIFFERENT ERROR DURING RETRY'
69 017264  000137  007636                  JMP     ERPRC1          ;REPORT THE ERROR
```

```
                                ;ROUTINE TO UPDATE THE PERFORMANCE SUMMARY STATISTICS
                                ;CALL:
                                ;       MOV     #DPB,R0         ;DPB ADDRESS
                                ;       JSR     PC,STATIS
                                ;       RETURN
                                
017270  032760  000300  000016  STATIS: BIT     #BIT7!BIT6,STATUS(R0) ;CHECK FOR DATA TERMINATION
017276  001533                  BEQ     7$              ;BR IF NOT DATA TERMINATION
017300  016037  000172  017570  MOV     BUFBA(R0),FACTOR        ;STORE THE FINAL BUFFER ADDRESS
017306  166037  000006  017570  SUB     BUF(R0),FACTOR  ;SUBTRACT THE INITIAL ADDRESS
017314  001524                  BEQ     7$              ;BR IF NO DATA TRANSFER
017316  006237  017570          ASR     FACTOR          ;CONVERT TO A WORD COUNT

017322  122760  000002  000024  CMPB    #2,$CODE(R0)    ;SEE IF COMMAND WAS A WRITE
017330  001404                  BEQ     1$              ;BRANCH IF YES
017332  122760  000000  000024  CMPB    #0,$CODE(R0)    ;PRESENT OPERATION AN AUTO WRITE CHECK ?
017340  001043                  BNE     4$              ;BR IF NO
017342  063760  017570  000042  1$:     ADD     FACTOR,$WTPAS(R0)       ;ADD WORDS WRITTEN PER PASS
017350  005560  000044          ADC     $WTPAS+2(R0)    ;ADD ANY CARRY
017354  063760  017570  000062  ADD     FACTOR,$WTOTL+4(R0)     ;ADD WORDS WRITTEN TO REP COUNTER
017362  005560  000064          ADC     $WTOTL+6(R0)    ;ADD ANY CARRY
017366  016046  000062          MOV     $WTOTL+4(R0),-(SP)      ;GET LOW DIVIDEND
017372  016046  000064          MOV     $WTOTL+6(R0),-(SP)      ;GET HIGH DIVIDEND
017376  012746  041100          MOV     #041100,-(SP)   ;GET LOW DIVISOR
017402  012746  000017          MOV     #17,-(SP)       ;GET HIGH DIVISOR
017406  004737  032174          JSR     PC,$DBDIV       ;DIVIDE BY 1 X10**6
017412  005766  000004          TST     4(SP)           ;DID REP COUNTER REACH LIMIT YET ?
017416  001412                  BEQ     2$              ;BR IF NO
017420  012660  000064          MOV     (SP)+,$WTOTL+6(R0)      ;GET HIGH REMAINDER
017424  012660  000062          MOV     (SP)+,$WTOTL+4(R0)      ;GET LOW REMAINDER
017430  062760  000001  000056  ADD     #1,$WTOTL(R0)   ;UPDATE THE TOTAL WORDS WRITTEN
017436  005560  000060          ADC     $WTOTL+2(R0)    ;ADD ANY CARRY
017442  000401                  BR      3$
017444  022626                  CMP     (SP)+,(SP)+     ;GET HIGH & LOW REMAINDER OFF STACK
017446  005726                  TST     (SP)+           ;GET QUOTIENT OFF STACK

017450  122760  000002  000024  4$:     CMPB    #2,$CODE(R0)    ;SEE IF COMMAND WAS A WRITE
017456  001443                  BEQ     7$              ;BRANCH IF YES
017460  063760  017570  000036  ADD     FACTOR,$RDPAS(R0)       ;ADD WORDS READ PER PASS
017466  005560  000040          ADC     $RDPAS+2(R0)    ;ADD ANY CARRY
017472  063760  017570  000072  ADD     FACTOR,$RTOTL+4(R0)     ;ADD WORDS READ TO REP COUNTER
017500  005560  000074          ADC     $RTOTL+6(R0)    ;ADD ANY CARRY
017504  016046  000072          MOV     $RTOTL+4(R0),-(SP)      ;GET LOW DIVIDEND
017510  016046  000074          MOV     $RTOTL+6(R0),-(SP)      ;GET HIGH DIVIDEND
017514  012746  041100          MOV     #041100,-(SP)   ;GET LOW DIVISOR
017520  012746  000017          MOV     #17,-(SP)       ;GET HIGH DIVISOR
017524  004737  032174          JSR     PC,$DBDIV       ;DIVIDE BY 1 X10**6
017530  005766  000004          TST     4(SP)           ;DID REP COUNTER REACH LIMIT YET ?
017534  001412                  BEQ     5$              ;BR IF NO
017536  012660  000074          MOV     (SP)+,$RTOTL+6(R0)      ;GET HIGH REMAINDER
017542  012660  000072          MOV     (SP)+,$RTOTL+4(R0)      ;GET LOW REMAINDER
017546  062760  000001  000066  ADD     #1,$RTOTL(R0)   ;UPDATE THE TOTAL WORDS READ
017554  005560  000070          ADC     $RTOTL+2(R0)    ;ADD ANY CARRY
017560  000401                  BR      6$
017562  022626                  CMP     (SP)+,(SP)+     ;GET HIGH & LOW REMAINDER OFF STACK
017564  005726                  TST     (SP)+           ;GET QUOTIENT OFF STACK
017566  000207                  RTS     PC
```

[illegible]

```
                                ;ROUTINE TO GET A BUFFER
                                ;CALL:
                                ;       MOV     #BUF,R0         ;BUF ADDRESS
                                ;       CLR     -(SP)           ;CLEAR THE STACK
                                ;       JSR     PC,GETBUF
                                ;       RETURN                  ;BUFFER ADDRESS WILL BE ON THE STACK
                                ;                               ;STACK WILL BE ZERO IF NO BUFFER AVAILABLE
                                ;
017572  010146                  GETBUF: MOV     R1,-(SP)        ;SAVE R1
017574  010246                          MOV     R2,-(SP)        ;SAVE R2
017576  010346                          MOV     R3,-(SP)        ;SAVE R3
017600  011702  001656                  MOV     BUFTBL,R2       ;NUMBER OF SEPARATE BUFFERS
017604  001444                  10$:    BEQ     50$             ;BR IF NONE AVAILABLE
017606  012701  001656                  MOV     #BUFTBL+2,R1    ;FIRST ADDRESS OF ALLOCATION TABLE
017612  026061  000020  000002  20$:    CMP     1+WORD(R0),2(R1) ;SEE IF THERE IS A BLOCK LARGE ENOUGH
017620  101405                          BLOS    30$             ;BRANCH IF IT IS
017622  005302                          DEC     R2              ;DECREMENT TABLE COUNT
017624  001436                          BEQ     50$             ;BR IF THROUGH TABLE
017626  062701  000004                  ADD     #4,R1           ;INCREMENT TABLE POINTER
017632  000767                          BR      10$             ;CONTINUE LOOKING
017634  011166  000010          30$:    MOV     (R1),10(SP)     ;BUFFER ADDRESS TO STACK
017640  164061  000020  000002          SUB     WORD(R0),2(R1)  ;ADJUST BUFFER WORD CNT
017646  001407                          BEQ     31$             ;BR IF DIFFERENCE IS ZERO
017650  004760  000020                  ASL     WORD(R0)        ;CONVERT # WORDS TO BYTES
017656  066011  000020                  ADD     WORD(R0),(R1)   ;MAKE NEW STARTING ADDRESS
017662  006260  000020                  ASR     WORD(R0)        ;RETURN # BYTES TO WORDS
017666  000414                          BR      50$             ;RETURN
017670  005337  001656          40$:    DEC     BUFTBL          ;DECREMENT ENTRY COUNT
017674  001411                  31$:    BEQ     50$             ;BR IF ALLOCATION TABLE EMPTY
017676  005302                          DEC     R2              ;DECREMENT TABLE COUNT
017700  001407                          BEQ     50$             ;BR IF ITEM WAS LAST ENTRY
017702  010103                          MOV     R1,R3           ;MOVE TABLE POINTER
017704  062703  000004                  ADD     #4,R3           ;POINT TO NEXT ENTRY
017710  012321          41$:    MOV     (R3)+,(R1)+             ;MOVE ITEMS
017712  012321                  MOV     (R3)+,(R1)+
017714  005302                  DEC     R2                      ;DECREMENT TABLE COUNT
017716  001374                  BNE     41$                     ;CONTINUE IF NOT AT END OF TABLE
017720  012603          50$:    MOV     (SP)+,R3                ;RESTORE R3
017722  012602                  MOV     (SP)+,R2                ;RESTORE R2
017724  012601                  MOV     (SP)+,R1                ;RESTORE R1
017726  000207                  RTS     PC                      ;RETURN
```

```
                                        ;ROUTINE TO PUT BUFFER D    N TABLE
                                        ;CALL:
                                        ;       R5      ADDR OF
                                        ;       R4      BLOCK ADDRESS
                                        ;       R3      NUMBER
                                        ;
017726  010146                  ENLOG:: MOV     R1,-(SP)        ;SAVE R1
017730  010246                          MOV     R2,-(SP)        ;SAVE R2
017732  010346                          MOV     R3,-(SP)        ;SAVE R3
017734  010446                          MOV     R4,-(SP)        ;SAVE R4
017736  010546                          MOV     R5,-(SP)        ;SAVE R5
017740  016701  001636                  MOV     ENTCNT,R1       ;R1 = NUMBER OF TABLE
017744  011702                          MOV     (R5),R2         ;R2 = BUFFER ADDR
017746  001430                          BEQ     10$             ;IF 0 THEN TABLE
017750  016403  000120                  MOV     BLKSIZ(R4),R3   ;R3 = BLOCK SIZE
017754  005001                          CLR     R1              ;CLEAR R1 FOR BYTE COUNT
017756  105012                  5$:     CLRB    (R2)            ;R2 = ADDR ADJACENT BLOCK
017760  005202                          INC     R2              ;R2 AT ADJACENT BLOCK
017762  005301                          DEC     R1              ;DEC BYTE COUNT
017764  001373                          BNE     5$              ;LOOP TILL 256 BYTES
017766  016701  000000                  MOV     BUFADR,R1       ;R1 = BUFFER ADDR
017772  010102                          MOV     R1,R2           ;R2 = BUFFER ADDR
017774  005202                  6$:     INC     R2              ;INC TO NEXT BYTE
017776  005311                          DEC     (R1)            ;DEC BYTE IN BUFFER
020000  001375                          BNE     6$              ;LOOP TILL 0
020002  016211  000000                  MOV     (R2),(R1)       ;PUT THE BLOCK ADDR INTO THE TABLE
020006  016401  000120  000002          MOV     BLKSIZ(R4),2(R1)        ;PUT BLOCK SIZE INTO TABLE
020014  005201  001636                  INC     ENTCNT          ;INCREMENT ENTRY COUNT
020020  005202                          INC     R2              ;INCREMENT BUFFER ADDR TO NEXT
020022  005041                          CLR     -(R1)           ;CLEAR LOW BYTE OF ADJACENT BLOCK IS IN THE TABLE
020024  016211  000000          20:     MOV     (R2),(R1)       ;BLOCK ADDR ADDED TO TABLE
020030  016401  000120                  MOV     BLKSIZ(R4),(R1) ;BLOCK SIZE ADDED TO TABLE
020034  005201  001636                  INC     ENTCNT          ;INCREMENT ENTRY COUNT
020040  005041                          CLR     -(R1)           ;CLEAR LOW BYTE
020042  016211  000000          30:     MOV     (R2),(R1)       ;ADDRESS BUFFER IS LOWER ADJACENT
020046  016401  000120  000002          MOV     BLKSIZ(R4),2(R1)        ;INCREMENTED BLOCK SIZE
020054  010246                  40:     MOV     R2,-(SP)        ;SAVE R2
020056  013702  001636                  MOV     ENTCNT,R2       ;R2 = ENTRY COUNT
020062  012705  001636                  MOV     #ENTCNT,R5      ;R5 = ADDR OF TABLE
020066  014504  000002          50:     MOV     2(R5),R4        ;R4 = BLOCK SIZE (IN WORDS)
020072  005704                          TST     R4              ;CHECK IF BUFFER EMPTY
020074  001411                          BEQ     60$             ;ADD BLOCK BY ADDRESS
020076  021405                          CMP     (R5),R4         ;R4 STILL BIGGER TO INSERTED ENTRY
020100  103005                          BHIS    60$             ;LOWER ADJACENT IN TABLE
020102  005702                          TST     R2              ;INCREMENT BUFFER
020104  001366                          BNE     50$             ;DECREMENT ENTRY COUNT
020106  005302                          DEC     R2              ;CHECK LOOPING
020110  005726                          TST     (SP)+           ;RESTORE STACK POINTER
020112  000415                          BR      70$             ;END
020114  012602                  60:     MOV     (SP)+,R2        ;RESTORE R2
020116  016465  000002  000002          MOV     2(R4),2(R5)     ;INCREMENT LOWER BLOCK LENGTH
020124  005337  001636                  DEC     ENTCNT          ;DECREMENT ENTRY COUNT
020130  010105                          MOV     R1,R5           ;GET READY TO COMPARES
020132  062705  000004                  ADD     #4,R5           ;INCREMENT TO NEXT ENTRY
020136  012521                  70:     MOV     (R5)+,(R1)+     ;COPY TABLE
020140  012521                          MOV     (R5)+,(R1)+     ;MOVE SIZE FIELD DOWN
020142  005302                          DEC     R2              ;DECREMENT ENTRY COUNT
020144  001374                          BNE     70$             ;BR IF NOT FINISHED
020146  012605                  80:     MOV     (SP)+,R5        ;RESTORE R5
020150  012604                          MOV     (SP)+,R4        ;RESTORE R4
```

```
                                ;FILL THE PASSED BUFFER (IF WRITE OR WRITE CHECK COMMAND)
                                ;CALL:
                                ;       MOV     #BUF,R0         ;BUFF ADDRESS
                                ;       MOV     #BUF.WORD(R0)   ;LOAD BUFFER ADDRESS INTO THE BUF
                                ;       MOVB    #PATTERN,PATT(R0)       ;PATTERN CODE
                                ;       JSR     PC,FILBUF
                                ;       RETURN
                                ;
000162  010046          FILBUF: MOV     R0,-(SP)                ;SAVE THE REGISTERS
000164  132760  000004  000000          BITB    #B.CODE(R0)     ;SEE IF WRITE COMMAND
000172  001020                  BNE     10$             ;BR IF NO
000174  016101  000000          MOV     BUF(R0),R1      ;BUFFER ADDRESS
000200  016002  000002          MOV     NUMB(R0),R2     ;POSITIVE WORD COUNT
000204  116000  000000          MOVB    PATT(R0),R0     ;RELATIVE PATTERN ADDRESS
000210  016005  002154  000000  10$:   MOV     $PBASE(R0),R5   ;PATTERN ADDRESS
000214  012703  000020          MOV     #16.,R3         ;PATTERN COUNT
000220  012521          20$:    MOV     (R5)+,(R1)+     ;MOVE THE PATTERN INTO THE BUFFER
000222  005302                  DEC     R2              ;DECREMENT THE WORD COUNT
000224  001403                  BEQ     40$             ;BR IF DONE (WORD COUNT = 0)
000226  005303                  DEC     R3              ;DECREMENT THE PATTERN COUNT
000230  001372                  BNE     20$             ;BR IF MORE PATTERN
000232  000766                  BR      10$             ;CONTINUE DISTRIBUTING THE PATTERN
000234  104413          40$:    MTPS    #0              ;RESTORE THE REGISTERS
000236  000207                  RTS     PC              ;RETURN
```

```
                                        ;START THE COMMAND FOR THE DPB IN R0
                                        ;CALL:
                                        ;       MOV     #DPB,R0         ;DPB ADDRESS
                                        ;       JSR     PC,GODRIV
                                        ;       RETURN
                                        GODRIV: MOV     R0,-(SP)        ;SAVE R0
 020240  010046                                 MOV     R0,2$           ;CURRENT DPB ADDRESS
 020242  010037  020252                         MOV     R0,2$           ;CURRENT DPB ADDRESS
 020246  004037  041364                 1$:     JSR     R0,RP07         ;CALL THE DRIVE HANDLER
 020252  000000                         2$:     .WORD   0               ;DRIVE BLOCK ADDRESS GOES HERE
 020254  000000                                 HALT                    ;DRIVER REJECTED REQUEST
 020256  012600                                 MOV     (SP)+,R0        ;RESTORE R0
 020260  062760  000001  000052                 ADD     #1,$OPERC(R0)   ;INCREMENT THE OPERATION COUNT
 020266  005560  000054                         ADC     $OPERC+2(R0)
 020272  026060  000034  000012                 CMP     $PREVA+2(R0),$CYL(R0)   ;DID COMMAND REQUIRE A CYLINDER CHANGE ?
 020300  001412                                 BEQ     3$              ;BR IF NO
 020302  062760  000001  000046                 ADD     #1,$SEEKS(R0)   ;INCREMENT SEEK COUNT PER PASS
 020310  005560  000050                         ADC     $SEEKS+2(R0)    ;ADD ANY CARRY
 020314  062760  000001  000076                 ADD     #1,$STOTL(R0)   ;INCREMENT SEEK COUNT TOTAL
 020322  005560  000100                         ADC     $STOTL+2(R0)    ;ADD ANY CARRY
 020326  000207                         3$:     RTS     PC
```

```
                                        ;ROUTINE TO SETUP PARAMETERS FOR A SEQUENTIAL READ OR WRITE OF THE DISK
                                        ;CALL
                                        ;       MOV     #DPB,R0         ;DPB ADDRESS
                                        ;       MOV     #NN,$PACK(R0)   ;GET COMMAND NUMBER
                                        ;       JSR     PC,SEQPAR       ;CALL ROUTINE
                                        ;       RETURN
                                        ;
                                        ;WHERE 'NN' CAN BE ONE OF THE FOLLOW NUMBERS:
                                        ;       'WT' COMMAND= 2
                                        ;       'W'  COMMAND= -1
                                        ;       'T'  COMMAND= 0         (NOT USED IN ROUTINE)
                                        ;        R'  COMMAND= 1
                                        
020330  005760  000054          SEQPAR: TST     $OPERC+2(R0)    ;IS THIS THE FIRST OPERATION ?
020334  001010                          BNE     1$              ;BR IF NO
020336  005760  000052                  TST     $OPERC(R0)      ;IS THIS THE FIRST OPERATION ?
020342  001005                          BNE     1$              ;BR IF NO
020344  012704  000010                  MOV     #$SEC,R4        ;GET INDEX TO SECTOR ADDR STORAGE IN DPB
020350  004737  021412                  JSR     PC,CKLMTS       ;GO CHECK DISK ADDRESS LIMITS
020354  000453                          BR      3$              ;BR IF NOT AT END OF SEQUENTIAL ADDRESSING

020356  116060  000166  000027  1$:     MOVB    $RPCS1(R0),$PREV0(R0)   ;SAVE CURRENT PARAMETERS
020364  016060  000010  000032          MOV     $SEC(R0),$PREVA(R0)     ;SAVE PREVIOUS TRACK/SECTOR ADDRESS
020372  016060  000012  000034          MOV     $CYL(R0),$PREVA+2(R0)   ;SAVE PREVIOUS CYLINDER ADDRESS
020400  016060  000174  000010          MOV     $RPDA(R0),$SEC(R0)      ;CURRENT SECTOR & TRACK ADDRESS
020406  016060  000222  000012          MOV     $RPDC(R0),$CYL(R0)      ;CURRENT CYLINDER ADDRESS

020414  012704  000010          2$:     MOV     #$SEC,R4        ;GET INDEX TO SECTOR ADDR STORAGE IN DPB
020420  004737  021412                  JSR     PC,CKLMTS       ;GO CHECK DISK ADDRESS LIMITS
020424  000427                          BR      3$              ;BR IF NOT AT END OF SEQUENTIAL ADDRESSING

020426  116060  000146  000010          MOVB    MINSEC(R0),$SEC(R0)     ;RESET SECTOR ADDRESS
020434  116060  000142  000011          MOVB    MINTRK(R0),$TRK(R0)     ;RESET TRACK ADDRESS
020442  016060  000136  000012          MOV     MINCYL(R0),$CYL(R0)     ;RESET CYLINDER ADDRESS
020450  112760  000004  000024          MOVB    #4,$CODE(R0)    ;SET CODE TO READ DATA
020456  122760  177776  000026          CMPB    #-2,$PACK(R0)   ;WAS WRITE DATA IN PROGESS ?
020464  001473                          BEQ     8$              ;BR IF YES (START TESTING, 'T' COMMAND)
020466  004737  031500                  JSR     PC,$EOP         ;THIS IS THE END OF PASS
020472  032777  000020  160454          BIT     #SW04,@SWR      ;DO NOT DROP DRIVE AT EOT (SW04=1) ?
020500  001467                          BEQ     9$              ;BR IF NO
020502  000744                          BR      2$              ;MUST SURE ADDRESS IS NOT 'BSF' TRACK

020504  012704  000010          3$:     MOV     #$SEC,R4        ;GET INDEX TO SECTOR STORAGE
020510  013705  001466                  MOV     WRDCNT,R5       ;WORD COUNT IS MAXIMUM
020514  004737  021572                  JSR     PC,CHKWC        ;CHECK WORD COUNT FOR MAXCYL/MAXTRK
020520  010560  000020                  MOV     R5,$WRDL(R0)    ;GET WORD COUNT
020524  042760  000377  000020          BIC     #377,$WRDL(R0)  ;IS IT LESS THAN ONE SECTOR WORD COUNT ?
020532  001002                          BNE     4$              ;BR IF NO
020534  105260  000021                  INCB    $WRDL+1(R0)     ;SET TO ONE SECTOR
020540  016060  000020  000004  4$:     MOV     $WRDL(R0),$WCNT(R0)     ;STORE FOR 2'S COMPLEMENT WORD
020546  016060  000020  000120          MOV     $WRDL(R0),$HLDWC(R0)    ;HOLD WORD FOR 'RELBUF' ROUTINE
020554  005460  000004                  NEG     $WCNT(R0)       ;CHANGE WORD COUNT TO 2'S COMPLEMENT
020560  012760  000400  000022          MOV     #256.,$SSEC(R0) ;SECTOR SIZE FOR READ OR WRITE

020566  105760  000026                  TSTB    $PACK(R0)       ;'R' OR 'W' COMMAND FOR THIS DRIVE ?
020572  100407                          BMI     6$              ;BR IF WRITE
020574  112760  000004  000024  5$:     MOVB    #4,$CODE(R0)    ;CODE FOR READ DATA
```

MAIN PROGRAM

```
     58 020602  112760  000171  000002          MOVB    @RDDAT,$COMND(R0)       ;DRIVE CODE FOR OPERATION
     59 020610  000415                          BR      "$                      ;SET UP FOR EXIT
     61 020612  005737  001424          6$:     TST     RDONLY                  ;LOCKED IN "READ ONLY" MODE ?
     62 020616  001366                          BNE     5$                      ;BR IF YES
     63 020620  112760  000002  000024          MOVB    @2,$CODE(R0)            ;CODE FOR WRTDAT
     67 020626  112760  000161  000002          MOVB    @WRTDAT,$COMND(R0)      ;OP CODE
     68 020634  004737  021352                  JSR     PC,GETPAT               ;GET PATTERN CODE
     69 020640  110560  000030                  MOVB    R5,$PATTC(R0)           ;PATTERN CODE
     70 020644  012760  177777  000130  7$:     MOV     @ 1,$NEXT(R0)           ;SET PARAMETERS SELECTED INDICATOR
     71 020652  000207                          RTS     PC                      ;RETURN
     72
     73 020654  105060  000026          8$:     CLRB    $PACK(R0)               ;SET 'T' COMMAND FLAG IN DPB TABLE
     74 020660  005060  000130          9$:     CLR     $NEXT(R0)               ;CLEAR 'PARAMETER SELECTED  INDICATOR
     75 020664  005726                          TST     (SP)+                   ;CLEAR STACK LEVEL
     76 020666  000137  006340                  JMP     MAIN                    ;JUMP TO MAIN BACKGROUND LOOP
```

```
                                        ;GENERATE PARAMETERS FOR THE OPERATION
                                        ;CALL:
                                        ;       MOV     #DPB,R0         ;DPB ADDRESS
                                        ;       JSR     PC,GENPAR
                                        ;       RETURN

 020672  004737  037206          GENPAR: JSR     PC,$RAND        ;CYCLE THE RANDOM NUMBER GENERATOR
 020676  005737  001424                  TST     RDONLY          ;LOCKED IN "READ ONLY" MODE ?
 020702  001016                          BNE     1$              ;BR IF YES
 020704  032777  000001  160242          BIT     #SW0,@SWR       ;SEE IF SW0 SET
 020712  001012                          BNE     1$              ;BR IF SET - READ ONLY
 020714  012705  000010                  MOV     #8.,R5          ;READ/WRITE SELECTION DIVISOR
 020720  004737  032102                  JSR     PC,GETREM       ;GET SELECTION VALUE
 020724  020537  001502                  CMP     R5,RATIO        ;DETERMINE IF READ OR WRITE
 020730  103003                          BHIS    1$              ;BR IF READ
 020732  012705  000002                  MOV     #2,R5           ;SELECT WRITE DATA COMMAND
 020736  000407                          BR      2$              ;SELECT ADDRESS

 020740  013705  037306          1$:     MOV     $LONUM,R5       ;SELECT READ OPERATION CODE
 020744  000305                          SWAB    R5              ;SWAP BYTES IN R5
 020746  042705  177776                  BIC     #^C1,R5         ;MASK OUT ALL BUT BIT 0
 020752  062705  000004                  ADD     #4,R5           ;TABLE OFFSET FOR READ CODE
 020756  110560  000122          2$:     MOVB    R5,$NCODE(R0)   ;COMMAND SELECTION CODE TO CONTROL BLOCK
 020762  016060  000174  000124          MOV     $RPDA(R0),$NSEC(R0)     ;SECTOR AND TRACK
 020770  016060  000222  000126          MOV     $RPDC(R0),$NCYL(R0)     ;CYLINDER NUMBER

 020776  005737  001510          THEAD:  TST     RANDOM          ;ENABLE RANDOM ADDRESS SELECT ?
 021002  001427                          BEQ     RANCYL          ;YES
 021004  005760  000054                  TST     $OPERC+2(R0)    ;IS THIS THE FIRST OPERATION ?
 021010  001003                          BNE     1$              ;BR IF NO
 021012  005760  000052                  TST     $OPERC(R0)      ;IS THIS THE FIRST OPERATION ?
 021016  001405                          BEQ     2$              ;BR IF YES

 021020  012704  000124          1$:     MOV     #$NSEC,R4       ;GET INDEX TO SECTOR ADDR STORAGE IN DPB
 021024  004737  021412                  JSR     PC,CKLMTS       ;GO CHECK DISK ADDRESS LIMITS
 021030  000412                          BR      3$              ;BR IF NOT AT END OF SEQUENTIAL ADDRESSING
 021032  116060  000146  000124  2$:     MOVB    MINSEC(R0),$NSEC(R0)    ;RESET SECTOR ADDRESS
 021040  116060  000142  000125          MOVB    MINTRK(R0),$NTRK(R0)    ;RESET TRACK ADDRESS
 021046  016060  000136  000126          MOV     MINCYL(R0),$NCYL(R0)    ;RESET CYLINDER ADDRESS
 021054  000761                          BR      1$              ;RE-CHECK FOR 'BSF' TRACK
 021056  000137  021230          3$:     JMP     RANSIZ          ;GO CHECK FOR RANDOM WORD SIZE
```

```
                                        ;GENERATE A RANDOM CYLINDER ADDRESS BETWEEN VALUES  'MINCYL' & 'MAXCYL'
                                        ;
021062  016005  000134        RANCYL: MOV     MAXCYL(R0),R5   ;GET MAXIMUM CYLINDER ADDRESS
021066  026005  000136                CMP     MINCYL(R0),R5   ;'MINCYL' AND 'MAXCYL' THE SAME ?
021072  001407                        BEQ     1$              ;BR IF THEY ARE
021074  166005  000136                SUB     MINCYL(R0),R5   ;GET NUMBER OF ALLOWABLE CYLINDERS
021100  005205                        INC     R5              ;INCREMENT DIFFERENCE TO USE AS DIVISOR
021102  004737  032102                JSR     PC,GETREM       ;GET THE RANDOM AUGMENT
021106  066005  000136                ADD     MINCYL(R0),R5   ;NEW CYLINDER ADDRESS
021112  010560  000126        1$:     MOV     R5,$NCYL(R0)    ;STORE CYLINDER ADDRESS IN DPB
                                        ;
                                        ;GENERATE A RANDOM TRACK ADDRESS BETWEEN VALUES 'MINTRK' & 'MAXTRK'
                                        ;
021116  016005  000140        RANTRK: MOV     MAXTRK(R0),R5   ;GET MAXIMUM TRACK ADDRESS
021122  026005  000142                CMP     MINTRK(R0),R5   ;'MINTRK' AND 'MAXTRK' THE SAME ?
021126  001407                        BEQ     1$              ;BR IF THEY ARE
021130  166005  000142                SUB     MINTRK(R0),R5   ;GET NUMBER OF ALLOWABLE TRACKS
021134  005205                        INC     R5              ;INCREMENT DIFFERENCE TO USE AS DIVISOR
021136  004737  032102                JSR     PC,GETREM       ;GET THE RANDOM AUGMENT
021142  066005  000142                ADD     MINTRK(R0),R5   ;NEW TRACK ADDRESS
021146  110560  000125        1$:     MOVB    R5,$NTRK(R0)    ;STORE TRACK ADDRESS IN DPB
                                        ;
                                        ;GENERATE A RANDOM SECTOR ADDRESS BETWEEN VALUES 'MINSEC' & 'MAXSEC'
                                        ;
021152  016005  000144        RANSEC: MOV     MAXSEC(R0),R5   ;GET MAXIMUM SECTOR ADDRESS
021156  026005  000146                CMP     MINSEC(R0),R5   ;'MINSEC' AND 'MAXSEC' THE SAME ?
021162  001407                        BEQ     1$              ;BR IF THEY ARE
021164  166005  000146                SUB     MINSEC(R0),R5   ;GET NUMBER OF ALLOWABLE SECTORS
021170  005205                        INC     R5              ;INCREMENT DIFFERENCE TO USE AS DIVISOR
021172  004737  032102                JSR     PC,GETREM       ;GET THE RANDOM AUGMENT
021176  066005  000146                ADD     MINSEC(R0),R5   ;NEW SECTOR ADDRESS
021202  110560  000124        1$:     MOVB    R5,$NSEC(R0)    ;STORE SECTOR ADDRESS IN DPB
                                        ;
                                        ;MAKE SURE ADDRESS JUST GENERATED IS NOT 'BAD SECTOR FILE'
                                        ;
021206  012704  000124                MOV     #$NSEC,R4       ;GET INDEX TO SECTOR ADDR STORAGE IN DPB
021212  004737  021412                JSR     PC,CKLMTS       ;GO CHECK DISK ADDRESS LIMITS
021216  000404                        BR      2$              ;BR IF NOT AT END OF SEQUENTIAL ADDRESSING
021220  004737  037206                JSR     PC,$RAND        ;CYCLE THE RANDOM NUMBER GENERATOR
021224  000137  021062                JMP     RANCYL          ;GO GENERATE NEW ADDRESS
021230                          2$:
```

```
                                        ;GENERATE A RANDOM BUFFER LENGTH BETWEEN 6 & THE VALUE IN 'WRDCNT'
                                        
021230  013705  001466          RANSIZ: MOV     WRDCNT,R5       ;GET MAX WORD COUNT
021234  005737  001500                  TST     RANDWC          ;SELECT A RANDOM WORD COUNT ?
021240  001011                          BNE     2$              ;BR IF NOT
021242  005205                          INC     R5              ;INCREMENT THE MAXIMUM WRD CNT
021244  004737  032102                  JSR     PC,GETREM       ;DIVIDE BY MAX VALUE
021250  020527  000006                  CMP     R5,#6           ;WORD COUNT LESS THAN 6 ?
021254  002003                          BGE     2$              ;BR IF NO
021256  004737  037206          1$:     JSR     PC,$RAND        ;CYCLE THE RANDOM NUMBER GENERATOR
021262  000762                          BR      RANSIZ
                                        
021264  012704  000124          2$:     MOV     #$NSEC,R4       ;GET INDEX TO SECTOR STORAGE
021270  004737  021572                  JSR     PC,CHKWC        ;SEE IF WORD COUNT IS TOO LARGE TO FIT
                                                                ;IN REMAINDER OF TRACK. IF SO, THEN ADJUST
                                                                ;WORD COUNT TO FIT ON TRACK.
021274  122760  000002  000122  3$:     CMPB    #2,$NCODE(R0)   ;WRITE OPERATION ?
021302  001005                          BNE     4$              ;BR IF NO
021304  042705  000377                  BIC     #377,R5         ;WRITTING PARTIAL SECTOR ?
021310  001002                          BNE     4$              ;BR IF NO, ELSE,
021312  012705  000400                  MOV     #256.,R5        ;WRITE AT LEAST ONE SECTOR
021316  010560  000020          4$:     MOV     R5,$WRDL(R0)    ;WORD COUNT
                                        
                                        ;GET A RANDOM PATTERN NUMBER
                                        
021322  122760  000002  000122  RANPAT: CMPB    #2,$NCODE(R0)   ;WRITE OPERATION ?
021330  001004                          BNE     RANXIT          ;BR IF NO
021332  004737  021352                  JSR     PC,GETPAT       ;GET PATTERN CODE
021336  110560  000123                  MOVB    R5,$NPATC(R0)   ;MOVE PATTERN CODE TO CONTROL BLOCK
021342  012760  177777  000130  RANXIT: MOV     #-1,$NEXT(R0)   ;SET PARAMETERS SELECTED INDICATOR
021350  000207                          RTS     PC              ;RETURN
                                        
                                        ;ROUTINE TO SELECT A PATTERN
                                        
021352  012705  000020          GETPAT: MOV     #16.,R5         ;SELECT PATTERN
021356  005737  001476                  TST     PATTERN         ;ENABLE RANDOM PATTERN SELECTION ?
021362  001403                          BEQ     1$              ;YES
021364  013705  001476                  MOV     PATTERN,R5      ;USE INDEXED PATTERN
021370  000406                          BR      2$              ;NO
021372  004737  037206          1$:     JSR     PC,$RAND        ;CYCLE THE RANDOM NUMBER GENERATOR
021376  004737  032102                  JSR     PC,GETREM       ;GET CODE
021402  005705                          TST     R5              ;WAS PATTERN ZERO SELECTED ?
021404  001762                          BEQ     GETPAT          ;BR IF YES
021406  006305          2$:     ASL     R5              ;MAKE CODE INTO TABLE INDEX
021410  000207                  RTS     PC
```

```
                                        ;THIS ROUTINE IS USED TO CHECK ADDRESS LIMITS BEFORE THE NEXT COMMAND
                                        ;IS PERFORMED. ALSO, IT WILL CHECK FOR MAXIMUM ADDRESS LIMITS TO LOOK
                                        ;FOR AN END TO THE SEQUENTIAL ADDRESSING AND WILL MAKE SURE THE CURRENT
                                        ;ADDRESS IS NOT THE 'BAD SECTOR FILE'
                                        ;CALL:
                                        ;       MOV     @DPB,R0         ;DPB ADDRESS
                                        ;       MOV     @POINTER,R4     ;POINTER TO SECTOR STORAGE (0SEC OR 0NSEC) IN DPB
                                        ;       JSR     PC,CKLMTS       ;CALL ADDRESS LIMITS ROUTINE
                                        ;       BR      ???             ;RETURN HERE IF NOT END OF SEQUENTIAL ADDRESSING
                                        ;       ------                  ;ELSE, RETURN HERE TO RESET DISK ADDRESS
                                        ;
                                        ;R0 - DPB ADDRESS BEFORE CALLING THE ROUTINE
                                        ;R4 - POINTER TO SECTOR STORAGE BEFORE CALLING THE ROUTINE

021412  060004                 CKLMTS: ADD     R0,R4           ;POINT TO SECTOR STORAGE POINT IN DPB
021414  026460  000002  000136         CMP     2(R4),MINCYL(R0)        ;IS CYLINDER ADDRESS BELOW MIN. ?
021422  002003                         BGE     1$              ;BR IF NO
021424  016064  000136  000002         MOV     MINCYL(R0),2(R4)        ;RESET CYLINDER TO MIN.
021432  126460  000001  000142 1$:     CMPB    1(R4),MINTRK(R0)        ;IS TRACK ADDRESS BELOW MIN. ?
021440  002003                         BGE     2$              ;BR IF NO
021442  116064  000142  000001         MOVB    MINTRK(R0),1(R4)        ;RESET TRACK TO MIN.
021450  121460  000146         2$:     CMPB    (R4),MINSEC(R0) ;IS SECTOR ADDRESS BELOW MIN. ?
021454  002002                         BGE     3$              ;BR IF NO
021456  116014  000146                 MOVB    MINSEC(R0),(R4) ;RESET SECTOR TO MIN.

                                        ;LOOK FOR MAXIMUM LIMITS, FOR END OF SEQUENTIAL ADDRESSING AND
                                        ;ALSO CHECK THAT ADDRESS IS NOT 'BAD SECTOR FILE'

021462  121460  000144         3$:     CMPB    (R4),MAXSEC(R0) ;IS SECTOR ADDRESS AT MAXIMUM ?
021466  003404                         BLE     5$              ;BR IF NO
021470  116014  000146                 MOVB    MINSEC(R0),(R4) ;RESET SECTOR ADDRESS
021474  105264  000001         4$:     INCB    1(R4)           ;INCREMENT TO NEXT TRACK ADDRESS
021500  126460  000001  000140 5$:     CMPB    1(R4),MAXTRK(R0)        ;IS TRACK ADDRESS OVER MAXIMUM ?
021506  003407                         BLE     6$              ;BR IF NO
021510  116014  000146                 MOVB    MINSEC(R0),(R4) ;RESET SECTOR ADDRESS
021514  116064  000142  000001         MOVB    MINTRK(R0),1(R4)        ;RESET TRACK ADDRESS
021522  005264  000002                 INC     2(R4)           ;INCREMENT TO NEXT CYLINDER ADDRESS
021526  026460  000002  000134 6$:     CMP     2(R4),MAXCYL(R0)        ;IS CYLINDER ADDRESS OVER MAXIMUM ?
021534  003403                         BLE     7$              ;BR IF NO
021536  062716  000002                 ADD     #2,(SP)         ;ADJUST RETURN TO RESET DISK ADDRESS PARAMETERS
021542  000412                         BR      8$

                                        ;CHECK NOT TO READ OR WRITE BAD SECTOR TRACK
021544  016046  000156         7$:     MOV     FE1(R0),-(SP)   ;GET 1ST FE CYLINDER (LAST CYL+1)
021550  005316                         DEC     (SP)            ;LOOK AT LAST USER CYLINDER
021552  026426  000002                 CMP     2(R4),(SP)+     ;ARE WE ON LAST USER CYLINDER ?
021556  001004                         BNE     8$              ;BR IF NO
021560  126460  000001  000154         CMPB    1(R4),TRKLMT(P0)        ;IS THIS THE BAD SECTOR TRACK ?
021566  001742                         BEQ     4$              ;BR IF YES
021570  000207                 8$:     RTS     PC              ;RETURN
```

```
                                        ;THIS ROUTINE IS USED TO CALCULATE AND CHECK THE WORD COUNT FOR THE
                                        ;DRIVE THAT IS TO DO A DATA TRANSFER ON THE MAXIMUM TRACK OF THE MAXIMUM
                                        ;CYLINDER. IF THE CALCULATED MAXIMUM WORD COUNT, EXCEEDS THE DESIRED WORD
                                        ;COUNT (CONTENTS OF R5), THEN THE DESIRED WORD COUNT IS CHANGED, SO THAT
                                        ;THE WORD COUNT WILL NOT CAUSE A TRACK OVERFLOW DURING THE TRANSFER.
                                        ;CALL:
                                        ;       MOV     #DPB,R0         ;DPB ADDRESS
                                        ;       MOV     #POINTER,R4     ;POINTER TO SECTOR STORAGE (#SEC OR #NSEC) IN DPB
                                        ;       JSR     PC,CHKWC        ;CALL CHECK WORD COUNT ROUTINE
                                        ;       RETURN                  ;RETURN WITH R5 CONTAINING THE DESIRED WORD COUNT
                                        ;
                                        ;R0 = DPB ADDRESS BEFORE CALLING THE ROUTINE
                                        ;R4 = POINTER TO SECTOR STORAGE BEFORE CALLING THE ROUTINE
                                        ;R5 = DESIRED WORD COUNT BEFORE CALLING THE ROUTINE

021572  060004                 CHKWC: ADD     R0,R4           ;POINT TO SECTOR STORAGE POINT IN DPB
021574  105760  000146                TSTB    MINSEC(R0)      ;ALLOW SPIRAL RD/WRT ?
021600  001023                        BNE     2$              ;BR IF NO
021602  126060  000144  000152        CMPB    MAXSEC(R0),SECLMT(R0)   ;ALLOW SPIRAL RD/WRT ?
021610  001017                        BNE     2$              ;BR IF NO
021612  105760  000142                TSTB    MINTRK(R0)      ;ALLOW SPIRAL RD/WRT ?
021616  001010                        BNE     1$              ;BR IF NO
021620  126060  000140  000154        CMPB    MAXTRK(R0),TRKLMT(R0)   ;ALLOW SPIRAL RD/WRT ?
021626  001004                        BNE     1$              ;BR IF NO
                                        ;WHEN SPIRAL RD/WRT IS ALLOWED, THEN CHECK
                                        ;TO MAKE SURE YOU DO NOT SPIRAL OVER MAXIMUM
                                        ;TRACK ON MAXIMUM CYLINDER
021630  026064  000134  000002        CMP     MAXCYL(R0),2(R4)        ;ON MAXIMUM CYLINDER ?
021636  001022                        BNE     4$              ;BR IF NO
021640  126064  000140  000001  1$:   CMPB    MAXTRK(R0),1(R4)        ;ON MAXIMUM TRACK ?
021646  001016                        BNE     4$              ;BR IF NO
                                        ;ADJUST WORD COUNT TO FIT ON REMAINDER OF TRACK
021650  111404                 2$:    MOVB    (R4),R4         ;GET STARTING SECTOR (#SEC OR #NSEC) ADDRESS
021652  016046  000144                MOV     MAXSEC(R0),-(SP)        ;GET MAXIMUM SECTOR
021656  160416                        SUB     R4,(SP)         ;GET NUMBER SECTORS TO BE XFERD
021660  005004                        CLR     R4              ;CLEAR R4
021662  062704  000400          3$:   ADD     #256.,R4        ;ADD 1 SECTOR OF WORDS TO R4
021666  005316                        DEC     (SP)            ;DONE ALL SECTORS YET ?
021670  002374                        BGE     3$              ;BR IF NO
021672  005726                        TST     (SP)+           ;RESTORE STACK
021674  020504                        CMP     R5,R4           ;TOO MANY WORDS FOR TRACK ?
021676  003420                        BLE     5$              ;BR IF NO
021700  010405                        MOV     R4,R5           ;YES, CHANGE WORD COUNT
021702  000416                        BR      5$

021704  016046  000156          4$:   MOV     FE1(R0),-(SP)   ;GET 1ST FE CYLINDER (LAST CYL+1)
021710  005316                        DEC     (SP)            ;LOOK AT LAST USER CYLINDER
021712  026426  000002                CMP     2(R4),(SP)+     ;ARE WE ON LAST USER CYLINDER ?
021716  001010                        BNE     5$              ;BR IF NO
021720  016046  000154                MOV     TRKLMT(R0),-(SP)        ;GET LAST TRACK
021724  005316                        DEC     (SP)            ;LOOK AT NEXT TO LAST TRACK
021726  005046                        CLR     -(SP)           ;PUSH STACK
021730  116416  000001                MOVB    1(R4),(SP)      ;GET CURRENT TRACK
021734  022626                        CMP     (SP)+,(SP)+     ;IS IT TRACK BEFORE BAD SECTOR TRACK ?
021736  001744                        BEQ     2$              ;BR IF YES (DON'T ALLOW SPIRAL TO BAD SEC TRK)
021740  000207                 5$:    RTS     PC
```

```
                                        ;ROUTINE TO GET THE PREVIOUSLY SELECTED PARAMETER VALUES
                                        ;CALL:
                                        ;       MOV     #DPB,R0         ;DPB ADDRESS
                                        ;       JSR     PC,GENPAR       ;GENERATE THE PARAMETERS
                                        ;       JSR     PC,LOOPAR       ;LOAD THE PARAMETERS JUST GENERATED
                                        ;       RETURN

021742  010546                LOOPAR: MOV     R5,-(SP)        ;SAVE R5
021744  105760  000026                TSTB    DPACK(R0)       ;'T' COMMAND FOR THIS DRIVE ?
021750  001127                        BNE     61              ;BR IF NO
021752  116060  000166  000027        MOVB    DRPCS1(R0),DPREVO(R0)   ;SAVE CURRENT PARAMETERS
021760  142760  177701  000027        BICB    #^C76,DPREVO(R0)        ;STRIP GO,AND IE BITS
021766  132760  000006  000122        BITB    #6,DMCODE(R0)   ;SEE IF NEXT OPERATION IS READ OR WRITE
021774  001007                        BNE     10              ;BR IF EITHER
021776  016060  000012  000034        MOV     DCYL(R0),DPREVA+2(R0)   ;SAVE STARTING CYLINDER
022004  016060  000010  000032        MOV     DSEC(R0),DPREVA(R0)     ;SAVE STARTING SECTOR AND TRACK
022012  000410                        BR      20
022014  004737  024444        10:     JSR     PC,READDR       ;GET THE DECREMENTED SECTOR AND TRACK ADDRESSES
022020  012660  000034                MOV     (SP)+,DPREVA+2(R0)      ;CYLINDER ADDRESS
022024  112660  000033                MOVB    (SP)+,DPREVA+1(R0)      ;TRACK ADDRESS
022030  112660  000032                MOVB    (SP)+,DPREVA(R0)        ;SECTOR ADDRESS

022034  005337  022256        20:     DEC     ITCNT+2         ;REPEAT THIS COMMAND AGAIN ?
022040  002007                        BGE     30              ;BR IF YES
022042  013737  022254  022256        MOV     ITCNT,ITCNT+2   ;RESTORE ITERATION COUNT
022050  032777  000100  157076        BIT     #SW06,@#SWR     ;LOOP ON PREVIOUS VALUES (SW6=1) ?
022056  001413                        BEQ     40              ;BR IF NO
022060  122760  000151  000002  30:     CMPB    #WCHD,DCOMND(R0)        ;IS COMMAND A WRITE CHECK DATA ?
022066  001060                        BNE     61              ;BR IF NO
022070  112760  000002  000024        MOVB    #2,DCODE(R0)    ;CODE FOR WRTDAT
022076  112760  000161  000002        MOVB    #WRTDAT,DCOMND(R0)      ;PUT WRITE DATA COMMAND BACK FOR LOOP
022104  000451                        BR      61

022106  116060  000122  000024  40:     MOVB    DMCODE(R0),DCODE(R0)    ;LOGICAL CODE FOR OPERATION
022114  116005  000122                MOVB    DMCODE(R0),R5   ;LOAD R5 FOR USE AS TABLE INDEX
022120  116560  002076  000002        MOVB    COMTBL(R5),DCOMND(R0)   ;COMMAND CODE
022126  122760  000151  000002        CMPB    #WCHD,DCOMND(R0)        ;IS NEW COMMAND A WRITE CHECK DATA ?
022134  001012                        BNE     50              ;BR IF NO
022136  122760  000060  000027        CMPB    #60,DPREVO(R0)  ;WAS PREVIOUS COMMAND A WRITE DATA ?
022144  001431                        BEQ     61              ;BR IF YES
022146  112760  000171  000002        MOVB    #RDDAT,DCOMND(R0)       ;CHANGE WRITE CHECK TO READ DATA COMMAND
022154  112760  000004  000024        MOVB    #4,DCODE(R0)    ;CODE NUMBER CHANGED TO READ DATA

022162  116060  000125  000030  50:     MOVB    DMPATC(R0),DPATTC(R0)   ;PATTERN CODE
022170  016060  000124  000010        MOV     DMSEC(R0),DSEC(R0)      ;TRACK AND SECTOR ADDRESSES
022176  016060  000126  000012        MOV     DMCYL(R0),DCYL(R0)      ;CYLINDER ADDRESS
022204  012760  000400  000022        MOV     #256.,DSSEC(R0) ;INITIAL VALUE OF SECTOR SIZE
022212  132760  000001  000024        BITB    #1,DCODE(R0)    ;HEADER OPERATION ?
022220  001403                        BEQ     60              ;BR IF NOT
022222  062760  000002  000022        ADD     #2,DSSEC(R0)    ;ADD HEADER SIZE
022230  016060  000020  000004  60:     MOV     DWROL(R0),DWCNT(R0)     ;GET WORD COUNT AND
022236  016060  000020  000120        MOV     DWROL(R0),DWLDWC(R0)    ;HOLD WORD FOR 'RELBUF' ROUTINE
022244  005460  000004                NEG     DWCNT(R0)       ;MAKE IT 2'S COMPLEMENT
022250  012605                        MOV     (SP)+,R5        ;RESTORE R5
022252  000207                        RTS     PC              ;RETURN

022254  000000  000000        ITCNT:  .WORD   0,0             ;'ITCNT' CONTAINS THE NUMBER OF TIMES TO
```

```
                                                        ;REPEAT THE COMMAND
                                                        ; ITCNT+2  IS THE COUNTER
                
                
                                ;ROUTINE TO COMPRESS A LIST
                                ;CALL:
                                ;       MOV     #ADDRS,R1       ;COMPRESS LIST STARTING AT THIS ADDRESS
                                ;       JSR     PC,CMPRES
                                ;       RETURN
                
000000 016111  000002           CMPRES: MOV     2(R1),(R1)      ;COMPRESS THE TABLE IN R1
000004 001402                           BEQ     10              ;BR WHEN ZERO FOUND
000006 005721                           TST     (R1)+           ;INCREMENT R1
000010 000773                           BR      CMPRES          ;CONTINUE COMPRESSING TABLE
000012 000207                   10:     RTS     PC              ;RETURN
```

```
                        .SBTTL  ERROR MESSAGE GENERATION ROUTINES

                        ;PRINT LINE 1 OF ERROR MESSAGE:
                        ; ERR-PDP 11

 000776  012777  000000  130652 LINE1: MOV     @#174.,ERRSAV           ;SWITCH 10 SET ?
 000702  001402                        BEQ     10$                     ;BR IF NOT
 000704  104401  001176                TYPE    ,ERRTLL                 ;TYPE THE ERRTLL
 000710  012777  020000  130636 10$:    MOV     #BIT13,ERRSAV           ;TURN BIT THROUGH (BASE-1) ?
 000716  001016                        BNE     20$                     ;BR IF YES
 000720  104401  001705                TYPE    ,CRLF                   ;CR-LF
 000724  104401  001703                TYPE    ,CRLF                   ;CR-LF
 000730  004737  000726                JSR     PC,DTIME                ;TYPE ELAPSED TIME
 000734  104401  004611                TYPE    ,@#MSG1                 ;TYPE 1 ERROR
 000740  000207                 20$:   RTS     PC                      ;RETURN & TYPE DESCRIPTION

                        ;PRINT LINE 2 OF ERROR MESSAGE
                        ;[illegible] ERROR • [illegible] ON:[illegible] ERROR • [illegible]
                        ;• [illegible] AT PASS [illegible]
                        ;[illegible]  [illegible]  [illegible]  [illegible]  [illegible]  [illegible]  [illegible]  [illegible]
                        ;[illegible]  [illegible]  [illegible]  [illegible]  [illegible]  [illegible]  [illegible]  STATUS
                        ;[illegible]  [illegible]  ([illegible] ONLY)
                        ;[illegible] [illegible] OR WORD COUNT NOT CONSISTENT
                        ;[illegible] • [illegible]  [illegible] • [illegible]
                        ;BUFFER ADR • [illegible]  WRD CNT • [illegible]  [illegible] [illegible] [illegible] • [illegible]

 000742                 LINE2:
 000742  010546                        MOV     R5,-(SP)                ;;PUSH R5 ON STACK
 000744  010446                        MOV     R4,-(SP)                ;;PUSH R4 ON STACK
 000746  010346                        MOV     R3,-(SP)                ;;PUSH R3 ON STACK
 000750  104414  001705                TYPE    ,CRLF                   ;CR-LF
 000754  012705  022502                MOV     #ERRTBL,R5              ;CLEAR MESSAGE ADDRESS STORAGE
 000760  005004                        CLR     R4                      ;[illegible] REGISTER
 000762  012737  000416  022502        MOV     @ERRPC,R0               ;ADDRESS OF [illegible] ERROR • • MSG
 000770  116000  000166                MOVB    ERRTBL(R0),R0           ;GET THE OPCODE
 000774  042700  177701                BIC     #^C76,R0                ;SAVE ONLY 5 SIGNIFICANT BITS
 001000  006737  000436                ASL     PC,10                   ;TYPE THE FIRST PREFIXLC
 001004  005737  022506                TST     S0                      ;SEE IF PREFIXLC ENTRY FOUND
 001010  001440                        BEQ     LINE2A                  ;BR IF NOT
 001012  012737  000436  022502        MOV     @ERRPC,R0               ;ADDRESS OF  PREVIOUS ORDER • • MSG
 001020  116000  000027                MOVB    ERRTBL(R0),R0           ;PREVIOUS OPERATION CODE
 001024  042700  177701                BIC     #^C76,R0                ;SAVE ONLY 5 SIGNIFICANT BITS
 001030  006737  022436                ASL     PC,10                   ;TYPE THE PREVIOUS PREFIXLC
 001034  000436                        BR      LINE2A                  ;CONTINUE
 001036  005005                 10$:   CLR     R5                      ;CLEAR THE TABLE INDEX
 001040  126700  000104         20$:   CMPB    R0,TBL(R5)              ;LOOK FOR THE OPCODE
 001044  001405                        BEQ     30$                     ;BR WHEN OPCODE COUNT = OPCODE
 001046  105765  000104                TSTB    TBL(R5)                 ;LOOK FOR END OF TABLE
 001052  100402                        BPL     30$                     ;BR IF END
 001054  005205                        INC     R5                      ;INCREMENT THE POINTER
 001056  000770                        BR      20$                     ;CONTINUE - NOT END OF TABLE
 001060  006305                 30$:   ASL     R5                      ;SHIFT INDEX
 001062  006305                        ASL     R5                      ;SHIFT THE INDEX
 001064  006305                        ASL     R5                      ;SHIFT THE INDEX
 001066  012737  002124  022506        MOV     #TBL,S0                 ;ADDRESS OF ASCII TEXT TABLE
 002474  060537  022506                ADD     R5,S0                   ;ADD THE INDEX
```

```
 95 022700  104414                       DISPLY                  ;TYPE IT
 96 022702  000000               00:     .WORD   0               ;ADDRESS OF 'PRESENT' OR 'PREVIOUS' MESSAGE
 97 022704  104414                       DISPLY                  ;TYPE THE OPERATION PREVIOUS
 98 022706  000000               50:     .WORD   0               ;ADDRESS OF MESSAGE
 99 022710  000207                       RTS     PC              ;RETURN TO MAIN ROUTINE
100
 61 022712  005737  001520       LINE3A: TST     BADSEC          ;PRINT THE BAD SECTOR LINE ?
 62 022716  001000                       BEQ     LINE3B          ;JMP IF NOT
 63 022720  104414  001205               DISPLY  .CRLF           ;CR LF
 64 022724  104414  054517               DISPLY  .ASBAD          ;ERROR ADDRESS DEFINED AS BAD AREA
 65 022730  104414  001205       LINE3B: DISPLY  .CRLF           ;CR LF
 66 022734  104414  054270               DISPLY  .DH10           ;STANDARD QDRIVE REGISTER HEADER
 67 022740  011700  001520               MOV     DRIVE,-(SP)     ;PUT THE DRIVE NUMBER ON THE STACK
 68 022744  004737  024366               JSR     PC,TYPEDEC      ;TYPE IT
 69 022750  104414  054410               DISPLY  .DH11           ;TYPE A BLANK
 70 022754  012705  054540               MOV     #DH10,R5        ;REGISTER INDEXES
 71 022760  004737  022742               JSR     PC,30           ;PRINT THE REGISTERS
 72
 73 022764  032777  000000  176362       BIT     #SWR5,@SWR      ;PRINT THE OPTIONAL REGISTERS ?
 74 022772  001001                       BNE     10              ;JMP IF NOT
 75 022774  104414  054505               DISPLY  .DH15           ;STANDARD QDRIVE REGISTER HEADER
 76 023000  012705  054550               MOV     #DH15,R5        ;SECOND DATA LINE
 77 023004  004737  022742               JSR     PC,30           ;PRINT THEM
 78
 79 023010  104414  054000               DISPLY  .DH16           ;STANDARD QDRIVE REGISTER HEADER
 80 023014  012705  054572               MOV     #DH16,R5        ;THIRD DATA LINE
 81 023020  004737  022742               JSR     PC,30           ;PRINT THE REGISTERS
 82
 83 023024  011700  001730               MOV     @CPUOP,-(SP)    ;CHECK THE CPU (DH) TYPE
 84 023030  042716  017777               BIC     #C17777,(SP)    ;ISOLATE THE CPU BITS
 85 023034  022726  010000               CMP     #10000,(SP)+    ;SEE IF DHB0
 86 023040  001006                       BNE     10              ;JMP IF NO
 87 023042  104414  054444               DISPLY  .DH17           ;STANDARD QDRIVE REGISTER HEADER
 88 023046  012705  054410               MOV     #DH17,R5        ;OPTIONAL FOURTH DATA LINE
 89 023052  004737  022742               JSR     PC,30           ;PRINT THE REGISTERS
 90 023056  032760  040100  000016  10:  BIT     #B16,STATUS(R0) ;DATA ERROR ?
 91 023064  001422                       BEQ     20              ;JMP IF NOT
 92 023066  016046  000020               MOV     BUFCL(R0),-(SP) ;TRANSFER SIZE
 93 023072  066016  000170               ADD     WRDC(R0),(SP)   ;ADD REMAINING WORD COUNT
 94 023076  006316                       ASL     (SP)            ;CONVERT TO AN BYTE INCREMENT
 95 023100  066016  000006               ADD     BUF(R0),(SP)    ;BUFFER STARTING ADDRESS
 96 023104  022660  000172               CMP     (SP)+,BUFBA(R0) ;CORRECT BUFFER ADDRESS ?
 97 022710  001410                       BEQ     20              ;JMP IF YES
 98 022712  104414  055306               DISPLY  .EM06           ;'BUS ADDRESS AND WORD COUNT ARE NOT CONSISTENT'
 99 022716  104414  001205               DISPLY  .CRLF           ;CR LF
100 022722  004737  023034               JSR     PC,LINE3D       ;PRINT LINE 3D OF ERROR MESSAGE
101 022726  004737  023446               JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
102 022732                       20:
    022732  012605                       MOV     (SP)+,R5        ;;POP STACK INTO R5
    022734  012604                       MOV     (SP)+,R4        ;;POP STACK INTO R4
    022736  012603                       MOV     (SP)+,R3        ;;POP STACK INTO R3
103 022740  000207                       RTS     PC              ;RETURN TO ERROR PROCESSING ROUTINE
104
105 022742  012546                   30: MOV     (R5)+,-(SP)     ;PUT THE REGISTER INDEX ON THE STACK
106 022744  060016                       ADD     R0,(SP)         ;ADD DRIVE'S TABLE ADDRESS
107 022746  017646  000000               MOV     @(SP),-(SP)     ;VALUE
108 022752  004737  024366               JSR     PC,LINOCT       ;TYPE IT
```

```
                                        ;CONNECT THE STATUS REGISTER
                                        ;TYPE 2 BLANKS
                                        ;AT END OF LINE ?
                                        ;NO IF NOT
                                        ;CR LF
                                        ;AND RETURN

                        ;TO WT LINE 3 OF ERROR MESSAGE
                        ; ERROR AT CCC TT SS   PRESS ANY • CCC TT SS·

023770  104416  056718  LINE3:  DISPLY  .LIN03          ;LINE 3 ENTRANCE
024002  000517                  BR      LIN3.1          ;FINISH PRINTOUT

                        ;PRINT LINE 3A OF ERROR MESSAGE
                        ; START CYL• CCC  END CYL• CCC

024004  104416  056751  LINE3A: DISPLY  .LIN03          ;LINE 3A ENTRANCE
024010  000514                  BR      LIN3.1          ;FINISH ERROR LINE

                        ;PRINT LINE 3B OF ERROR MESSAGE
                        ; START CYL• CCC  END CYL• CCC  ACTUAL CYL• CCC·

024012  004737  023150  LINE3B: JSR     PC,LIN3.3       ;LINE 3B ENTRANCE
024016  104416  001205          DISPLY  ,$CRLF
024022  000207                  RTS     PC

                        ;PRINT LINE 3C OF ERROR MESSAGE
                        ;'START CYL• CCC  END CYL• CCC  ACTUAL CYL• CCC  TRK• TT

024026  004737  023550  LINE3C: JSR     PC,LIN3.3       ;LINE 3C ENTRANCE
024032  000137  023402          JMP     LIN3.4          ;FINISH MESSAGE

                        ;PRINT LINE 3D OF ERROR MESSAGE
                        ;'RPBA • XXXXXX   RPWC • XXXXXX·

024036  032777  000040  156112 LINE3D: BIT     @SWR05,#BSWR    ;SWITCH 5 SET ?
024044  001416                  BEQ     10              ;BR IF IT IS
024046  104414  055070          DISPLY  .LIN03          ;'RPBA • '
024052  016046  000172          MOV     @RPBA(R0),-(SP) ;BUFFER ADDR REG CONTENTS
024056  004737  024366          JSR     PC,LINOCT       ;CONVERT TO OCTAL AND TYPE IT
024062  104414  055077          DISPLY  .LIN03          ;'  RPWC • '
024066  016046  000170          MOV     @RPWC(R0),-(SP) ;WORD COUNT REGISTER CONTENTS
024072  004737  024366          JSR     PC,LINOCT       ;CONVERT TO OCTAL AND TYPE IT
024076  104414  001205          DISPLY  ,$CRLF
024102  000207          10:     RTS     PC

                        ;PRINT LINE 3E OF ERROR MESSAGE
                        ;'START CYL • CCC   START TRK • TT   START SEC • SS·

024104  104414  054772 LINE3E: DISPLY  .LIN03          ;'START CYL • '
024110  016046  000012          MOV     @CYL(R0),-(SP)  ;MOVE CYL TO STACK
024114  004737  024420          JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
024120  104414  056610          DISPLY  .BLNKS2         ;TYPE 2 BLANKS
024124  104414  055110          DISPLY  .LIN13          ;'START TRK • '
024130  005046                  CLR     -(SP)           ;CLEAR STACK
024132  116016  000011          MOVB    @TRK(R0),(SP)   ;TRACK TO STACK
024136  004737  024420          JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
```

```
                                        DISPLY  .BLNKS2          ;TYPE 2 BLANKS
                                        DISPLY  .LINSS3          ;'START SEC = '
                                        CLR      (SP)            ;CLEAR STACK
                                        MOVB    $SEC(R0),(SP)    ;SECTOR ADDR TO STACK
                                        JSR     PC,LINDEC        ;TYPE IT IN DECIMAL
                                        DISPLY  ,$CRLF
                                        RTS     PC

                                ;PRINT LINE 3F OF ERROR MESSAGE
                                ;'RPDA = XXXXXX   RPCA = XXXXXX'

                        LINE3F: BIT     #SW5,@SWR        ;SWITCH 5 SET ?
                                BEQ     1$               ;BR IF NOT
                                DISPLY  .LINDA3          ;'RPDA = '
                                MOV     $RPDA(R0),-(SP) ;PUT SECTOR/TRACK ADDRESS ON THE STACK
                                JSR     PC,LINOCT        ;TYPE IT
                                DISPLY  .BLNKS2          ;TYPE 2 BLANKS
                                DISPLY  .LINCA3          ;' RPDC = '
                                MOV     $RPDC(R0),-(SP) ;PUT DESIRED CYLINDER ADDRESS ON THE STACK
                                JSR     PC,LINOCT        ;TYPE IT
                                DISPLY  ,$CRLF
                        1$:     RTS     PC

                                ;'CCC TT SS   PREV ADR = CCC TT SS'

                        LIN3.1: JSR     PC,READDR        ;DECREMENT TRACK AND SECTOR ADDRESS
                                JSR     PC,LINDEC        ;TYPE IT IN DECIMAL
                                DISPLY  ,T               ;PRINT 'T'
                                JSR     PC,LINDEC        ;TYPE TRACK IN DECIMAL
                                DISPLY  ,S               ;PRINT 'S'
                                JSR     PC,LINDEC        ;TYPE SECTOR ADDRESS
                                DISPLY  ,LINP3           ;PRINT 'PREV ADDR'
                                MOV     $PREVA+2(R0),-(SP) ;PREVIOUS CYLINDER
                                JSR     PC,LINDEC        ;TYPE IT IN DECIMAL
                                DISPLY  ,T               ;PRINT 'T'
                                CLR     -(SP)            ;MAKE ROOM ON THE STACK
                                MOVB    $PREVA+1(R0),(SP)  ;PREVIOUS TRACK ADDRESS
                                JSR     PC,LINDEC        ;TYPE IT IN DECIMAL
                                DISPLY  ,S               ;PRINT 'S'
                                CLR     -(SP)            ;MAKE ROOM ON THE STACK
                                MOVB    $PREVA(R0),(SP)   ;PREVIOUS SECTOR DDRESS
                                JSR     PC,LINDEC        ;TYPE IT IN DECIMAL
                                DISPLY  ,$CRLF
                                RTS     PC

                        ;'START CYL= CCC  END CYL= CCC'

                        LIN3.3: DISPLY  ,LINS3           ;LINE '3B & 3C' ENTRANCE
                                MOV     $PREVA+2(R0),-(SP) ;PREVIOUS CYLINDER
                                JSR     PC,LINDEC        ;TYPE IT IN DECIMAL
                                DISPLY  ,LINEN3          ;PRINT ' END CYL'
                                MOV     $RPDC(R0),-(SP) ;PRESENT CYLINDER
                                JSR     PC,LINDEC        ;TYPE IT IN DECIMAL
                                RTS     PC

                        ;'ACTUAL CYL= CCC  TRK= TT'
```

```
023402  104414  055022          LIN3.4: DISPLY  ,LINA3           ;PRINT 'ACTUAL CYL= '
023406  013746  063364                  MOV     CYLNDR, (SP)      ;ACTUAL CYLINDER
023412  042716  010000                  BIC     #BIT12,(SP)       ;CLEAR THE FORMAT BIT
023416  004737  024420                  JSR     PC,LINDEC         ;TYPE IT IN DECIMAL
023422  104414  055041                  DISPLY  ,LINT3            ;PRINT 'TRK= '
023426  005046                          CLR     (SP)              ;CLEAR STACK WORD
023430  116016  000175                  MOVB    $RPDA+1(R0),(SP)  ;PUT TRACK ON STACK
023434  004737  024420                  JSR     PC,LINDEC         ;TYPE IT IN DECIMAL
023440  104414  001203                  DISPLY  ,$CRLF
023444  000207                          RTS     PC

                                ;PRINT LINE 4 OF ERROR MESSAGE
                                ;'BUFFER ADR= XXXXXX  WRD CNT= XXXX  NMBR WRDS XFRD= XXX'

023446  032760  000100  000016  LINE4:  BIT     #BIT06,$TATUS(R0)  ;DATA ERROR ?
023454  001427                          BEQ     1$                ;BR IF NOT
023456  104414  055140                  DISPLY  ,LINM4            ;PRINT 'BUFFER ADR= '
023462  016046  000006                  MOV     $BUF(R0),-(SP)    ;BUFFER ADDR ON STACK
023466  004737  024366                  JSR     PC,LINOCT         ;CONVERT TO OCTAL & PRINT
023472  104414  055156                  DISPLY  ,LINS4            ;PRINT 'WRD CNT= '
023476  016046  000020                  MOV     $WRDL(R0),-(SP)   ;BUFFER SIZE
023502  004737  024420                  JSR     PC,LINDEC         ;TYPE IT IN DECIMAL
023506  104414  055172                  DISPLY  ,LINX4            ;'  NMBR WRDS XFRD = '
023512  016046  000172                  MOV     $RPBA(R0),-(SP)   ;VALUE IN BUFFER ADDR REGISTER
023516  166016  000006                  SUB     $BUF(R0),(SP)     ;SUBTRACT STARTING ADDRESS
023522  006216                          ASR     (SP)              ;CONVERT INTO A WORD COUNT
023524  004737  024420                  JSR     PC,LINDEC         ;TYPE IT IN DECIMAL
023530  104414  001203                  DISPLY  ,$CRLF            ;CR LF
023534  000207                  1$:     RTS     PC                ;RETURN

                                ;PRINT LINE 5 OF ERROR MESSAGE
                                ;'EXPCTD DATA= XXXXXX  RECEVD DATA= XXXXXX  WORD POS= XXX'

023536  104414  055215          LINE5:  DISPLY  ,LIND5            ;PRINT 'EXPCTD DATA'
023542  162760  000002  000172          SUB     #2,$RPBA(R0)      ;BACK THE ADDRESS UP
023550  013746  001234                  MOV     $CPUOP,-(SP)      ;CHECK THE CPU (RH) TYPE
023554  042716  003777                  BIC     #^C174000,(SP)    ;LEAVE THE CPU BITS
023560  022726  030000                  CMP     #30000,(SP)+      ;SEE IF RH70
023564  001012                          BNE     1$                ;BR IF NO
023566  162760  000004  000172          SUB     #4,$RPBA(R0)      ;BACKUP THE BUFFER POINTER
023574  032760  004000  000240          BIT     #BIT11,$RPCS3(R0)       ;SEE WHICH WORD HALF DIDN'T COMPARE
023602  001403                          BEQ     1$                ;IF EQ, EVEN HALF DIDN'T COMPARE
023604  162760  000002  000172          SUB     #2,$RPBA(R0)      ;BACKUP THE BUFFER POINTER AGAIN
023612  017046  000172          1$:     MOV     @$RPBA(R0),-(SP)        ;'EXPCTD' DATA - AT THE BUFFER LOCATION
023616  004737  024366                  JSR     PC,LINOCT         ;TYPE IT
023622  104414  055233                  DISPLY  ,LINB5            ;PRINT 'RECEVD DATA'
023626  016046  000210                  MOV     $RPDB(R0),-(SP)   ;RECEVD DATA FROM BUFFER
023632  004737  024366                  JSR     PC,LINOCT         ;TYPE IT
023636  016046  000170                  MOV     $RPWC(R0),-(SP)   ;WORD LENGTH ON STACK
023642  066016  000020                  ADD     $WRDL(R0),(SP)    ;MAKE INTO A POSITIVE NUMBER
023646  005046                          CLR     -(SP)             ;UPPER DIVIDEND TO ZERO
023650  016046  000022                  MOV     $SSEC(R0),-(SP)   ;SECTOR SIZE ON THE STACK
023654  004737  032126                  JSR     PC,$DIV           ;DIVIDE WORDS XFERED BY SECTOR SIZE
023660  012616                          MOV     (SP)+,(SP)        ;MOVE REMAINDER UP THE STACK
023662  005316                          DEC     (SP)              ;DECREMENT WORD POSITION BY 1
023664  032760  040000  000176          BIT     #BIT14,$RPCS2(R0)       ;IS 'WCE' SET ?
023672  001416                          BEQ     2$                ;BR IF NO
```

```
023674  013746  001234                  MOV     $CPUOP, (SP)     ;CHECK THE CPU (RH) TYPE
023700  042716  003777                  BIC     #^C174000,(SP)   ;LEAVE THE CPU BITS
023704  022726  030000                  CMP     #30000,(SP)+     ;SEE IF RH70
023710  001007                          BNE     2$               ;BR IF NO
023712  162716  000002                  SUB     #2,(SP)          ;SUBTRACT 2 FOR A DOUBLE WORD
023716  032760  004000  000240          BIT     #BIT11,$RPCS3(R0)       ;SEE WHICH WORD HALF DIDN'T COMPARE
023724  001401                          BEQ     2$               ;IF EQ, EVEN HALF DIDN'T COMPARE
023726  005316                          DEC     (SP)             ;SUBTRACT 1 FOR AN ODD WORD
023730  104414  055253          2$:     DISPLY  .LINP5           ;PRINT 'WORD POS
023734  004737  024420                  JSR     PC,LINDEC        ;TYPE THE POSITION
023740  104414  001203                  DISPLY  .$CRLF
023744  000207                          RTS     PC

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

023746                          LINE5A:
023746  104414  055270                  DISPLY  .LINS5           ;'HEADER CONTENTS OF ERROR SECTOR'
023752  013746  063364                  MOV     CYLNDR, (SP)     ;HEADER POSITION
023756  004737  024366                  JSR     PC,LINOCT        ;TYPE IT
023762  104414  056610                  DISPLY  .BLNKS2          ;TYPE 2 BLANKS
023766  013746  063366                  MOV     CYLNDR+2,-(SP)   ;HEADER POSITION +2
023772  004737  024366                  JSR     PC,LINOCT        ;TYPE IT
023776  104414  056610                  DISPLY  .BLNKS2          ;TYPE 2 BLANKS
024002  104414  056613                  DISPLY  .LINX5
024006  104414  001203                  DISPLY  .$CRLF
024012  000207                          RTS     PC

                                ;PRINT LINE 5B OF ERROR MESSAGE
                                ;'RPEC1 = XXXXXX   RPEC2 = XXXXXX'

024014  104414  055323          LINE5B: DISPLY  .LINEP5          ;'RPEC1 = '
024020  016046  000232                  MOV     $RPEC1(R0),-(SP) ;PUT REGISTER CONTENTS ON THE STACK
024024  004737  024366                  JSR     PC,LINOCT        ;TYPE IT
024030  104414  056610                  DISPLY  .BLNKS2          ;TYPE 2 BLANKS
024034  104414  055333                  DISPLY  .LINE05          ;' RPEC2 = '
024040  016046  000234                  MOV     $RPEC2(R0),-(SP) ;PUT REGISTER CONTENTS ON THE STACK
024044  004737  024366                  JSR     PC,LINOCT        ;TYPE IT
024050  104414  001203                  DISPLY  .$CRLF
024054  000207                          RTS     PC               ;RETURN

                                ;PRINT LINE 6 OF ERROR MESSAGE
                                ;'SECTOR IS ECC CORRECTABLE'

024056  104414  055345          LINE6:  DISPLY  .LINB6           ;PRINT 'SECTOR IS ECC CORRECTABLE '
024062  104414  001203                  DISPLY  .$CRLF
024066  000207                          RTS     PC

                                ;PRINT LINE 6A OF THE ERROR MESSAGE
                                ;'SECTOR READ CORRECTLY AFTER N RETRY(S)'

024070  104414  055400          LINE6A: DISPLY  .LINC6           ;PRINT 'SECTOR READ CORRECTLY AFTER N RETRY(S)'
024074  000406                          BR      LIN6.2           ;TYPE THE REST OF THE LINE

                                ;PRINT LINE 6C OF THE ERROR MESSAGE
                                ;'CORRECTED ON N RETRY(S)'
```

ERROR MESSAGE GENERATION ROUTINES

```
                                ;PRINT LINE 6C OF THE ERROR MESSAGE
                                ;'CORRECTED ON N RETRY(S)'
                                ;
024076  104414  055435  LINE6C: DISPLY  .LIN6          ;'CORRECTED ON N RETRY(S)
024102  000403                  BR      LIN6.2          ;TYPE THE REST OF THE LINE
                                ;
                                ;PRINT LINE 6D OF THE ERROR MESSAGE
                                ;'UNCORRECTABLE AFTER N RETRY(S)'
                                ;
024104  104414  055465  LINE6D: DISPLY  .LINU06         ;'UNCORRECTABLE AFTER N RETRY(S)'
024110  000400                  BR      LIN6.2          ;FINISH
                                ;
                                ;RETRY COUNT TYPEOUT
                                ;
024112  005046  LIN6.2: CLR     -(SP)           ;CLEAR STACK
024114  113716  001325          MOVB    RETRY+1,(SP)    ;RETRY COUNT
024120  004737  024420          JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
024124  104414  055453          DISPLY  .LINR6          ;' RETRY(S)'
024130  104414  001203          DISPLY  .$CRLF
024134  000207                  RTS     PC
                                ;
                                ;PRINT LINE 7 OF THE ERROR MESSAGE
                                ;'TOTALS; ERRORS:XXX  WRDS WRITN:XXXXX X10+6 WRDS READ:XXXXX X10+6'
                                ;
024136  104414  055574  LINE7:  DISPLY  .LIN7T           ;TOTALS; ERRORS
024142  016046  000102          MOV     $TOTAL(R0),(SP)         ;TO STACK
024146  004737  024420          JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
024152  104414  055615          DISPLY  .LIN7X          ;PRINT ' WRDS WRITN'
024156  012746  000056          MOV     @$WTOTL,-(SP)   ;ADDRESS OF LOW WORD ON STACK
024162  060016                  ADD     R0,(SP)
024164  004737  037404          JSR     PC,$D82D        ;CONVERT
024170  004737  032526          JSR     PC,$SUPRL       ;PRINT
024174  104414  057771          DISPLY  .PERIOD         ;TYPE '.'
024200  104414  057632          DISPLY  .MSGX10         ;TYPE ' X10+6'
024204  104414  055633          DISPLY  .LIN7R          ;PRINT ' WRDS READ'
024210  012746  000066          MOV     @$RTOTL,-(SP)   ;LOW WORD ADDRESS
024214  060016                  ADD     R0,(SP)
024216  004737  037404          JSR     PC,$D82D        ;CONVERT
024222  004737  032526          JSR     PC,$SUPRL       ;PRINT IT
024226  104414  057771          DISPLY  .PERIOD         ;TYPE '.'
024232  104414  057632          DISPLY  .MSGX10         ;TYPE ' X10+6'
024236  104414  001203          DISPLY  .$CRLF          ;CR-LF
024242  032777  100000  154704  BIT     @SW15,@SWR      ;SEE IF 'HALT ON ERROR' - SWITCH 15
024250  001401                  BEQ     1$              ;BR IF NOT
024252  000000                  HALT                    ;SWITCH 15 HALT
024254  000207          1$:     RTS     PC
                                ;
                                ;PRINT LINE 7A OF ERROR MESSAGE
                                ;'TOTALS; SEEKS= XXXXX  MIS POS ERRORS= XXX  SKI ERRORS= XXX'
                                ;
024256  104414  055535  LINE7A: DISPLY  .LIN7P           ;'TOTALS; SEEKS= '
024262  012746  000076          MOV     @$STOTL,-(SP)   ;TOTAL SEEKS
024266  060016                  ADD     R0,(SP)         ;DEVICE TABLE ADDRESS
024270  004737  037404          JSR     PC,$D82D        ;CONVERT THE SEEK COUNT
024274  004737  032526          JSR     PC,$SUPRL       ;PRINT IT
024300  104414  057771          DISPLY  .PERIOD         ;TYPE '.'
024304  104414  055512          DISPLY  .LIN7M          ;'  MIS POS ERRORS= '
024310  016046  000112          MOV     $MISPO(R0),-(SP)        ;TOTAL ERRORS
024314  004737  024420          JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
024320  104414  055555          DISPLY  .LIN7S          ;'  SKI ERRORS= '
```

```
                                        MOV     $SKI(R0), (SP)  ;CONVERT & PRINT IT
                                        JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
                                        DISPLY  ,$CRLF          ;CR LF
                                        BIT     #SW15,@SWR      ;SEE IF HALT ON ERROR - SWITCH 15 SET
                                        BEQ     1$              ;BR IF NOT
                                        HALT                    ;SWITCH 15 HALT
                                1$:     RTS     PC

                                ;PRINT LINE 8 OF THE ERROR MESSAGE
                                ;'DIFFERENT ERROR DURING RETRY'

                                LINE8:  DISPLY  ,LIN8M
                                        JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
                                        RTS     PC

                                ;OCTAL TYPEOUT ROUTINE
                                ;CALL:
                                ;       MOV     NUM,-(SP)       ;PUT THE NUMBER ON THE STACK
                                ;       JSR     PC,LINOCT
                                ;       RETURN

                                LINOCT: MOV     2(SP),-(SP)     ;PUT NUMBER IN PROPER LOCATION ON STACK
                                        JSR     PC,$SB2O        ;CONVERT THE NUMBER TO OCTAL
                                        MOV     (SP)+,1$        ;GET THE ADDRESS OF THE ASCII STRING
                                        ADD     #5.,1$          ;ADDRESS THE LAST 6 ASCII DIGITS
                                        DISPLY                  ;TYPE IT
                                1$:     .WORD   0               ;ADDRESS
                                        MOV     (SP)+,(SP)      ;CORRECT THE STACK
                                        RTS     PC              ;RETURN

                                ;ROUTINE TO CONVERT THE INPUT NUMBER TO DECIMAL AND TYPE IT
                                ;LEFT JUSTIFIED
                                ;CALL:
                                ;       MOV     NUM,-(SP)       ;PUT THE NUMBER ON THE STACK
                                ;       JSR     PC,LINDEC
                                ;       RETURN

                                LINDEC: MOV     2(SP),-(SP)     ;SET UP STACK FOR CONVERT
                                        JSR     PC,$SB2D        ;CONVERT IT TO DECIMAL
                                        JSR     PC,$SUPRL       ;TYPE IT (LEFT JUSTIFIED)
                                        DISPLY  ,PERIOD         ;TYPE '.'
                                        MOV     (SP)+,(SP)      ;RESTORE STACK POINTER
                                        RTS     PC

                                .SBTTL  GENERAL SUPPORT SUBROUTINES

                                ;DECREMENT THE SECTOR-TRACK ADDRESS
                                ;CALL:
                                ;       MOV     #DPB,R0         ;DPB ADDRESS
                                ;       JSR     PC,READDR
                                ;       RETURN
                                ;
                                ;ON RETURN THE STACK CONTAINS THE FOLLOWING:
                                ;       4(SP) = SECTOR ADDRESS
                                ;       2(SP) = TRACK ADDRESS
                                ;        (SP) = CYLINDER ADDRESS
                                ;
```

```
479 024444  162706  000006          READDR: SUB     #6,SP               ;DECREMENT THE STACK POINTER
480 024450  016616  000006                  MOV     6(SP),(SP)          ;MOVE THE RETURN ADDR DOWN THE STACK
481 024454  005066  000006                  CLR     6(SP)               ;CLEAR STACK FOR SECTOR
482 024460  005066  000004                  CLR     4(SP)               ;CLEAR STACK FOR TRACK
483 024464  116066  000174  000006          MOVB    @RPDA(R0),6(SP)     ;SECTOR ON STACK
484 024472  116066  000175  000004          MOVB    @RPDA+1(R0),4(SP)           ;TRACK ADDRESS
485 024500  016066  000222  000002          MOV     @RPDC(R0),2(SP)     ;CYLINDER ADDRESS
486 024506  005766  000006          1$:     TST     6(SP)               ;SECTOR 0 ?
487 024512  001403                  BEQ     2$                          ;BRANCH IF SO
488 024514  105366  000006                  DECB    6(SP)               ;DECREMENT ONE SECTOR
489 024520  000424                  BR      4$                          ;BRANCH TO EXIT
490 024522  005766  000004          2$:     TST     4(SP)               ;ALSO ON TRACK 0 ?
491 024526  001406                  BEQ     3$                          ;BRANCH IF SO
492 024530  116066  000152  000006          MOVB    SECLMT(R0),6(SP)            ;LAST SECTOR
493 024536  105366  000004                  DECB    4(SP)               ;DECREMENT ONE TRACK
494 024542  000413                  BR      4$                          ;EXIT
495 024544  005766  000002          3$:     TST     2(SP)               ;ALSO ON CYLINDER 0 ?
496 024550  001410                  BEQ     4$                          ;BRANCH IF SO
497 024552  116066  000152  000006          MOVB    SECLMT(R0),6(SP)            ;LAST SECTOR
498 024560  116066  000154  000004          MOVB    TRKLMT(R0),4(SP)            ;GET LAST TRACK
499 024566  005366  000002                  DEC     2(SP)               ;DECREMENT ONE CYLINDER COUNT
500 024572  000207                  4$:     RTS     PC                  ;RETURN
```

```
                                .SBTTL  KW11 CLOCK CHECK ROUTINE
                                ;
                                ;ROUTINE TO CHECK FOR KW11 L OR KW11 P CLOCKS
                                ;
024574  005037  001310          CKCLK:  CLR     CLKFLG          ;ASSUME "NO CLOCK" AVAILABLE
024600  013746  000004                  MOV     ERRVEC,-(SP)    ;;PUSH ERRVEC ON STACK
024604  012737  024656  000004          MOV     #CKCLK1,ERRVEC  ;SETUP ERROR TRAP VECTOR FOR P CLOCK CHECK
024612  005777  154460                  TST     @#LKCSR         ;CHECK FOR KW11-P
024616  012737  000001  001310          MOV     #1,CLKFLG       ;SET P-CLOCK FLAG
024624  013701  001302                  MOV     #LPVEC,R1       ;KW11-P VECTOR ADDRESS
024630  012721  026352                  MOV     #KWSVR,(R1)+    ;SETUP KW11-P VECTOR
024634  012711  000300                  MOV     #PR6,(R1)       ;SET INTERRUPT PRIORITY TO 6
024640  012777  174575  154432          MOV     #-1667.,@#LKCSB ;LOAD COUNTER BUFFER WITH 16.67
024646  012777  000131  154422          MOV     #131,@#LKCSR    ;SET KW11-P INTERRUPT, CNT UP, 10US, REPEAT MODE
024654  000442                  BR      CKCLK3
                                ;
024656  012716  024664          CKCLK1: MOV     #1$,(SP)        ;SETUP RETURN ADDRESS
024662  000002                  RTI
024664  012737  024730  000004  1$:     MOV     #CKCLK2,ERRVEC  ;SET UP ERROR TRAP VECTOR FOR L CLOCK CHECK
024672  005777  154406                  TST     @#LKS           ;CHECK FOR KW11-L
024676  012737  177777  001310          MOV     #-1,CLKFLG      ;SET L-CLOCK FLAG
024704  013701  001306                  MOV     #LPVEC,R1       ;KW11-L VECTOR ADDRESS
024710  012721  026352                  MOV     #KWSVR,(R1)+    ;SETUP KW11-L VECTOR
024714  012711  000300                  MOV     #PR6,(R1)       ;SET INTERRUPT PRIORITY TO 6
024720  012777  000100  154356          MOV     #100,@#LKS      ;SET KW11-L INTERRUPT
024726  000415                  BR      CKCLK3
                                ;
024730  012716  024736          CKCLK2: MOV     #1$,(SP)        ;SETUP RETURN ADDRESS
024734  000002                  RTI
024736  104401  057714          1$:     TYPE    ,NEDCLK         ;'P OR L CLOCK MUST BE ON SYSTEM'
024742  105737  001150                  TSTB    @#AUTOB         ;RUNNING IN AUTO MODE ?
024746  001402                  BEQ     2$              ;BR IF NOT
024750  000137  032024                  JMP     @#GET42         ;ABORT PROGRAM
024754  000000          2$:     HALT                    ;HALT
024756  000137  003522                  JMP     START           ;TRY AGAIN
                                ;
024762  012637  000004          CKCLK3: MOV     (SP)+,ERRVEC    ;RESTORE THE ERROR VECTOR
024766  000207                  RTS     PC
                                ;
                                ;THIS ROUTINE IS USED TO SHUT OFF INTERRUPT TO THE SYSTEM CLOCK
                                ;CALL
                                ;       JSR     PC,CLKOFF       ;CALL ROUTINE
                                ;
024770  005737  001310          CLKOFF: TST     CLKFLG          ;IS CLOCK AVAILABLE ?
024774  001410                  BEQ     2$              ;BR IF NO
024776  100404                  BMI     1$              ;BR IF L-CLOCK
025000  042777  000101  154270          BIC     #101,@#LKCSR    ;SHUT OFF KW11-P INTERRUPT
025006  000403                  BR      2$
025010  042777  000100  154266  1$:     BIC     #100,@#LKS      ;SHUT OFF KW11 L INTERRUPT
025016  000207          2$:     RTS     PC              ;EXIT
```

```
                                ;ROUTINE TO DISPLAY STATISTICS FOR ASSIGNED DRIVES
                                ;CALL:
                                ;       JSR     PC,STATPR
                                ;       RETURN

025020                          STATPR:
025020  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
025022  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
025024  005737  001542                  TST     ASMLST          ;ANY DRIVES ASSIGNED ?
025030  001431                          BEQ     5$              ;BR IF NOT
025032  005004                  2$:     CLR     R4              ;CLEAR THE DRIVE INDEX
025034  104401  001203                  TYPE    ,$CRLF          ;CR-LF
025040  006304                  3$:     ASL     R4              ;CHANGE TO INDEX WORDS
025042  016400  002056                  MOV     BLKADR(R4),R0   ;GET THE DRIVE'S BLOCK ADDRESS
025046  006204                          ASR     R4              ;RESTORE R4
025050  136437  040630  001542          BITB    ATABIT(R4),ASMLST       ;IS THIS DRIVE ASSIGNED ?
025056  001412                          BEQ     4$              ;BR IF NOT
025060  104401  001203                  TYPE    ,$CRLF          ;CR-LF
025064  004737  026226                  JSR     PC,$TIME        ;TYPE ELAPSED TIME
025070  104401  057401                  TYPE    ,DASH5          ;TYPE '-----'
025074  104401  057425                  TYPE    ,MSGSUM         ;TYPE 'SUMMARY, '
025100  004737  025150                  JSR     PC,TYPSUM       ;TYPE THE SUMMARY
025104  005204                  4$:     INC     R4              ;INCREMENT THE INDEX
025106  020427  000010                  CMP     R4,#8.          ;FINISHED ?
025112  001352                          BNE     3$              ;BR IF NO
025114                          5$:
025114  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
025116  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
025120  000207                          RTS     PC              ;RETURN

                                ;ROUTINE TO TYPE THE SUMMARY FOR ONLY ONE DRIVE
                                ;CALL:
                                ;       MOV     #DPB,R0         ;DPB ADDRESS
                                ;       JSR     PC,ONESUM
                                ;       RETURN

025122  010046                  ONESUM: MOV     R0,-(SP)        ;SAVE R0
025124  010446                          MOV     R4,-(SP)        ;SAVE R4
025126  005004                          CLR     R4              ;CLEAR R4 FOR DRIVE NUMBER
025130  111004                          MOVB    (R0),R4         ;DRIVE NUMBER
025132  004737  025150                  JSR     PC,TYPSUM       ;TYPE THE SUMMARY
025136  104401  057407                  TYPE    ,DASH13         ;TYPE '-------------'
025142  012604                          MOV     (SP)+,R4        ;RESTORE R4
025144  012600                          MOV     (SP)+,R0        ;RESTORE R0
025146  000207                          RTS     PC              ;RETURN

                                ;TYPE THE SUMMARY
                                ;CALL:
                                ;       MOV     #DRIVE,R4       ;DRIVE NUMBER
                                ;       MOV     #DPB,R0         ;DPB ADDRESS
                                ;       RETURN

025150  000240                  TYPSUM: NOP

                                ;TYPE REST OF SUMMARY LINE 1
025152  104401  056661                  TYPE    ,DRVMSG         ;TYPE 'DRIVE'
```

```
                        ;RH11 CLOCK CHECK ROUTINE

 69 025156  010046                  MOV     R0,-(SP)        ;;SAVE R0 FOR TYPEOUT
                                                             ;;TYPE DRIVE NUMBER
    025160  104403                  TYPOS                   ;;GO TYPE--OCTAL ASCII
    025162     002                  .BYTE   2               ;;TYPE 2 DIGIT(S)
    025163     000                  .BYTE   0               ;;SUPPRESS LEADING ZEROS
 70 025164  104401  056611          TYPE    .BLNKS1         ;TYPE 1 BLANK
 71 025170  104401  056655          TYPE    .DASH           ;TYPE '-'
 72 025174  104401  056611          TYPE    .BLNKS1         ;TYPE 1 BLANK
 73 025200  012737  057155  025226  MOV     #RP07,2$        ;ADDRESS OF RP07 MESSAGE
 74 025206  122764  000005  040344  CMPB    @S.DRVTYP(R0)   ;IS DEVICE AN RP07 ?
 75 025214  001403                  BEQ     1$              ;BR IF YES
 76 025216  012737  057162  025226  MOV     #RP07P,2$       ;ADDRESS OF RP07+ MESSAGE
 77 025224  104401                  1$:     TYPE            ;TYPE THE DRIVE TYPE MESSAGE
 78 025226  000000                  2$:     .WORD   0       ;MESSAGE ADDRESS HERE
 79 025230  104401  056655          TYPE    .COMMA          ;TYPE '-'
 80 025234  104401  056611          TYPE    .BLNKS1         ;TYPE 1 BLANK
 81 025240  004737  033044          JSR     PC,TYDRV        ;TYPE DRIVE SERIAL NUMBER
 82 025244  104401  025252          TYPE    .65$            ;;TYPE ASCIZ STRING
    025250  000404                  BR      64$             ;;GET OVER THE ASCIZ
                        ;;65$:  .ASCIZ  /, PASS /
    025262                      64$:
 83 025262  016046  000114          MOV     @PASSC(R0),-(SP)        ;PUT THE PASS COUNT ON THE STACK
 84 025266  004737  033412          JSR     PC,$SB2D        ;CONVERT IT
 85 025272  004537  032632          JSR     R5,REPLZ        ;TYPE IT
 86 025276  000003                  .WORD   3               ;TYPE 3 DIGITS
 87 025300  104401  057771          TYPE    .PERIOD         ;TYPE '.'
 88
 89                                 ;TYPE LINE 2 OF SUMMARY
 90                                 ;;;     TYPE    .$CRLF          ;CR-LF
 91
 92                                 ;TYPE LINE 3 OF SUMMARY
 93 025304  104401  025312          TYPE    .67$            ;;TYPE ASCIZ STRING
    025310  000407                  BR      66$             ;;GET OVER THE ASCIZ
                        ;;67$:  .ASCIZ  <CRLF>/MROS WRITN /
    025330                      66$:
 94 025330  104401  057612          TYPE    .MSPASS         ;TYPE '/ PASS '
 95 025334  010046                  MOV     R0,-(SP)        ;GET ADDRESS OF DPB
 96 025336  062716  000042          ADD     #MTPAS,(SP)     ;POINT TO LOW NUMBER OF MROS WRITTEN PER PASS
 97 025342  004737  037404          JSR     PC,$DB2D        ;CONVERT DECIMAL NUMBER
 98 025346  004737  032442          JSR     PC,SUPRS        ;SUPPRESS LEADING ZEROS AND TYPE
 99 025352  104401  057771          TYPE    .PERIOD         ;TYPE '.'
100 025356  104401  057622          TYPE    .MSTOTL         ;TYPE '  TOTAL '
101 025362  010046                  MOV     R0,-(SP)        ;GET ADDRESS OF DPB
102 025364  062716  000056          ADD     #MTOTL,(SP)     ;POINT TO LOW NUMBER OF MROS WRITTEN
103 025370  004737  037404          JSR     PC,$DB2D        ;CONVERT DECIMAL NUMBER
104 025374  004737  032442          JSR     PC,SUPRS        ;SUPPRESS LEADING ZEROS AND TYPE
105 025400  104401  057771          TYPE    .PERIOD         ;TYPE '.'
106 025404  104401  057632          TYPE    .MSGX10         ;TYPE ' X10^6'
107
108                                 ;TYPE LINE 4 OF SUMMARY
109 025410  104401  025416          TYPE    .69$            ;;TYPE ASCIZ STRING
    025414  000407                  BR      68$             ;;GET OVER THE ASCIZ
                        ;;69$:  .ASCIZ  <CRLF>/MROS READ  /
    025434                      68$:
110 025434  104401  057612          TYPE    .MSPASS         ;TYPE '/ PASS '
111 025440  010046                  MOV     R0,-(SP)        ;GET ADDRESS OF DPB
112 025442  062716  000036          ADD     #RDPAS,(SP)     ;POINT TO LOW NUMBER OF MROS READ PER PASS
```

```
                                        ;CONVERT DECIMAL NUMBER
                                        ;SUPPRESS LEADING ZEROS AND TYPE
                                        ;TYPE ' :'
                                        ;TYPE ' TOTAL '
                                        ;GET ADDRESS OF DMB
                                        ;POINT TO LOW NUMBER OF WDS READ
                                        ;CONVERT DECIMAL NUMBER
                                        ;SUPPRESS LEADING ZEROS AND TYPE
                                        ;TYPE ' :'
                                        ;TYPE ' #10#B'

                        ;TYPE LINE 5 OF SUMMARY
                                        ;;TYPE ASCIZ STRING
                                        ;;GET OVER THE ASCIZ
                        ;;710:  .ASCIZ  <CRLF>/SEEKS          /
                        ;60:
                                        ;TYPE ' / PASS '
                                        ;PUT @STOTL ON THE STACK
                                        ;POINT TO LOW NUMBER OF SEEK COUNT
                                        ;CONVERT DECIMAL NUMBER
                                        ;SUPPRESS LEADING ZEROS AND TYPE
                                        ;TYPE ' :'
                                        ;TYPE ' TOTAL '
                                        ;PUT @STOTL ON THE STACK
                                        ;POINT TO LOW NUMBER OF SEEK COUNT
                                        ;CONVERT DECIMAL NUMBER
                                        ;SUPPRESS LEADING ZEROS AND TYPE
                                        ;TYPE ' :'

                        ;TYPE LINES 6 AND 7 OF SUMMARY
                                        ;;TYPE ASCIZ STRING
                                        ;;GET OVER THE ASCIZ
                        ;;730:  .ASCIZ  <CRLF>/CUMULATIVE ERRORS:/
                        ;20:
                                        ;;TYPE ASCIZ STRING
                                        ;;GET OVER THE ASCIZ
                        ;;750:  .ASCIZ  <CRLF>/SOFT /
                        ;40:
                                        ;;SAVE @SOFT(R0) FOR TYPEOUT
                                        ;;GO TYPE -DECIMAL ASCII WITH SIGN
                                        ;TYPE ' :'
                                        ;;TYPE ASCIZ STRING
                                        ;;GET OVER THE ASCIZ
                        ;;770:  .ASCIZ  / HARD /
                        ;60:
                                        ;;SAVE @HARD(R0) FOR TYPEOUT
                                        ;;GO TYPE--DECIMAL ASCII WITH SIGN
                                        ;TYPE ' :'
                                        ;;TYPE ASCIZ STRING
                                        ;;GET OVER THE ASCIZ
                        ;;790:  .ASCIZ  / SKI /
                        ;80:
                                        ;;SAVE @SKI(R0) FOR TYPEOUT
                                        ;;GO TYPE--DECIMAL ASCII WITH SIGN
                                        ;TYPE ' :'
                                        ;;TYPE ASCIZ STRING
                                        ;;GET OVER THE ASCIZ
```

```
                                                ;010:   ASCII  / MISP /
                                                ;ASC:
                        MOV     ERRSPC(R0), -(SP)       ;SAVE ERRSPC(R0) FOR TYPEOUT
                        .TYOCT                  ;GO TYPE - DECIMAL ASCII WITH SIGN
                        .TYPE   .ERRTXT         ;TYPE
                        .TYPE   .010            ;TYPE ASCII STRING
                        BR      $20             ;SKIP OVER THE ASCII
                ;010:   ASCII  / OTHER /
                $20:
                        MOV     TOTAL(R0),-(SP)         ;CALCULATE NUMBER OF OTHER ERRORS
                        SUB     LOST(R0),(SP)   ;SUB-BACK LOST FROM TOTAL
                        SUB     OVRRUN(R0),(SP) ;SUB-BACK OVRRUN FROM TOTAL
                        SUB     DSR(R0),(SP)    ;SUB-BACK DSR FROM TOTAL
                        SUB     ERRSPC(R0),(SP) ;SUB-BACK ERRSPC FROM TOTAL
                        .TYOCT                  ;GO TYPE - DECIMAL ASCII WITH SIGN
                        .TYPE   .ERRTXT         ;TYPE
                        .TYPE   .CRLF           ;CR-LF
                        RTS     PC
```

```
15
16
                                ;ROUTINE TO INCREMENT XSEC
                                ;
                                ;NOTE:  XSEC WILL NOT BE INCREMENTED BEYOND 77777 (32767.)

 026162  005737  001336        INCSEC: TST     BADSEC          ;SEE IF BAD TRK/SEC INDICATOR SET
 026166  001006                  BNE     10              ;OR IF IT'S SET, DON'T INCREMENT COUNT
 026170  026027  000160  077777        CMP     XSEC(R0),#77777 ;IS XSEC ALREADY AT MAXIMUM ?
 026176  103002                  BHIS    10              ;OR IF IT IS
 026200  005260  000160          INC     XSEC(R0)                ;INCREMENT XSEC
 026204  000207                  10:     RTS     PC              ;RETURN
17
                                ;ROUTINE TO INCREMENT XSECB
                                ;
                                ;NOTE:  XSECB WILL NOT BE INCREMENTED BEYOND 77777 (32767.)

 026206  005737  001336        INCSECB: TST    BADSEC          ;SEE IF BAD TRK/SEC INDICATOR SET
 026212  001006                  BNE     10              ;OR IF IT'S SET, DON'T INCREMENT COUNT
 026214  026027  000166  077777        CMP     XSECB(R0),#77777 ;IS XSECB ALREADY AT MAXIMUM ?
 026222  103002                  BHIS    10              ;OR IF IT IS
 026224  005260  000166          INC     XSECB(R0)               ;INCREMENT XSECB
 026230  000207                  10:     RTS     PC              ;RETURN
18
                                ;ROUTINE TO INCREMENT XSECPO
                                ;
                                ;NOTE:  XSECPO WILL NOT BE INCREMENTED BEYOND 77777 (32767.)

 026232  005737  001336        INCSEC: TST     BADSEC          ;SEE IF BAD TRK/SEC INDICATOR SET
 026236  001006                  BNE     10              ;OR IF IT'S SET, DON'T INCREMENT COUNT
 026240  026027  000172  077777        CMP     XSECPO(R0),#77777       ;IS XSECPO ALREADY AT MAXIMUM ?
 026246  103002                  BHIS    10              ;OR IF IT IS
 026250  005260  000172          INC     XSECPO(R0)              ;INCREMENT XSECPO
 026254  000207                  10:     RTS     PC              ;RETURN
19
                                ;ROUTINE TO INCREMENT XSECB
                                ;
                                ;NOTE:  XSECB WILL NOT BE INCREMENTED BEYOND 77777 (32767.)

 026256  005737  001336        INCSEC: TST     BADSEC          ;SEE IF BAD TRK/SEC INDICATOR SET
 026262  001006                  BNE     10              ;OR IF IT'S SET, DON'T INCREMENT COUNT
 026264  026027  000152  077777        CMP     XSECB(R0),#77777        ;IS XSECB ALREADY AT MAXIMUM ?
 026272  103002                  BHIS    10              ;OR IF IT IS
 026274  005260  000152          INC     XSECB(R0)               ;INCREMENT XSECB
 026300  000207                  10:     RTS     PC              ;RETURN
20
                                ;ROUTINE TO INCREMENT XTOTAL
                                ;
                                ;NOTE:  XTOTAL WILL NOT BE INCREMENTED BEYOND 77777 (32767.)

 026302  005737  001336        INCTOT: TST     BADSEC          ;SEE IF BAD TRK/SEC INDICATOR SET
 026306  001006                  BNE     10              ;OR IF IT'S SET, DON'T INCREMENT COUNT
 026310  026027  000102  077777        CMP     XTOTAL(R0),#77777       ;IS XTOTAL ALREADY AT MAXIMUM ?
 026316  103002                  BHIS    10              ;OR IF IT IS
 026320  005260  000102          INC     XTOTAL(R0)              ;INCREMENT XTOTAL
 026324  000207                  10:     RTS     PC              ;RETURN
```

KW11 CLOCK CHECK ROUTINE

```
                                        ;ROUTINE TO TYPE THE ELAPSED CPU RUN TIME
                                        ;
000000  005737  001310        BTIME:  TST     CLKFLG          ;IS CLOCK AVAILABLE ?
000004  001446                BEQ     10$             ;BR IF NO
000006  012737  000002  026302        MOV     #2,21           ;# OF DIGITS TO TYPE
000014  013746  001300                MOV     @#0.(SP)        ;GET MINUTES ON THE STACK
000020  021627  000100                CMP     (SP),#100       ;>100 MINUTES OR MORE ?
000024  002407                        BLT     10$             ;BR IF NO
000026  012737  000003  026302        MOV     #3,21           ;TYPE 3 DIGITS
000034  021627  001750                CMP     (SP),#1000      ;>1000 MINUTES OR MORE ?
000040  002402                        BLT     10$             ;BR IF NO
000042  012737  000004  026302        MOV     #4,21           ;TYPE 4 DIGITS
000050  004737  015412        10$:    JSR     PC,$ITOA2       ;CONVERT TO DECIMAL
000054  004537  012626                JSR     R5,$TYPE2       ;TYPE IT
000060  000000                20$:    .WORD   0               ;NUMBER OF HOUR DIGITS TO TYPE
000062  104401  000156                TYPE    .COLON          ;
000066  013746  001342                MOV     @#MINUTE,-(SP)  ;PUT MINUTES ON THE STACK
000072  004737  015412                JSR     PC,$ITOA2       ;CONVERT TO DECIMAL
000076  004537  012626                JSR     R5,$TYPE2       ;TYPE IT
000102  000002                        .WORD   2               ;TYPE 2 DIGITS
000104  104401  000156                TYPE    .COLON          ;
000110  013746  001344                MOV     @#SECOND,-(SP)  ;PUT SECONDS ON THE STACK
000114  004737  015412                JSR     PC,$ITOA2       ;CONVERT TO DECIMAL
000120  004537  012626                JSR     R5,$TYPE2       ;TYPE IT
000124  000002                        .WORD   2               ;TYPE 2 DIGITS
000126  000207                30$:    RTS     PC
                                        ;
                                        ;KW11 CLOCK INTERRUPT SERVICE ROUTINE
                                        ;
000130  005337  001346        KWSVR:  DEC     @#ONESEC        ;INCREMENT THE 1/60 SECOND COUNTER
000134  001037                        BNE     10$             ;BR IF A SECOND NOT COUNTED
000136  013737  001312  001346        MOV     @#HERTZ,@#ONESEC ;RESTORE THE VALUE
000144  005237  001344                INC     @#SECOND        ;COUNT THE SECOND
000150  023727  001344  000074        CMP     @#SECOND,#60.   ;AT MAXIMUM ?
000156  103426                        BLO     10$             ;BR IF NOT
000160  005037  001344                CLR     @#SECOND        ;CLEAR THE SECOND'S COUNTER
000164  005237  001472                INC     @#INTRVL+2      ;COUNT SUMMARY INTERVAL COUNTER
000170  005237  001464                INC     @#CMPTIM+2      ;COUNT COMPARE TIME INTERVAL COUNTER
000174  005237  001342                INC     @#MINUTE        ;COUNT THE MINUTE
000200  023727  001342  000074        CMP     @#MINUTE,#60.   ;AT MAXIMUM ?
000206  103412                        BLO     10$             ;BR IF NOT
000210  005037  001342                CLR     @#MINUTE        ;CLEAR THE MINUTE'S COUNTER
000214  005237  001340                INC     @#HOUR          ;COUNT THE HOURS
000220  023727  001340  023417        CMP     @#HOUR,#9999.   ;AT MAXIMUM
000226  101402                        BLOS    10$             ;BR IF NOT
000230  005037  001340                CLR     @#HOUR          ;CLEAR THE HOURS
000234  012746  000024        10$:    MOV     #20.,-(SP)      ;20MS ON THE STACK @ 50HZ
000240  023727  001312  000062        CMP     @#HERTZ,#50.    ;CPU RUNNING @ 50HZ ?
000246  001402                        BEQ     20$             ;BR IF YES
000250  012716  000020                MOV     #16.,(SP)       ;16MS ON THE STACK @ 60HZ
000254  004737  044402        20$:    JSR     PC,@#RPTMR      ;DRIVER TIMER ROUTINE
000260  005737  001470                TST     @#INTRVL        ;DISPLAY THE PERFORMANCE SUMMARY ?
000264  001411                        BEQ     30$             ;BR IF NOT
000266  023737  001472  001470        CMP     @#INTRVL+2,@#INTRVL ;DISPLAY INTERVAL FINISHED ?
000274  002405                        BLT     30$             ;BR IF NO
000276  012737  177777  001314        MOV     #-1,@#STATIN    ;SET PERFORMANCE SUMMARY DISPLAY FLAG
```

```
CZRJOBO RP07 PERF EXER MACRO V04 00 1 DEC 83 10:52:28 PAGE 41 1                                SEQ 0142
KW11 CLOCK CHECK ROUTINE

     58 026526  005037  001472                CLR     INTRVL+2        ;CLEAR THE PERFORMANCE INTERVAL COUNTER
     59 026532  000002             38:        RTI
```

```
                                        ;COMMAND DECODE ROUTINE
                                        ;CALL:
                                        ;       MOV     #1,CFLAG        ;'CFLAG' IS NORMALLY SET BY THE TTY SERVICE
                                        ;                               ;ROUTINE IN INTERRUPT MODE
                                        ;       JSR     PC,KSR
                                        ;       RETURN1                 ;SYSTEM BUSY RETURN
                                        ;       RETURN2                 ;RETURN AFTER KEYBOARD SERVICED

026534  005737  040626          KSR:  TST     DTUW            ;ANY DATA TRANSFERS UNDER WAY ?
026540  100375                  KSR1: BPL     KSR             ;BR IF YES
026542  104412                        SAVREG                  ;SAVE THE REGISTERS
026544  012737  000200  177776        MOV     #PR4,PS         ;SET PRIORITY TO 4
026552  013704  040640          1$:   MOV     RPADR,R4        ;GET RP/RH BASE ADDRESS
026556  012764  000040  000010        MOV     #CLR,RPCS2(R4)  ;CLEAR MASSBUS CONTROLLER
026564  005037  001334                CLR     CFLAG           ;CLEAR THE 'CONTROL C  FLAG
026570  104401  001203                TYPE    ,$CRLF          ;CR-LF
026574  004737  026226                JSR     PC,$TIME        ;TYPE ELAPSED TIME
026600  104401  056653                TYPE    ,COMMA          ;TYPE ','
026604  104401  056611                TYPE    ,BLNKS1         ;TYPE 1 BLANK
026610  104401  034736                TYPE    ,$MSWR+2        ;TYPE 'SWR = '
026614  017746  152334                MOV     @SWR,-(SP)      ;;SAVE @SWR FOR TYPEOUT
                                                              ;;CONTENTS OF SWITCH REGISTER
026620  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
026622  004737  033510                JSR     PC,$TKINT       ;INITIALIZE TTY KEYBOARD
026626  004737  024770                JSR     PC,CLKOFF       ;SHUT OFF CLOCK INTERRUPT WHILE WAITING
026632  104401  060061                TYPE    ,ENTCOM         ;'ENTER COMMAND'

026636  104411                        RDLIN                   ;READ THE KEYBOARD
026640  012605                        MOV     (SP)+,R5        ;GET ADDRESS OF INPUT STRING
026642  005737  001334                TST     CFLAG           ;WAS (^C) TYPED?
026646  001405                        BEQ     2$              ;BR IF NO
026650  005737  001542                TST     ASNLST          ;ANY DRIVES ASSIGNED ?
026654  001141                        BNE     13$             ;BR IF YES
026656  000137  003522                JMP     START           ;JUMP TO START

026662  004737  024574          2$:   JSR     PC,CKCLK        ;START SYSTEM CLOCK
026666  005205                        INC     R5              ;POINT TO SECOND CHARACTER
026670  122715  000124                CMPB    #'T,(R5)        ;EQ TO A 'T' ?
026674  001465                        BEQ     9$              ;YES
026676  122715  000101                CMPB    #'A,(R5)        ;EQ TO AN 'A
026702  001410                        BEQ     3$              ;BR IF IT IS
026704  121527  000067                CMPB    (R5),#'7        ;DRIVE NUMBER GREATER THAN AN ASCII 7 ?
026710  101117                        BHI     12$             ;BR IF IT IS
026712  121527  000060                CMPB    (R5),#'0        ;DRIVE NUMBER LESS THAN AN ASCII 0 ?
026716  103514                        BLO     12$             ;BR IF IT IS
026720  142715  177770                BICB    #^C7,(R5)       ;LEAVE ONLY LOWER 3 BITS IF CHAR NOT 'A'
026724  122765  000124  177777  3$:   CMPB    #'T,-1(R5)      ;EQ TO 'T'
026732  001003                        BNE     4$              ;BR IF NOT EQ
026734  004737  030110                JSR     PC,NEWASN       ;ASSIGN DRIVE FOR TEST
026740  000507                        BR      13$             ;EXIT
026742  122765  000104  177777  4$:   CMPB    #'D,-1(R5)      ;EQ TO 'D' ?
026750  001003                        BNE     5$              ;BR IF NOT EQ
026752  004737  027700                JSR     PC,DROPD        ;DROP DRIVE
026756  000500                        BR      13$             ;EXIT
026760  122765  000123  177777  5$:   CMPB    #'S,-1(R5)      ;EQ TO 'S'
026766  001003                        BNE     6$              ;BR IF NOT EQ
```

```
 59 026770  004737  030006                    JSR     PC,$CMND        ;TYPE STATISTICS
 60 026774  000471                            BR      13$             ;EXIT
 61 026776  122765  000127  177777  6$:       CMPB    #'W, 1(R5)      ;EQ TO 'W'
 62 027004  001012                            BNE     8$              ;BR IF NOT EQ
 64 027006  005737  001424                    TST     RDONLY          ;LOCKED IN 'READ ONLY' MODE ?
 65 027012  001053                            BNE     11$             ;BR IF YES
 67 027014  032777  000001  152132            BIT     #SW0,@SWR       ;IS SWITCH 0 SET ?
 68 027022  001047                            BNE     11$             ;BR IF SET, CAN'T DO 'W' COMMAND
 69 027024  004737  030132          7$:       JSR     PC,DATAPK       ;WRITE DATA
 70 027030  000453                            BR      13$             ;EXIT
 71 027032  122765  000122  177777  8$:       CMPB    #'R, 1(R5)      ;EQ TO 'R' ?
 72 027040  001043                            BNE     12$             ;BR IF NOT EQ
 73 027042  004737  030120                    JSR     PC,REDAPK       ;READ DATA
 74 027046  000444                            BR      13$             ;EXIT
 75 027050  122765  000127  177777  9$:       CMPB    #'W, 1(R5)      ;WT COMMAND ?
 76 027056  001034                            BNE     12$             ;NO
 78 027060  005737  001424                    TST     RDONLY          ;LOCKED IN 'READ ONLY' MODE ?
 79 027064  001026                            BNE     11$             ;BR IF YES
 81 027066  032777  000001  152060            BIT     #SW0,@SWR       ;IS SWITCH 0 SET ?
 82 027074  001022                            BNE     11$             ;BR IF SET, CAN'T DO 'W' COMMAND
 83 027076  122765  000101  000001            CMPB    #'A,1(R5)       ;ALL DRIVES ?
 84 027104  001413                            BEQ     10$             ;YES
 85 027106  126527  000001  000067            CMPB    1(R5),#'7       ;GREAT THAN 7
 86 027114  101015                            BHI     12$             ;YES
 87 027116  126527  000001  000060            CMPB    1(R5),#'0       ;LESS THAN 0
 88 027124  103411                            BLO     12$             ;YES
 89 027126  142765  177770  000001            BICB    #^C7,1(R5)      ;CHOP OFF THE HIGHER BITS
 90 027134  004737  030144          10$:      JSR     PC,WATPAK       ;ASSIGN DRIVES WITH WT COMMAND
 91 027140  000407                            BR      13$
 92 027142  104401  057773          11$:      TYPE    ,MSWRO          ;TYPE 'CAN'T WRITE IN READ ONLY MODE'
 93 027146  000601                            BR      1$              ;TRY AGAIN
 94 027150  104401  060036          12$:      TYPE    ,INVLD          ;TYPE 'INVALID COMMAND' MESSAGE
 95 027154  000137  026552                    JMP     1$              ;TRY AGAIN
 96 027160  104413                  13$:      RESREG                  ;RESTORE R0 - R5
 97 027162  005777  151774                    TST     @$TKB           ;CLEAR THE TTY BUFFER
 98 027166  052777  000100  151764            BIS     #BIT06,@$TKS    ;SET TTY INTERRUPT ENABLE
 99 027174  005037  177776                    CLR     PS              ;SET PRIORITY BACK TO ZERO
100 027200  000207                            RTS     PC              ;RETURN
```

```
                                ;ROUTINE TO PROCESS THE ASSIGN REQUEST ( 'I', 'R , OR  W  COMMANDS)
                                ;
027202  111504          ASSIGN: MOVB    (R5),R4         ;PUT DRIVE # IN R4
027204  005037  001334    1$:   CLR     CFLAG           ;CLEAR CONTROL C FLAG
027210  005037  001426          CLR     DRVPAR          ;ASSUME CHANGING DRIVE PARAMETERS
027214  104401  060244          TYPE    ,MSPRM          ;TYPE 'CHANGE DRIVE PARAMETERS ?'
027220  104411                  RDLIN                   ;READ THE ENTRY
027222  012600                  MOV     (SP)+,R0        ;SAVE ADDRESS OF RESPONSE
027224  005737  001334          TST     CFLAG           ;WAS (^C) TYPED?
027230  001365                  BNE     1$              ;BR IF YES
027232  105710                  TSTB    (R0)            ;WAS RESPONSE A CARRIAGE RETURN (DEFAULT 'N )?
027234  001414                  BEQ     3$              ;BR IF YES
027236  105760  000001          TSTB    1(R0)           ;WAS IT TERMINATED WITH CARRIAGE RETURN ?
027242  001006                  BNE     2$              ;BR IF NO
027244  122710  000131          CMPB    #'Y,(R0)        ;WAS IT A 'Y' RESPONSE ?
027250  001410                  BEQ     4$              ;BR IF YES
027252  122710  000116          CMPB    #'N,(R0)        ;WAS IT A 'N' RESPONSE ?
027256  001403                  BEQ     3$              ;BR IF YES
027260  104401  060163    2$:   TYPE    ,BADENT         ;TYPE BAD ENTRY MESSAGE
027264  000747                  BR      1$              ;TRY AGAIN
027266  005237  001426    3$:   INC     DRVPAR          ;DO NOT CHANGE DRIVE PARAMETERS
027272  122704  000101    4$:   CMPB    #'A,R4          ;ASSIGN ALL DRIVES ?
027276  001426                  BEQ     ASGN2           ;BR IF YES
                                ;
027300  012737  056732  031356 ASGN1:  MOV     #UNTASN,ASNMSG  ;ERROR MESSAGE
027306  005737  001430          TST     XXDP            ;LOADED FROM THIS DEVICE ?
027312  001407                  BEQ     1$              ;BR IF NO
027314  123704  001430          CMPB    XXDP,R4         ;LOADED FROM THIS DRIVE ?
027320  001004                  BNE     1$              ;BR IF NO
027322  012737  057041  031356        MOV     #LODEV,ASNMSG   ;'LOAD DEVICE' MESSAGE ADDRESS
027330  000407                  BR      2$
027332  136437  040630  001542  1$:   BITB    ATABIT(R4),ASNLST       ;DRIVE ALREADY ASSIGNED ?
027340  001003                  BNE     2$              ;BR IF IT IS
027342  004737  027444          JSR     PC,ASGN3        ;SEE IF DRIVE ON THE SYSTEM
027346  000207                  RTS     PC              ;RETURN
027350  000137  031332      2$: JMP     ASNERR          ;EXIT ERROR
                                ;
027354  005004          ASGN2:  CLR     R4              ;START WITH DRIVE 0
027356  012737  056732  031356 1$:    MOV     #UNTASN,ASNMSG  ;ERROR MESSAGE
027364  005737  001430          TST     XXDP            ;LOADED FROM THIS DEVICE ?
027370  001407                  BEQ     2$              ;BR IF NO
027372  123704  001430          CMPB    XXDP,R4         ;LOADED FROM THIS DRIVE ?
027376  001004                  BNE     2$              ;BR IF NO
027400  012737  057041  031356        MOV     #LODEV,ASNMSG   ;'LOAD DEVICE' MESSAGE ADDRESS
027406  000413                  BR      4$
027410  136437  040630  001542  2$:   BITB    ATABIT(R4),ASNLST       ;ALREADY ASSIGNED ?
027416  001007                  BNE     4$              ;YES
027420  004737  027444          JSR     PC,ASGN3        ;ASSIGN THE DRIVE
027424  005204              3$: INC     R4              ;INCREMENT DRIVE #
027426  020427  000007          CMP     R4,#7           ;ALL DRIVE CHECKED ?
027432  003751                  BLE     1$              ;NO
027434  000207                  RTS     PC              ;YES
027436  004737  031332      4$: JSR     PC,ASNERR       ;ERROR MESSAGE
027442  000770                  BR      3$              ;TO LOOP
                                ;
027444  136437  040630  001542 ASGN3:  BITB    ATABIT(R4),ASNLST       ;DRIVE ALREADY ASSIGNED ?
027452  001056                  BNE     ASGN4           ;BR IF IT IS
```

```
                                                ;GET DRIVE NUMBER
                                                ;MAKE R4 WORD INDEX
                                                ;PUT BLOCK'S ADDR INTO R0
                                                ;RECALIBRATE DRIVE
                                                ;MAKE R4 BYTE INDEX
                                                ;DRIVE AVAILABLE?
                                                ;BR IF DRIVE OFFLINE OR NONEXISTENT
                                                ;BR IF DRIVE UNSAFE
                                                ;CLEAR BLOCK FOR DRIVE JUST ASSIGNED
                                                ;GET DRIVE SERIAL NUMBER
                                                ;INTERLEAVE SECTOR SET ?
                                                ;BR IF NO
                                                ;CHANGE DRIVE PARAMETERS ?
                                                ;BR IF NO
                                                ;CR-LF
                                                ;TYPE 'DRIVE'
                                                ;;SAVE R4 FOR TYPEOUT
                                                ;;TYPE DRIVE NUMBER
                                                ;;GO TYPE--OCTAL ASCII
                                                ;;TYPE 2 DIGIT(S)
                                                ;;SUPPRESS LEADING ZEROS
                                                ;TYPE ','
                                                ;TYPE 1 BLANK
                                                ;TYPE DRIVE SERIAL NUMBER
                                                ;MAKE R4 WORD INDEX
                                                ;GET THE DRIVE'S ADDRESS LIMITS
                                                ;DPB ADDRESS
                                                ;SET COMMAND INDICATOR
                                                ;MAKE R4 BYTE INDEX
                                                ;RETURN

                                                ;'UNSAFE' MESSAGE ADDRESS
                                                ;TO ERROR ROUTINE

                                                ;DRIVE PRESENT?
                                                ;BR IF NOT
                                                ;BR IF DRIVE OFFLINE
                                                ;ADDRESS OF  NOT RP07' MSG
                                                ;EXIT
                                                ;ADDRESS OF 'NOT PRESENT' MSG
                                                ;EXIT
                                                ;ADDRESS OF 'DRIVE OFFLINE' MESSAGE
                                                ;TO ERROR ROUTINE

                                                ;ADDRESS OF 'NON-INTERLEAVED' MESSAGE
                                                ;TO ERROR ROUTINE
```

```
                                        ;'D  COMMAND (ROUTINE TO DROP A DRIVE)
                                        ;
027700  005004                  DROPD:  CLR     R4              ;START WITH DRIVE 0
027702  012703  000010                  MOV     #8.,R3          ;COUNTER
027706  122715  000101                  CMPB    #'A,(R5)        ;DROP ALL DRIVES ?
027712  001403                          BEQ     1$              ;BR IF YES
027714  111504                          MOVB    (R5),R4         ;GET DRIVE NUMBER
027716  012703  000001                  MOV     #1,R3           ;SET R3 FOR ONE DRIVE
027722  136437  040630  001542  1$:     BITB    ATABIT(R4),ASNLST       ;DRIVE ASSIGNED ?
027730  001417                          BEQ     3$              ;BR IF NOT
027732  146437  040630  001542          BICB    ATABIT(R4),ASNLST       ;DELETE THE DRIVE FROM THE ASSIGNED LIST
027740  146437  040630  032100          BICB    ATABIT(R4),AUTLST       ;DELETE DRIVE FROM AUTO ASSIGN LIST
027746  006304                          ASL     R4              ;MAKE ADDR INTO A WORD INDEX
027750  016464  002056  001544          MOV     BLKADR(R4),DDRVS(R4),   ;PUT ADDRESS IN DROP LIST
027756  006204                          ASR     R4
027760  005303                  2$:     DEC     R3              ;ANY MORE DRIVES ?
027762  001410                          BEQ     4$              ;BR IF NOT
027764  005204                          INC     R4
027766  000755                          BR      1$
027770  012737  056710  031356  3$:     MOV     #UNTNOT,ASNMSG  ;ADDR OF 'NOT ASSIGNED' MESSAGE
027776  004737  031332                  JSR     PC,ASNERR       ;REPORT IT
030002  000766                          BR      2$
030004  000207                  4$:     RTS     PC

                                        ;'S' COMMAND (ROUTINE TO TYPE DRIVE PERFORMANCE SUMMARY)

                                        SCMND:
030006  013746  001542                  MOV     ASNLST,-(SP)    ;;PUSH ASNLST ON STACK
030012  122715  000101                  CMPB    #'A,(R5)        ;ALL STATISTICS ?
030016  001416                          BEQ     2$              ;BR IF YES
030020  111504                          MOVB    (R5),R4         ;GET DRIVE NUMBER
030022  136416  040630                  BITB    ATABIT(R4),(SP) ;IS THIS DRIVE ASSIGNED ?
030026  001404                          BEQ     1$              ;BR IF NO
030030  116437  040630  001542          MOVB    ATABIT(R4),ASNLST       ;GET DRIVE ASSIGN BIT
030036  000411                          BR      3$

030040  012737  056710  031356  1$:     MOV     #UNTNOT,ASNMSG  ;ADDR OF 'NOT ASSIGNED' MSG
030046  004737  031332                  JSR     PC,ASNERR       ;TYPE ERROR MESSAGE
030052  000413                          BR      4$              ;EXIT
030054  105737  001542          2$:     TSTB    ASNLST          ;ANY DRIVE ASSIGNED ?
030060  001410                          BEQ     4$              ;BR IF NO
030062  004737  025020          3$:     JSR     PC,STATPR       ;TYPE ALL STATISTICS
030066  104401  057407                  TYPE    ,DASH13         ;TYPE '-------------'
030072  104401  001203                  TYPE    ,$CRLF          ;CR-LF
030076  104401  057573                  TYPE    ,MSGCON         ;TYPE 'CONTINUING...'

030102                          4$:
030102  012637  001542                  MOV     (SP)+,ASNLST    ;;POP STACK INTO ASNLST
030106  000207                          RTS     PC
```

```
                                        ;'T' COMMAND (ROUTINE TO TEST A DRIVE)
                                        ;
030110  005037  030162          NEWASN: CLR     PACK            ;SET 'T' COMMAND INDICATOR
030114  000137  027202                  JMP     ASSIGN          ;GO TO THE ASSIGN ROUTINE
                                        ;
                                        ;'R' COMMAND (ROUTINE TO DO SEQUENTIAL READ DATA)
                                        ;
030120  012737  000001  030162  REDAPK: MOV     #1,PACK         ;SET 'R' COMMAND INDICATOR
030126  000137  027202                  JMP     ASSIGN          ;ASSIGN THE REQUESTED DRIVE
                                        ;
                                        ;'W' COMMAND (ROUTINE TO DO SEQUENTIAL WRITE DATA)
                                        ;
030132  012737  177777  030162  DATAPK: MOV     #-1,PACK        ;SET 'W' COMMAND INDICATOR
030140  000137  027202                  JMP     ASSIGN          ;ASSIGN REQUESTED DRIVE
                                        ;
                                        ;'WT' COMMAND (ROUTINE TO DO WRITE DATA AND TEST A DRIVE)
                                        ;
030144  116515  000001          WATPAK: MOVB    1(R5),(R5)      ;ADJUST DRIVE NUMBER ADDRESS
030150  012737  177776  030162          MOV     #-2,PACK        ;SET 'WT' COMMAND INDICATOR
030156  000137  027202                  JMP     ASSIGN          ;JUMP TO ASSIGN ROUTINE
                                        ;
030162  000000                  PACK:   .WORD   0               ;TEMPORARY STORAGE FOR COMMAND INDICATOR
```

```
                                ;ROUTINE TO CLEAR THE DPB FOR THE ASSIGNED DRIVE
                                ;CALL:
                                ;       MOV     #DPB,R0         ;DPB ADDRESS
                                ;       JSR     PC,CLRDPB
                                ;       RETURN
                                ;
                                ;R0 = DPB ADDRESS BEFORE CALLING THE ROUTINE

 030164                         CLRDPB:
 030164  010146                         MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 030166  010346                         MOV     R3,-(SP)        ;;PUSH R3 ON STACK
 030170  010446                         MOV     R4,-(SP)        ;;PUSH R4 ON STACK
 030172  010546                         MOV     R5,-(SP)        ;;PUSH R5 ON STACK
 030174  005737  040100                 TST     PWRFLG          ;RETURNING FROM POWER FAIL ?
 030200  001073                         BNE     4$              ;BRANCH IF YES
 030202  010004                         MOV     R0,R4           ;GET THE DPB ADDRESS
 030204  062704  000002                 ADD     #2,R4           ;ADDRESS OF FIRST LOCN TO BE CLEARED
 030210  012703  000012                 MOV     #<$CYL-$COMND>/2,R3      ;NUMBER OF LOCNS TO BE CLEARED
 030214  005024                 1$:     CLR     (R4)+           ;CLEAR LOCATIONS '$COMND' - '$CYL' IN DPB
 030216  162703  000002                 SUB     #2,R3           ;DONE CLEARING YET ?
 030222  001374                         BNE     1$              ;BR IF NO

 030224  062704  000002                 ADD     #2,R4           ;SKIP OVER THE '$REG' LOCATION
 030230  012703  000114                 MOV     #<$NEXT-$TATUS>/2,R3     ;NUMBER OF LOCNS TO BE CLEARED
 030234  005024                 2$:     CLR     (R4)+           ;CLEAR LOCATIONS '$TATUS' - '$NEXT' IN DPB
 030236  162703  000002                 SUB     #2,R3           ;DONE CLEARING YET ?
 030242  001374                         BNE     2$              ;BR IF NO

 030244  062704  000026                 ADD     #<$DRVSN-$FIRST>,R4      ;SKIP OVER '$FIRST', MIN/MAX ADRS
                                                                 ;LIMITS AND 'CYL, TRK, SEC, FE1' LIMITS
 030250  012703  000062                 MOV     #<$RPCS3-$DRVSN>/2,R3    ;NUMBER OF LOCNS TO BE CLEARED
 030254  005024                 3$:     CLR     (R4)+           ;CLEAR LOCATIONS '$DRVSN' - '$RPCS3' IN DPB
 030256  162703  000002                 SUB     #2,R3           ;DONE CLEARING YET ?
 030262  001374                         BNE     3$              ;BR IF NO

                                                                 ;INITIALIZE SOME OTHER LOCATIONS
 030264  113760  001514  000024         MOVB    BEGCOD,$CODE(R0)        ;INITIAL COMMAND CODE
 030272  013701  001514                 MOV     BEGCOD,R1       ;GET THE ACTUAL OP CODE
 030276  116160  002076  000002         MOVB    COMTBL(R1),$COMND(R0)   ;OPERATION CODE
 030304  113760  001512  000030         MOVB    BEGPAT,$PATTC(R0)       ;PATTERN CODE
 030312  106360  000030                 ASLB    $PATTC(R0)      ;CONVERT CODE TO A TABLE INDEX
 030316  013760  001516  000020         MOV     BEGWC,$WRDL(R0) ;BEGINNING WORD COUNT
 030324  013760  001516  000004         MOV     BEGWC,$WCNT(R0) ;VALUE FOR DATA TRANSFER
 030332  005460  000004                 NEG     $WCNT(R0)       ;MAKE IT INTO 2'S COMPLEMENT
 030336  012760  000400  000022         MOV     #256.,$SSEC(R0) ;INITIAL VALUE OF SECTOR SIZE
 030344  012760  000001  000114         MOV     #1,$PASSC(R0)   ;PRESET PASS COUNT TO 1
 030352  132760  000001  000024         BITB    #1,$CODE(R0)    ;HEADER COMMAND ?
 030360  001403                         BEQ     4$              ;BR IF NOT
 030362  062760  000002  000022         ADD     #2,$SSEC(R0)    ;ADD HEADER SIZE TO SECTOR SIZE
 030370                         4$:
 030370  012605                         MOV     (SP)+,R5        ;;POP STACK INTO R5
 030372  012604                         MOV     (SP)+,R4        ;;POP STACK INTO R4
 030374  012603                         MOV     (SP)+,R3        ;;POP STACK INTO R3
 030376  012601                         MOV     (SP)+,R1        ;;POP STACK INTO R1
 030400  000207                         RTS     PC              ;RETURN
```

```
                                        ;ROUTINE TO GET ADDRESS LIMITS FROM THE OPERATOR
                                        ;CALL:
                                        ;       MOV     #DPB,R0         ;DPB ADDRESS
                                        ;       JSR     PC,DRVPRM       ;CALL ROUTINE
                                        ;
                                        ;R0 = DPB ADDRESS BEFORE CALLING THE ROUTINE

030402  010346                 DRVPRM: MOV     R3,-(SP)        ;SAVE R3
030404  010446                         MOV     R4,-(SP)        ;SAVE R4
030406  004737  030766          10:     JSR     PC,GETLMT       ;GET ADDRESS LIMITS
030412  062760  177777  000132          ADD     #-1,FIRST(R0)   ;SEE IF FIRST TIME STARTED
030420  103426                         BCS     40              ;BR IF NOT
030422  016060  000150  000134          MOV     CYLLMT(R0),MAXCYL(R0)   ;LOAD MAXIMUM CYLINDER
030430  016060  000154  000140          MOV     TRKLMT(R0),MAXTRK(R0)   ;LOAD MAXIMUM TRACK
030436  016060  000152  000144          MOV     SECLMT(R0),MAXSEC(R0)   ;LOAD MAXIMUM SECTOR
030444  005737  001422                 TST     TSTANY          ;ARE YOU TESTING ANYWHERE ON MEDIA ?
030450  001004                         BNE     20              ;BR IF YES
030452  016060  000156  000136          MOV     FE1(R0),MINCYL(R0)      ;RESET MINIMUM CYLINDER ADDRESS
030460  000402                         BR      30
030462  005060  000136          20:     CLR     MINCYL(R0)      ;CLEAR MINIMUM CYLINDER
030466  005060  000142          30:     CLR     MINTRK(R0)      ;CLEAR MINIMUM TRACK
030472  005060  000146                  CLR     MINSEC(R0)      ;CLEAR MINIMUM SECTOR

030476  105737  001150          40:     TSTB    @AUTOB          ;RUNNING IN AUTO MODE ?
030502  001074                         BNE     70              ;BR IF YES
030504  005737  001332                 TST     CHGADR          ;PROGRAM STARTED AT 200 ?
030510  003071                         BGT     70              ;BR IF YES
030512  005737  001426                 TST     DRVPAR          ;CHANGE DRIVE PARAMETERS ?
030516  001066                         BNE     70              ;BR IF NO
030520  016403  061644                 MOV     TABLE(R4),R3    ;PARAMETER TABLE ADDRESS
030524  016063  000150  000002          MOV     CYLLMT(R0),2(R3)        ;LOAD CYLINDER LIMIT FOR MINCYL
030532  016063  000150  000010          MOV     CYLLMT(R0),10(R3)       ;LOAD CYLINDER LIMIT FOR MAXCYL
030540  016063  000154  000016          MOV     TRKLMT(R0),16(R3)       ;LOAD TRACK LIMIT FOR MINTRK
030546  016063  000154  000024          MOV     TRKLMT(R0),24(R3)       ;LOAD TRACK LIMIT FOR MAXTRK
030554  016063  000152  000032          MOV     SECLMT(R0),32(R3)       ;LOAD SECTOR LIMIT FOR MINSEC
030562  016063  000152  000040          MOV     SECLMT(R0),40(R3)       ;LOAD SECTOR LIMIT FOR MAXSEC
030570  104401  060102                 TYPE    ,ENTLMT         ;' ADDRESS LIMITS;'
030574  004737  031170                 JSR     PC,PARENT       ;GET THE DRIVE'S PARAMETERS

030600  016003  000136                  MOV     MINCYL(R0),R3   ;STORE MINCYL VALUE
030604  016004  000134                  MOV     MAXCYL(R0),R4   ;STORE MAXCYL VALUE
030610  020304                         CMP     R3,R4           ;IS MIN. LESS THAN OR EQUAL TO MAX. ?
030612  003404                         BLE     50              ;BR IF YES
030614  010360  000134                  MOV     R3,MAXCYL(R0)   ;SWAP MIN. TO MAX.
030620  010460  000136                  MOV     R4,MINCYL(R0)   ;SWAP MAX. TO MIN.
030624  016003  000142          50:     MOV     MINTRK(R0),R3   ;STORE MINTRK VALUE
030630  016004  000140                  MOV     MAXTRK(R0),R4   ;STORE MAXTRK VALUE
030634  020304                         CMP     R3,R4           ;IS MIN. LESS THAN OR EQUAL TO MAX. ?
030636  003404                         BLE     60              ;BR IF YES
030640  010360  000140                  MOV     R3,MAXTRK(R0)   ;SWAP MIN. TO MAX.
030644  010460  000142                  MOV     R4,MINTRK(R0)   ;SWAP MAX. TO MIN.
030650  016003  000146          60:     MOV     MINSEC(R0),R3   ;STORE MINSEC VALUE
030654  016004  000144                  MOV     MAXSEC(R0),R4   ;STORE MAXSEC VALUE
030660  020304                         CMP     R3,R4           ;IS MIN. LESS THAN OR EQUAL TO MAX. ?
030662  003404                         BLE     70              ;BR IF YES
030664  010360  000144                  MOV     R3,MAXSEC(R0)   ;SWAP MIN. TO MAX.
030670  010460  000146                  MOV     R4,MINSEC(R0)   ;SWAP MAX. TO MIN.
```

```
64
65 050676  005737  001422          78:     TST     TSTERR          ;ARE YOU TESTING ANYWHERE ON MEDIA ?
66 050700  001016                          BNE     90              ;BR IF YES
67 050702  026060  000136  000196          CMP     MTNCYL(R0),FE1(R0)      ;IS MIN. CYLINDER < 1ST FE CYLINDER ?
68 050710  103003                          BHIS    81              ;BR IF NO
69 050712  016060  000196  000136          MOV     FE1(R0),MTNCYL(R0)      ;RESET MINIMUM CYLINDER ADDRESS
70 050720  026060  000134  000196  80:     CMP     MXXCYL(R0),FE1(R0)      ;IS MAX. CYLINDER < 1ST FE CYLINDER ?
71 050726  103003                          BHIS    90              ;BR IF NO
72 050730  016060  000150  000134          MOV     CYLLMT(R0),MXXCYL(R0)   ;RESET MAXIMUM CYLINDER ADDRESS
73
74 050736  016060  000136  000012  90:     MOV     MTNCYL(R0),DCYL(R0)     ;INITIAL CYLINDER VALUE
75 050744  116060  000142  000011          MOVB    MINTRK(R0),DTRK(R0)     ;INITIAL TRACK VALUE
76 050752  116060  000146  000010          MOVB    MINSEC(R0),DSEC(R0)     ;INITIAL SECTOR VALUE
77 050760  012604                          MOV     (SP)+,R4        ;POP STACK INTO R4
   050762  012603                          MOV     (SP)+,R3        ;POP STACK INTO R3
78 050764  000207                          RTS     PC              ;RETURN
```

```
                                        ;GET TIME TO GET THE ADDRESS LIMITS FOR THE CURRENT DRIVE TYPE
                                        ;CALL
                                        ;       JSR     PC,GETLMT       ;CALL ROUTINE
                                        ;
                                        ;R0 = DPB ADDRESS BEFORE CALLING THE ROUTINE

                                        GETLMT:
                                                MOV     R1,-(SP)        ;PUSH R1 ON STACK
                                                CLR     R1              ;START FOR 0
                                                MOV     @#40 .PTR(R0)   ;GET ADDRESS OF 1ST PT CYLINDER
                                                MOV     @#40 ,CYLMT(R0)         ;ASSURE CYLINDER LIMIT FOR DRV?
                                                MOV     @#0 .TRKMT(R0)  ;ASSURE TRACK LIMIT     -
                                                MOV     @#1 .SECLMT(R0) ;ASSURE SECTOR LIMIT    -
                                                MOVB    (R0),R1         ;GET DRIVE NUMBER
                                                CMPB    R1,@#NDRV(R1)   ;IS DRIVE ON DRV? ?
                                                BLO     10              ;BR IF YES
                                                MOV     @#40 .PTR(R0)   ;GET ADDRESS OF 1ST PT CYLINDER
                                                MOV     @#40 ,CYLMT(R0)         ;GET CYLINDER LIMIT FOR DRV?.
                                                MOV     @#1 .TRKMT(R0)  ;GET TRACK LIMIT        -
                                                MOV     @#0 .SECLMT(R0) ;GET SECTOR LIMIT.      -
                                        10:
                                                MOV     (SP)+,R1        ;POP STACK INTO R1
                                                RTS     PC              ;RETURN
```

```
                        ;[illegible] TO GET THE DRIVE SERIAL NUMBER FROM MEMORY REGISTER
                        ;THIS NUMBER IS CONTAINED IN THE REGISTER ARE ONLY THE 4 LSB'S OF THE
                        ;SERIAL NUMBER
                        ;CALL:
                        ;       MOV     #DRV,R0         ;DRIVE NUMBER IN
                        ;       JSR     PC,GETSER       ;CALL ROUTINE
                        ;
                        ;R0 = DRIVE NUMBER BEFORE CALLING THE ROUTINE

011070                  GETSER:
011070  010046                 MOV     R0,-(SP)        ;PUSH R0 ON STACK
011072  010146                 MOV     R1,-(SP)        ;PUSH R1 ON STACK
011074  010246                 MOV     R2,-(SP)        ;PUSH R2 ON STACK
011076  010546                 MOV     R5,-(SP)        ;PUSH R5 ON STACK
011100  010002                 MOV     R0,R2           ;GET INDEX TO DRV
011102  006727  001562         ASH     #3,R2           ;SAVE ALL REGISTERS
011106  012702  000006         MOV     #6,R2           ;6 DIGITS TO STORE
011112  016001  000216         MOV     DRVSER(R0),R1   ;SERIAL NUMBER
011116  000305                 SWAB    R5              ;R5=0
011120  006101          10$:    ROL     R1              ;PUT THE NEXT DIGIT
011122  006105                 ROL     R5              ;INTO R5
011124  006101                 ROL     R1
011126  006105                 ROL     R5
011130  006101                 ROL     R1
011132  006105                 ROL     R5
011134  006101                 ROL     R1
011136  006105                 ROL     R5
011140  005302  000002         DEC     R2              ;DONE 1ST DIGIT
011144  110546  000160         MOVB    R5,@#160000(R0) ;SAVE DRIVE SERIAL NUMBER DIGIT
011150  005200                 INC     R0              ;GET NEXT INDEX FOR DRIVE SERIAL NUMBER
011152  005702                 TST     R2              ;ALL DIGITS TYPED?
011154  001360                 BNE     10$             ;NO -- BRANCH
011156  012605                 MOV     (SP)+,R5        ;POP STACK INTO R5
011160  012602                 MOV     (SP)+,R2        ;POP STACK INTO R2
011162  012601                 MOV     (SP)+,R1        ;POP STACK INTO R1
011164  012600                 MOV     (SP)+,R0        ;POP STACK INTO R0
011166  000207                 RTS     PC              ;RETURN
```

```
                                                .SBTTL  PARAMETER ENTRY ROUTINE
                                                ;
                                                ;PARAMETER ENTRY ROUTINE
                                                ;CALL:
                                                ;       MOV     #ADDR,R5        ;PARAMETER TABLE ADDRESS
                                                ;       JSR     PC,PARENT       ;GET THE PARAMETERS
                                                ;
 8 031170  010506                               PARENT: MOV     R5,-(SP)        ;SAVE THE PARAMETER TABLE ADDRESS
 9 031172  005037  001330                       10:     CLR     CFLAG           ;CLEAR THE 'CONTROL C' FLAG
10 031176  012537  031206                               MOV     (R5)+,20        ;ADDRESS OF PARAMETER NAME
11 031202  001451                               1$:     BEQ     9$              ;BR IF AT END OF TABLE
12 031204  104401                                       TYPE                    ;TYPE THE PARAMETER NAME
13 031206  000000                               20:     .WORD   0               ;ADDRESS OF PARAMETER NAME TEXT
14 031210  012502                                       MOV     (R5)+,R2        ;MAXIMUM PARAMETER VALUE
15 031212  012503                                       MOV     (R5)+,R3        ;ADDRESS OF PARAMETER
16 031214  011546                                       MOV     (R5),-(SP)      ;CURRENT VALUE OF PARAMETER
17 031216  004737  031412                               JSR     PC,$BIN2O       ;CONVERT IT TO DECIMAL
18 031222  004737  032436                               JSR     PC,$OPNSL       ;TYPE IT (LEFT JUSTIFIED)
19                                                                              ;TYPE THE CURRENT VALUE OF THE PARAMETER
20 031226  104001  056611                               TYPE    .BLNKS1         ;TYPE 1 BLANK
21 031232  104401  047447                               TYPE    .QUES           ;'?'
22 031236  104401  056611                               TYPE    .BLNKS1         ;TYPE 1 BLANK
23 031242  104411                                       RDLIN                   ;READ THE KEYBOARD
24 031244  010601                                       MOV     (SP),R1         ;INPUT ASCII STRING ADDRESS
25 031246  005737  001334                               TST     CFLAG           ;WAS (^C) TYPED?
26 031252  001021                                       BNE     6$              ;BR IF IT WAS
27 031254  004537  033254                               JSR     R5,CK.DIG       ;CHECK THE DIGIT(S)
   031260  031176                                       10                      ;CARRIAGE RETURN ONLY ENTERED
   031262  031326                                       70                      ;PERIOD ONLY ENTERED
   031264  031300                                       40                      ;ILLEGAL INPUT
   031266  031274                                       30                      ;TERMINATED WITH A CARRIAGE RETURN
   031270  031300                                       40                      ;TERMINATED WITH A ','
   031272  031312                                       50                      ;TERMINATED WITH A '.'
28 031274  010215                               30:     MOV     R2,(R5)         ;MOVE NEW VALUE TO PARAMETER LOCATION
29 031276  000737                                       BR      1$              ;GET MORE PARAMETERS
30 031300  104401  060163                       40:     TYPE    .BADENT         ;'BAD ENTRY'
31 031304  162705  000006                               SUB     #6,R5           ;DECREMENT THE TABLE POINTER
32 031310  000732                                       BR      1$              ;TRY AGAIN
33 031312  010215                               50:     MOV     R2,(R5)         ;NEW VALUE
34 031314  000404                                       BR      7$              ;EXIT
35 031316  005037  001334                       60:     CLR     CFLAG           ;CLEAR THE 'CONTROL C' FLAG
36 031322  011603                                       MOV     (SP),R3         ;RELOAD THE PARAMETER TABLE ADDRESS
37 031324  000724                                       BR      1$              ;TRY AGAIN
38 031326  005726                               70:     TST     (SP)+           ;CORRECT THE STACK POINTER
39 031330  000207                               7$:     RTS     PC              ;RETURN
```

```
                                        ;TYPEOUT ASSIGN/DROP ERROR MESSAGE
                                        ;CALL:
                                        ;       MOV     @MESADR,ASNMSG  ;ERROR MESSAGE ADDRESS
                                        ;       JSR     PC,ASNERR
                                        ;       RETURN
                                        ;
 031332  104401  001203        ASNERR: TYPE    ,$CRLF          ;CR-LF
 031336  104401  056647                TYPE    ,QUES           ;'?'
 031342  104401  056661                TYPE    ,DRVMSG         ;TYPE 'DRIVE'
 031346  010446                        MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
                                                                ;;TYPE DRIVE NUMBER
 031350  104403                        TYPOS                   ;;GO TYPE--OCTAL ASCII
 031352     002                        .BYTE   2               ;;TYPE 2 DIGIT(S)
 031353     000                        .BYTE   0               ;;SUPPRESS LEADING ZEROS
 031354  104401                        TYPE                    ;TYPE SPECIFIC MESSAGE
 031356  000000        ASNMSG: .WORD   0               ;MESSAGE ADDRESS
 031360  000207                RTS     PC
                                        ;DROP DRIVE IF A FATAL ERROR OCCURS
                                        ;CALL:
                                        ;       JSR     PC,DROP
                                        ;       RETURN
                                        ;
 031362  005037  043626        DROP:   CLR     PERM            ;CLR PERMENANT ERROR FLAG
 031366  005004                        CLR     R4              ;CLEAR R4 FOR DRIVE NUMBER
 031370  111004                        MOVB    (R0),R4         ;MOVE DRIVE NUMBER TO R4
 031372  146437  040630  001542        BICB    ATABIT(R4),ASNLST       ;REMOVE DRIVE FROM ASSIGNED LIST
 031400  146437  040630  032100        BICB    ATABIT(R4),AUTLST       ;DELETE DRIVE FROM AUTO ASSIGN LIST
 031406  006304                        ASL     R4              ;MAKE DRIVE NUMBER INTO A TABLE INDEX
 031410  010064  001544                MOV     R0,DDRVS(R4)    ;PUT DRIVE IN DROP LIST
 031414  104401  001203                TYPE    ,$CRLF
 031420  104401  057641                TYPE    ,DROPNG         ;TYPE '?FATAL OR EXCESSIVE ERRORS'
 031424  104401  057676                TYPE    ,MSGON          ;TYPE 'ON'
 031430  104401  056661                TYPE    ,DRVMSG         ;TYPE 'DRIVE'
 031434  006204                        ASR     R4              ;DRIVE NUMBER
 031436  010446                        MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
                                                                ;;TYPE DRIVE NUMBER
 031440  104403                        TYPOS                   ;;GO TYPE--OCTAL ASCII
 031442     002                        .BYTE   2               ;;TYPE 2 DIGIT(S)
 031443     000                        .BYTE   0               ;;SUPPRESS LEADING ZEROS
 031444  104401  001203                TYPE    ,$CRLF          ;CR-LF
 031450  000207                1$:     RTS     PC

                                        ;ROUTINE TO DROP DRIVE IF ERRORS BECOMES EXCESSIVE

 031452  032777  000020  147474 ABNRML: BIT     @#SW04,@SWR     ;SEE IF SWITCH 4 SET
 031460  001006                        BNE     1$              ;BR IF IT'S SET
 031462  023760  001456  000102        CMP     MAXER,$TOTAL(R0)        ;CHECK TOTAL ERROR VALUE
 031470  101002                        BHI     1$              ;BR IF ERRORS DO NOT EXCEED MAX
 031472  000137  031362                JMP     DROP            ;DROP THE DRIVE
 031476  000207                1$:     RTS     PC              ;RETURN
```

```
1
2                       ;ROUTINE TO CHECK FOR END OF PASS AND END OF TEST
```

```
1                                       .SBTTL  END OF PASS ROUTINE

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

031500                                  $EOP:
031500  010446                                  MOV     R4,(SP)         ;SAVE R4
031502  111004                          1$:     MOVB    (R0),R4         ;MOVE DRIVE NUMBER

031504  105737  001150                          TSTB    $AUTOB          ;RUNNING IN AUTO MODE ?
031510  001410                                  BEQ     2$              ;BR IF NO
031512  136437  040630  032100                  BITB    ATABIT(R4),AUTLST       ;IS DRIVE ALREADY ASSIGNED TO AUTO LIST ?
031520  001110                                  BNE     6$              ;BR IF YES
031522  156437  040630  032100                  BISB    ATABIT(R4),AUTLST       ;ADD DRIVE TO AUTO ASSIGN LIST
031530  000441                                  BR      3$

031532  026037  000114  001474  2$:     CMP     $PASSC(R0),PASSES       ;SEE IF AT END OF TEST
031540  103435                                  BLO     3$              ;BR IF NOT
031542  032777  000020  147404                  BIT     @SW04,@SWR      ;TYPE END OF TEST MESSAGE (SW04=1) ?
031550  001031                                  BNE     3$              ;BR IF NO
031552  104401  001203                          TYPE    .$CRLF          ;CR-LF
031556  104401  001203                          TYPE    .$CRLF          ;CR-LF
031562  104401  057407                          TYPE    .DASH13         ;TYPE '-------------'
031566  104401  057477                          TYPE    .MSGEOT         ;TYPE 'END OF TEST-----------------EOT'
031572  146437  040630  001542                  BICB    ATABIT(R4),ASNLST       ;DELETE DRIVE FROM ASSIGNED LIST
031600  006304                                  ASL     R4              ;MAKE DRIVE NUMBER INTO TABLE INDEX
031602  010064  001544                          MOV     R0,DDRVS(R4)    ;PUT BLOCK ADDRESS INTO DROP LIST
031606  105737  001542                          TSTB    ASNLST          ;ALL DRIVES ARE DROPPED ?
031612  001062                                  BNE     7$              ;BR IF NO
031614  005237  001216                          INC     $DEVCT          ;INCREMENT DEVICE COUNT
031620  005237  001214                          INC     $PASS           ;INCRFMENT THE PASS COUNT
031624  042737  100000  001214                  BIC     @100000,$PASS   ;AVOID NEGATIVE NUMBER
031632  000452                                  BR      7$

031634  032777  000400  147312  3$:     BIT     @SW08,@SWR      ;INHIBIT END OF PASS TYPEOUT (SW08=1) ?
031642  001022                                  BNE     4$              ;BR IF YES
031644  104401  001203                          TYPE    .$CRLF          ;CR LF
031650  104401  001203                          TYPE    .$CRLF          ;CR LF
031654  104401  057407                          TYPE    .DASH13         ;TYPE '-------------'
031660  104401  057451                          TYPE    .MSGEOP         ;TYPE 'END OF PASS'
031664  104401  001203                          TYPE    .$CRLF          ;CR-LF
031670  004737  026226                          JSR     PC,$TIME        ;TYPE ELAPSED TIME
031674  104401  057401                          TYPE    .DASH5          ;TYPE '---  '
031700  104401  057425                          TYPE    .MSGSUM         ;TYPE 'SUMMARY'
031704  004737  025122                          JSR     PC,ONESUM       ;TYPE ONE DRIVE SUMMARY

031710  010346                          4$:     MOV     R3,-(SP)        ;SAVE R3
031712  010004                                  MOV     R0,R4           ;DRIVE'S BLOCK ADDRESS
031714  062704  000036                          ADD     #$RDPAS,R4      ;ADD THE STARTING ADDR OF SECTIONS TO CLEAR
031720  012703  000016                          MOV     #<$OPERC-$RDPAS>/2,R3   ;NUMBER OF LOCNS TO BE CLEARED
031724  005024                          5$:     CLR     (R4)+           ;CLEAR LOCATIONS '$RDPAS' - '$OPERC' IN DPB
031726  162703  000002                          SUB     #2,R3           ;DONE CLEARING YET ?
031732  001374                                  BNE     5$              ;BR IF NO
031734  012603                                  MOV     (SP)+,R3        ;RESTORE R3
031736  005260  000114                          INC     $PASSC(R0)      ;INCREMENT THE PASS COUNT
```

END OF PASS ROUTINE

```
031742  105737  001150          6$:     TSTB    $AUTOB          ;RUNNING IN AUTO MODE ?
031746  001404                          BEQ     7$              ;BR IF NO
031750  023737  001542  032100          CMP     ASNLST,AUTLST   ;HAVE ALL DRIVES COMPLETED PASS IN AUTO MODE ?
031756  001402                          BEQ     8$              ;BR IF YES
031760  012604                  7$:     MOV     (SP)+,R4        ;RESTORE R4
031762  000207                          RTS     PC              ;RETURN

031764  005237  032100          8$:     INC     AUTLST          ;CLEAR AUTO ASSIGN LIST FOR NEXT PASS AND
031770  001375                          BNE     8$              ;WAIT FOR TTY
031772  005237  001216                  INC     $DEVCT          ;INCREMENT DEVICE COUNT
031776  005237  001214                  INC     $PASS           ;;INCREMENT THE PASS NUMBER
032002  042737  100000  001214          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
032010  005327                          DEC     (PC)+           ;;LOOP?
032012  000001                  $EOPCT: .WORD   1
032014  003013                          BGT     $DOAGN          ;;YES
032016  012737                          MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
032020  000001                  $ENDCT: .WORD   1
032022  032012                          $EOPCT
032024  013700  000042          $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
032030  001405                          BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
032032  000005                          RESET                   ;;CLEAR THE WORLD
032034  004710                  $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
032036  000240                          NOP                     ;;SAVE ROOM
032040  000240                          NOP                     ;;FOR
032042  000240                          NOP                     ;;ACT11
032044                          $DOAGN:
032044  000137                          JMP     @(PC)+          ;;RETURN
032046  032050                  $RTNAD: .WORD   RTURN
2
3 032050  005037  177776        RTURN:  CLR     PS              ;SET PRIORITY TO 0
4 032054  012706  001100                MOV     #STACK,SP       ;RESTORE STACK
5 032060  005237  001212                INC     $TESTN          ;INCREMENT THE TEST NUMBER IN THE MAIL BOX
6 032064  004737  033510                JSR     PC,$TKINT       ;MAKE SURE KEYBOARD INTERRUPT AND
7 032070  004737  024574                JSR     PC,CKCLK        ;SYSTEM CLOCK ARE STILL ON.
8 032074  000137  006340                JMP     MAIN            ;RETURN TO LOOP
9
10 032100  000000         AUTLST: .WORD   0               ;AUTO ASSIGN LIST (USED IN AUTO RUN MODE)
```

```
                                        ;ROUTINE TO GET THE REMAINDER OF THE RANDOM NUMBER
                                        ;CALL:
                                        ;       MOV     NUMBER,R5       ;DIVISOR INTO R5
                                        ;       JSR     PC,GETREM
                                        ;       RETURN                  ;REMAINDER IS IN R5

032102  013746  037306          GETREM: MOV     $LONUM,-(SP)    ;STORE RANDOM NUMBER ON THE STACK FOR DIVIDE
032106  013746  037304                  MOV     $HINUM,(SP)     ;UPPER PART
032112  010546                          MOV     R5,(SP)         ;PUT THE DIVISOR ONTO THE STACK
032114  004737  032126                  JSR     PC,$DIV         ;DIVIDE THE RANDOM NUMBERS
032120  012605                          MOV     (SP)+,R5        ;PUT THE REMAINDER INTO R5
032122  005726                          TST     (SP)+           ;ADJUST THE STACK POINTER
032124  000207                          RTS     PC

                                        .SBTTL  INTEGER DIVIDE ROUTINE

                                        ;;*************************************************************
                                        ;*THIS ROUTINE WILL DIVIDE A 32-BIT TWO'S COMPLEMENT INTEGER
                                        ;*DIVIDEND BY A 16-BIT TWO'S COMPLEMENT INTEGER DIVISOR GIVING
                                        ;*A 16-BIT TWO'S COMPLEMENT INTEGER QUOTIENT AND A 16-BIT REMAINDER.
                                        ;*DIVISION WILL BE PERFORMED SO THAT THE REMAINDER IS OF THE
                                        ;*SAME SIGN AS THE DIVIDEND.
                                        ;*CALL:
                                        ;*      MOV     LOW DIVIDEND,-(SP)      ;;THE HIGH DIVIDEND MUST BE < 1/2
                                        ;*      MOV     HIGH DIVIDEND,-(SP)     ;;AS LARGE AS THE DIVISOR
                                        ;*      MOV     DIVISOR,-(SP)
                                        ;*      JSR     PC,$DIV
                                        ;*      RETURN                          ;;QUOTIENT & REMAINDER ARE ON THE STACK
                                        ;*
                                        ;*      STACK   NO ERROR        OVERFLOW        DIVIDE BY ZERO
                                        ;*      -----   --------        --------        ---- ---------
                                        ;*      TOP     REMAINDER       ALL ZEROS       ALL ONES
                                        ;*      +2      QUOTIENT        ALL ZEROS       ALL ONES
                                        ;*
                                        ;*
                                        ;*NOTE: THIS ROUTINE WILL LINK TO THE DIVISION SUBROUTINE ('M.DPID').

032126  104412                  $DIV:   SAVREG                  ;STORE R0 - R5
032130  016605  000026                  MOV     26(SP),R5       ;DIVISOR
032134  005004                          CLR     R4              ;OTHER DIVISOR WORD
032136  016602  000030                  MOV     30(SP),R2       ;UPPER DIVIDEND WORD
032142  016603  000032                  MOV     32(SP),R3       ;LOWER DIVIDEND WORD
032146  005000                          CLR     R0              ;CLEAR OTHER DIVIDEND REGISTERS
032150  005001                          CLR     R1
032152  004737  032250                  JSR     PC,M.DPID       ;GO TO THE DIVIDE ROUTINE
032156  010166  000030                  MOV     R1,30(SP)       ;REMAINDER ON THE STACK
032162  010366  000032                  MOV     R3,32(SP)       ;QUOTIENT ON THE STACK
032166  104413                          RESREG                  ;RESTORE R0 - R5
032170  012616                          MOV     (SP)+,(SP)      ;MOVE RETURN UP THE STACK
032172  000207                          RTS     PC
```

```
                                .SBTTL  DOUBLE DIVIDE ROUTINE

                                ;;*****************************************************************
                                ;*THIS ROUTINE WILL DIVIDE A 32 BIT TWO'S COMPLEMENT INTEGER
                                ;*DIVIDEND BY A 32-BIT TWO'S COMPLEMENT INTEGER DIVISOR GIVING
                                ;*A 16-BIT TWO'S COMPLEMENT INTEGER QUOTIENT AND A 32-BIT REMAINDER.
                                ;*DIVISION WILL BE PERFORMED SO THAT THE REMAINDER IS OF THE
                                ;*SAME SIGN AS THE DIVIDEND.
                                ;*CALL:
                                ;*      MOV     LOW DIVIDEND,-(SP)      ;;THE HIGH DIVIDEND MUST BE < 1/2
                                ;*      MOV     HIGH DIVIDEND,-(SP)     ;;AS LARGE AS THE DIVISOR
                                ;*      MOV     LOW DIVISOR,-(SP)
                                ;*      MOV     HIGH DIVISOR,-(SP)
                                ;*      JSR     PC,$DBDIV
                                ;*      RETURN                          ;;QUOTIENT & REMAINDER ARE ON THE STACK
                                ;*
                                ;*      STACK   NO ERROR        OVERFLOW        DIVIDE BY ZERO
                                ;*        - -   ---- ---        --------        ------- - -
                                ;*      TOP     REMAINDER       ALL ZEROS       ALL ONES (MSD)
                                ;*      +2      REMAINDER       ALL ZEROS       ALL ONES (LSD)
                                ;*      +4      QUOTIENT        ALL ZEROS       ALL ONES
                                ;*
                                ;*
                                ;*NOTE: THIS ROUTINE WILL LINK TO THE DIVISION SUBROUTINE ('M.DPID').

032174  104412                  $DBDIV: SAVREG                          ;STORE R0 - R5
032176  016604  000026                  MOV     26(SP),R4               ;HIGH DIVISOR WORD
032202  016605  000030                  MOV     30(SP),R5               ;LOW DIVISOR WORD
032206  016602  000032                  MOV     32(SP),R2               ;UPPER DIVIDEND WORD
032212  016603  000034                  MOV     34(SP),R3               ;LOWER DIVIDEND WORD
032216  005000                          CLR     R0                      ;CLEAR OTHER DIVIDEND REGISTERS
032220  005001                          CLR     R1
032222  004737  032250                  JSR     PC,M.DPID               ;GO TO THE DIVIDE ROUTINE
032226  010066  000030                  MOV     R0,30(SP)               ;REMAINDER ON THE STACK (MSD)
032232  010166  000032                  MOV     R1,32(SP)               ;REMAINDER ON THE STACK (LSD)
032236  010366  000034                  MOV     R3,34(SP)               ;QUOTIENT ON THE STACK
032242  104413                          RESREG                          ;RESTORE R0 - R5
032244  012616                          MOV     (SP)+,(SP)              ;MOVE RETURN UP THE STACK
032246  000207                          RTS     PC
```

```
                                .SBTTL  DOUBLE PRECISION DIVISION SUBROUTINE
                                ;
                                ;CALL:
                                ;       JSR     PC,M.DPID
                                ;
                                ;       DIVIDEND - R0 R1 R2 R3 (R0=MSD)
                                ;       DIVISOR - R4 R5 (R4=MSD)
                                ;
                                ;RETURN
                                ;
                                ;       REMAINDER AFTER DIVISION - R0 R1 (R0=MSD)
                                ;       QUOTIENT AFTER DIVISION - R2-R3 (R2=MSD)
                                ;
032250  012746  000040  M.DPID: MOV     #40.,-(SP)      ;COUNTER FOR DIVISION CYCLES
032254  010446                  MOV     R4,-(SP)        ;HIGH ORDER
032256  010546                  MOV     R5,-(SP)        ;LOW ORDER DIVISOR TO THE STACK
032260  005466  000002          NEG     2(SP)           ;FORM NEGATIVE
032264  005416                  NEG     @SP             ;VERSION OF THE DIVISOR
032266  005666  000002          SBC     2(SP)
032272  061601                  ADD     @SP,R1
032274  005500                  ADC     R0              ;PERFORM THE INITIAL SUBTRACTION
032276  066600  000002          ADD     2(SP),R0
032302  103445                  BCS     M.DP50          ;IF CARRY THEN OVERFLOW HAS OCCURRED
032304  005046                  CLR     -(SP)           ;THIS IS A LONGER LASTING CARRY BIT
032306  006103          M.DP40: ROL     R3
032310  006102                  ROL     R2
032312  006101                  ROL     R1
032314  006100                  ROL     R0
032316  005716                  TST     @SP             ;TEST "CARRY" INDICATOR
032320  001410                  BEQ     M.DP41          ;IF NO "CARRY" THEN ADD ELSE SUBTRACT
032322  005016                  CLR     @SP             ;CLEAR UP FOR NEXT TIME
032324  066601  000002          ADD     2(SP),R1
032330  005500                  ADC     R0              ;ADD -(DIVISOR)
032332  005516                  ADC     @SP     ;  I    ;SET "CARRY"
032334  066600  000004          ADD     4(SP),R0;<-
032340  000404                  BR      M.DP42
032342  060501          M.DP41: ADD     R5,R1
032344  005500                  ADC     R0              ;ADD +(DIVISOR)
032346  005516                  ADC     @SP     ;  I    ;SET "CARRY"
032350  060400                  ADD     R4,R0   ;<-
032352  005516          M.DP42: ADC     @SP             ;SET 'CARRY"
032354  005716                  TST     @SP             ;TEST THE UPDATE INDICATOR
032356  001401                  BEQ     .+4     ;->     ;IF ZERO FORGET IT
032360  005203                  INC     R3      ;  I    ;NO CARRY POSSIBLE HERE
032362  005366  000006          DEC     6(SP)   ;<-     ;DECREMENT COUNTER
032366  003347                  BGT     M.DP40          ;BRANCH IF MORE TO DO
032370  006003                  ROR     R3
032372  103404                  BCS     M.DP44
032374  060501                  ADD     R5,R1
032376  005500                  ADC     R0
032400  060400                  ADD     R4,R0
032402  000241                  CLC
032404  006103          M.DP44: ROL     R3
032406  062706  000010          ADD     #10,SP          ;ADJUST STACK BY 4 WORDS
032412  000242                  CLV
032414  000207                  RTS     PC
032416  062706  000006  M.DP50: ADD     #6,SP
```

```
     58 032422  000262                        SEV
     59 032424  000207                        BIS     PC
```

```
                                        .SBTTL  SUPPRS  TYPE ASCIZ, REPLACE LEADING 0'S WITH BLANKS
                                        .SBTTL  SUPPRSL  TYPE ASCIZ, LEFT JUSTIFY
                                        ;
                                        ;;*****************************************************************
                                        ;CALL:
                                        ;       MOV     @#NUMADR,-(SP)  ;FIRST ADDRESS OF ASCIZ STRING
                                        ;       JSR     PC,SUPPRS
                                        ;  OR
                                        ;       MOV     @#NUMADR,-(SP)  ;FIRST ADDRESS OF ASCIZ STRING
                                        ;       JSR     PC,SUPPRSL
                                        ;
032426  010046                  SUPPRSL: MOV     R0,-(SP)        ;SAVE R0
032430  016600  000004                   MOV     4(SP),R0        ;GET POINTER TO MESSAGE
032434  005037  032516                   CLR     SUPR2
032440  000405                           BR      SUPR1
                                        ;
032442  010046                  SUPPRS:  MOV     R0,-(SP)        ;SAVE R0
032444  016600  000004                   MOV     4(SP),R0        ;GET POINTER TO MESSAGE
032450  010037  032516                   MOV     R0,SUPR2        ;GET POINTER FOR TYPING
032454                          SUPR1:
032454  105710                  10:      TSTB    (R0)            ;TEST FOR TERMINATOR
032456  001406                           BEQ     20              ;YES
032460  122710  000060                   CMPB    #'0,(R0)        ;IS THIS A "0" ?
032464  001006                           BNE     30              ;NO
032466  112720  000040                   MOVB    #40,(R0)+       ;REPLACE IT WITH A "BLANK"
032472  000770                           BR      10              ;NEXT CHAR.
032474  005300                  20:      DEC     R0              ;BACKUP 1
032476  112710  000060                   MOVB    #'0,(R0)        ;MAKE IT "0"
032502  005737  032516          30:      TST     SUPR2           ;LEFT JUSTIFY ?
032506  001002                           BNE     40              ;NO
032510  010037  032516                   MOV     R0,SUPR2        ;YES
032514  104401                  40:      TYPE
032516  000000                  SUPR2:   .WORD   0
032520  012600                           MOV     (SP)+,R0        ;RESTORE R0
032522  012616                           MOV     (SP)+,(SP)      ;RESTORE STACK
032524  000207                           RTS     PC
```

```
                                        .SBTTL      TYPE ASCIZ, REPLACE LEADING 0'S WITH BLANKS
                                        .SBTTL  $SUPRL   TYPE ASCIZ, LEFT JUSTIFY
                                
                                        ;;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                                        ;
                                        ;THIS ROUTINE IS SAME AS 'SUPPL' AND 'SUPPS', EXCEPT THAT IT
                                        ;WILL SUPPRESS THE ERROR TYPEOUT IF SW13=1. THIS IS ACCOMPLISHED BY
                                        ;USING THE TRAP CALL 'DISPLY', INSTEAD OF 'TYPE'.
                                        ;CALL:
                                        ;       MOV     @#LUNOR,-(SP)   ;FIRST ADDRESS OF ASCIZ STRING
                                        ;       JSR     PC,$SUPPS
                                        ;  OR
                                        ;       MOV     @#LUNOR,-(SP)   ;FIRST ADDRESS OF ASCIZ STRING
                                        ;       JSR     PC,$SUPRL
                                
016526 010046                           $SUPRL: MOV     R0,-(SP)        ;SAVE R0
016530 016600 000004                            MOV     4(SP),R0        ;GET POINTER TO MESSAGE
016534 005037 032616                            CLR     $SUPR2
032540 000405                                   BR      $SUPR1
                                
032542 010046                           $SUPPS: MOV     R0,-(SP)        ;SAVE R0
032544 016600 000004                            MOV     4(SP),R0        ;GET POINTER TO MESSAGE
032550 010037 032616                            MOV     R0,$SUPR2       ;GET POINTER FOR TYPING
                                        $SUPR1:
032554 105710                           10:     TSTB    (R0)            ;TEST FOR TERMINATOR
032556 001406                                   BEQ     20              ;YES
032560 122710 000060                            CMPB    #'0,(R0)        ;IS THIS A "0" ?
032564 001006                                   BNE     30              ;NO
032566 112720 000040                            MOVB    #40,(R0)+       ;REPLACE IT WITH A "BLANK"
032572 000770                                   BR      10              ;NEXT CHAR.
032574 005300                           20:     DEC     R0              ;BACKUP 1
032576 112710 000060                            MOVB    #'0,(R0)        ;MAKE IT "0"
032602 005737 032616                    30:     TST     $SUPR2          ;LEFT JUSTIFY ?
032606 001002                                   BNE     40              ;NO
032610 010037 032616                            MOV     R0,$SUPR2       ;YES
032614 104414                           40:     DISPLY                  ;TYPE, UNLESS SW13=1
032616 000000                           $SUPR2: .WORD   0
032620 012600                                   MOV     (SP)+,R0        ;RESTORE RC
032622 012616                                   MOV     (SP)+,(SP)      ;RESTORE STACK
032624 000207                                   RTS     PC
```

```
                                        ;ROUTINE TO REPLACE LEADING ZEROS IN A NUMERIC STRING WITH SPACES
                                        ;CALL:
                                        ;       MOV     ADDR,-(SP)      ;ADDRESS OF NUMBER (IN ASCII)
                                        ;       JSR     R5,REPLZ        ;REPLACE PRECEDING ZEROS WITH BLANKS
                                        ;       .WORD   N               ;'N' IS NUMBER OF DIGITS TO BE TYPED
                                        ;OR
                                        ;       MOV     ADDR,-(SP)      ;ADDRESS OF NUMBER (IN ASCII)
                                        ;       JSR     R5,FILLZ        ;TYPE PRECEDING ZEROS
                                        ;       .WORD   N               ;'N' IS NUMBER OF DIGITS TO BE TYPED

012626  005737  012734      FILLZ:  TST     FILLO           ;LEAVE ZEROS?

012632  010046              REPLZ:  MOV     R0,-(SP)        ;SAVE R0
012634  016600  000004              MOV     4(SP),R0        ;ADDRESS OF NUMBER TO R0
012640  005737  012734              TST     FILLO           ;LEAVE PRECEDING ZEROS ?
012644  001014                      BNE     3$              ;BR IF YES
012646  122710  000060      1$:     CMPB    #'0,(R0)        ;BYTE EQUAL TO ASCII '0' ?
012652  001006                      BNE     2$              ;BR IF NOT
012654  112710  000040              MOVB    #40,(R0)        ;REPLACE THE ZERO WITH A SPACE
012660  005200                      INC     R0              ;INCREMENT THE BYTE ADDRESS
012662  000771                      BR      1$              ;GO BACK AND LOOK FOR MORE LEADING ZEROS
012664  105710          2$:     TSTB    (R0)            ;SEE IF ZERO BYTE TERMINATOR
012666  001003                      BNE     3$              ;BR IF NOT
012670  005300                      DEC     R0              ;BACKUP STRING POINTER
012672  112710  000060              MOVB    #'0,(R0)        ;PUT A ZERO BACK IN
012676  016600  000004      3$:     MOV     4(SP),R0        ;PUT ADDRESS OF FIRST CHARACTER ON STACK
012702  105720          4$:     TSTB    (R0)+           ;SEE IF ZERO BYTE TERMINATOR
012704  001376                      BNE     4$              ;BR IF NOT
012706  005300                      DEC     R0              ;BACKUP STRING POINTER
012710  166500                      SUB     (R5),R0         ;ADJUST ADDRESS
012712  010037  012720              MOV     R0,5$           ;GET ADDRESS FOR TYPEOUT
012716  104401                      TYPE                    ;TYPE THE NUMBER
012720  000000          5$:     .WORD   0               ;ADDRESS OF NUMBER
012722  012600                      MOV     (SP)+,R0        ;POP STACK INTO R0
012724  012516                      MOV     (SP)+,(SP)      ;MOVE STACK
012726  005037  012734              CLR     FILLO           ;SET FILL FLAG
012732  000205                      RTS     R5              ;RETURN

012734  000000          FILLO:  .WORD   0               ;IF SET, LEAVE PRECEDING ZEROS FOR TYPE
```

```
                        ;THIS ROUTINE IS SAME AS .RDPLF AND .TYPLF, EXCEPT THAT IT
                        ;WILL SUPPRESS THE LEADING ZEROS OF SWR0-1. THIS ACCOMPLISHED BY
                        ;USING THEM CALL .DISPLY, INSTEAD OF .TYPE.
                        ;CALL:
                        ;       MOV     ADDR,-(SP)      ;ADDRESS OF NUMBER (IN ASCII)
                        ;       JSR     R5,.DSPLF       ;REPLACE PRECEDING ZEROS WITH BLANKS
                        ;       .WORD   N               ; N IS NUMBER OF DIGITS TO BE TYPED
                        ;OR
                        ;       MOV     ADDR,-(SP)      ;ADDRESS OF NUMBER (IN ASCII)
                        ;       JSR     R5,.DFILL       ;TYPE PRECEDING ZEROS
                        ;       .WORD   N               ; N IS NUMBER OF DIGITS TO BE TYPED
                        ;
000000  005267  000736  DFILL:: INC     FILL0           ;LEAVE ZEROS
                        ;
000004  010046  DSPLF:: MOV     R0,-(SP)        ;SAVE R0
000006  016400  000000          MOV     @4(SP),R0       ;ADDRESS OF NUMBER TO R0
000012  005737  000736          TST     FILL0           ;LEAVE PRECEDING ZEROS ?
000016  001010                  BNE     10              ;BR IF YES
000020  122710  000060  10:     CMPB    #'0,(R0)        ;BYTE EQUAL TO ASCII '0' ?
000024  001004                  BNE     20              ;BR IF NOT
000026  112710  000040          MOVB    #40,(R0)        ;REPLACE THE ZERO WITH A SPACE
000032  005200                  INC     R0              ;INCREMENT THE BYTE ADDRESS
000034  000771                  BR      10              ;GO BACK AND LOOK FOR MORE LEADING ZEROS
000036  105710  20:     TSTB    (R0)            ;SEE IF ZERO BYTE TERMINATOR
000040  001003                  BNE     30              ;BR IF NOT
000042  005300                  DEC     R0              ;BACKUP STRING POINTER
000044  112710  000060          MOVB    #'0,(R0)        ;PUT A ZERO BACK IN
000050  016400  000004  30:     MOV     4(SP),R0        ;PUT ADDRESS OF FIRST CHARACTER ON STACK
000054  105720  40:     TSTB    (R0)+           ;SEE IF ZERO BYTE TERMINATOR
000056  001376                  BNE     40              ;BR IF NOT
000060  005300                  DEC     R0              ;BACKUP STRING POINTER
000062  160500                  SUB     (R5),R0         ;ADJUST ADDRESS
000064  010037  033030          MOV     R0,$$$0         ;GET ADDRESS FOR TYPEOUT
000070  104414                  .DISPLY                 ;TYPE, UNLESS SW13=1
000072  000000          50:     .WORD   0               ;ADDRESS OF NUMBER
000074  012600                  MOV     (SP)+,R0        ;POP STACK INTO R0
000076  016605  000002          MOV     2(SP),(SP)      ;RESTORE STACK
000102  005037  000736          CLR     FILL0           ;RESET FILL FLAG
000106  000205                  RTS     R5              ;RETURN
```

```
                                        ;ROUTINE TO TYPE THE DRIVE SERIAL NUMBER IN DECIMAL
                                        ;CALL:
                                        ;       MOV     @DPB,R0         ;ADDRESS OF DRIVE PARAMETER BLOCK
                                        ;       JSR     PC,TYDRV        ;CALL ROUTINE
                                        ;   OR
                                        ;       MOV     @DPB,R0         ;ADDRESS OF DRIVE PARAMETER BLOCK
                                        ;       JSR     PC,TYPDRV       ;CALL ROUTINE(WITH NO HEADER MESSAGE)
                                        ;
                                        ;R0 = DPB ADDRESS BEFORE CALLING THE ROUTINE

033044  104401  060252          TYDRV:  TYPE    .DRVSN          ;TYPE 'DRV S/N:'
033050  104401  057152          TYPDRV: TYPE    .MSGPC          ;TYPE PC
033054  010037  033070                  MOV     R0,10           ;ADDRESS OF DPB
033060  062737  000160  033070          ADD     #16DRVSN,10     ;INDEX TO DRIVE SERIAL NUMBER
033066  104401                          TYPE                    ;TYPE THE DRIVE SERIAL NUMBER
033070  000000          10:     .WORD   0               ;ADDRESS OF DRIVE SERIAL NUMBER FIELD
033072  104401  057771                  TYPE    .PERIOD         ;TYPE '.'
033076  000207                          RTS     PC              ;RETURN

                                        ;ROUTINE TO TYPE ERRORS
                                        ;CALL
                                        ;       DISPLY                  ;MUST DEFINED IN 'TRAP' TABLE
                                        ;       MESADR                  ;ADDRESS OF MESSAGE
                                        ;       RETURN

033100  052777  020000  146046  @DSPLY: BIT     @#BIT13,@#SWR   ;INHIBIT ERROR TYPEOUT ?
033106  001004                          BNE     1$              ;BR IF YES
033110  005037  177776                  CLR     @#PS            ;SET PRIORITY TO ZERO
033114  000137  035732                  JMP     @#TYPE          ;TYPE THE MESSAGE
033120  062716  000002          1$:     ADD     #2,(SP)         ;INCREMENT THE RETURN
033124  000002                          RTI                     ;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

033126  121127  000060          CK.OCT: CMPB    (R1),#'0        ;LESS THAN ZERO?
033132  103407                          BLO     1$              ;YES -- BRANCH
033134  121127  000067                  CMPB    (R1),#'7        ;GREATER THAN SEVEN?
033140  101004                          BHI     1$              ;YES -- BRANCH
033142  111102                          MOVB    (R1),R2         ;GET THE CHARACTER
033144  042702  177770                  BIC     #^C7,R2         ;STRIP AWAY THE ASCII
033150  005725                          TST     (R5)+           ;ADJUST FOR RETURN
033152  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     R ,CK.DEC       ;CHECK THE CHARACTER
                                        ;       RETURN1                 ;NOT BETWEEN 0 AND 9
                                        ;       RETURN2                 ;BETWEEN 0 AND 9
                                        ;                               ;R2 = DIGIT

033154  121127  000060         CK.DEC: CMPB    (R1),#'0        ;LESS THAN ZERO?
033160  103407                         BLO     1$              ;YES -- BRANCH
033162  121127  000071                 CMPB    (R1),#'9        ;GREATER THAN NINE?
033166  101004                         BHI     1$              ;YES -- BRANCH
033170  111102                         MOVB    (R1),R2         ;GET THE CHARACTER
033172  042702  000060                 BIC     #'0,R2          ;STRIP AWAY THE ASCII
033176  005725                         TST     (R5)+           ;ADJUST FOR RETURN
033200  000205                 1$:     RTS     R5              ;RETURN

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

033202  105711                 CK.CHR: TSTB    (R1)            ;"CARRIAGE RETURN"?
033204  001417                         BEQ     3$              ;YES -- BRANCH
033206  121127  000054                 CMPB    (R1),#',        ;"COMMA"?
033212  001413                         BEQ     2$              ;YES -- BRANCH
033214  121127  000056                 CMPB    (R1),#'.        ;"PERIOD"?
033220  001407                         BEQ     1$              ;YES -- BRANCH
033222  004537  033154                 JSR     R5,CK.DEC       ;"DIGIT"?
033226  000410                         BR      4$              ;NO -- BRANCH
033230  004537  033126                 JSR     R5,CK.OCT       ;OCTAL ?
033234  005725                         TST     (R5)+           ;DIGIT BETWEEN 8-9
033236  005725                         TST     (R5)+           ;DIGIT BETWEEN 0-7
033240  005725                 1$:     TST     (R5)+           ;PERIOD
033242  005725                 2$:     TST     (R5)+           ;COMMA
033244  005725                 3$:     TST     (R5)+           ;CARRIAGE RETURN
033246  005201                         INC     R1              ;MOVE POINTER TO NEXT CHARACTER
033250  011505                 4$:     MOV     (R5),R5         ;UNKNOWN CHARACTER
033252  000205                         RTS     R5              ;RETURN
```

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

033254  010446                  CK.DIG: MOV     R4,-(SP)        ;SAVE R4
033256  010346                          MOV     R3,-(SP)        ;SAVE R3
033260  010246                          MOV     R2,-(SP)        ;SAVE THE MAX. SIZE ON THE STACK
033262  005002                          CLR     R2              ;START WITH 0
033264  005003                          CLR     R3
033266  005004                          CLR     R4
033270  004537  033202                  JSR     R5,CK.CHR       ;CHECK ONE CHARACTER
033274  033370                          6$                      ;ILLEGAL CHARACTER
033276  033376                          9$                      ;CARRIAGE RETURN
033300  033370                          6$                      ;"."
033302  033372                          7$                      ;"."
033304  033310                          1$                      ;DIGIT 0-7
033306  033310                          1$                      ;DIGIT 8-9
033310  062705  000004          1$:     ADD     @4,R5           ;STEP RETURN POINTER PAST "CR" & "PERIOD" RETURNS
033314  006303                  2$:     ASL     R3              ;INPUT NUMBER *2
033316  010346                          MOV     R3,-(SP)        ;SAVE *2
033320  006303                          ASL     R3              ;*4
033322  006303                          ASL     R3              ;*8
033324  062603                          ADD     (SP)+,R3        ;(*2)+(*8) = *10
033326  060203                          ADD     R2,R3           ;UPDATE THE INPUT NUMBER
033330  004537  033202                  JSR     R5,CK.CHR       ;CHECK ONE CHARACTER
033334  033374                          8$                      ;ILLEGAL CHARACTER
033336  033360                          5$                      ;CARRIAGE RETURN
033340  033356                          4$                      ;"."
033342  033350                          3$                      ;"."
033344  033314                          2$                      ;DIGIT 0-7
033346  033314                          2$                      ;DIGIT 8-9
033350  105711                  3$:     TSTB    (R1)            ;DOES A "CR" FOLLOW THE "PERIOD"
033352  001010                          BNE     8$              ;BR IF NOT
033354  005724                          TST     (R4)+           ;INCREMENT THE RETURN
033356  005724                  4$:     TST     (R4)+           ;INCREMENT THE RETURN
033360  005724                  5$:     TST     (R4)+           ;INCREMENT THE RETURN
033362  020316                          CMP     R3,(SP)         ;CHECK THE MAGNITUDE OF THE NUMBER
033364  101004                          BHI     9$              ;BR IF ENTERED NUMBER TOO LARGE
033366  000402                          BR      8$              ;BYPASS INCREMENT
033370  005725                  6$:     TST     (R5)+           ;INCREMENT RETURN PAST INVALID RETURN
033372  005725                  7$:     TST     (R5)+           ;INCREMENT RETURN
033374  060405                  8$:     ADD     R4,R5           ;SETUP RETURN POINTER
033376  010302                  9$:     MOV     R3,R2           ;ENTERED VALUE
033400  005726                          TST     (SP)+           ;CLEAN MAX. SIZE OFF OF STACK
033402  012603                          MOV     (SP)+,R3        ;RESTORE R3
033404  012604                          MOV     (SP)+,R4        ;RESTORE R4
033406  011505                          MOV     (R5),R5         ;GET RETURN ADDRESS
033410  000205                          RTS     R5              ;RETURN
```

```
                                        ;THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
                                        ;UNSIGNED DECIMAL ASCIZ NUMBER.
                                        ;CALL
                                        ;       MOV     NUMBER, (SP)    ;PUT THE NUMBER ON THE STACK
                                        ;       JSR     PC,$SB2D        ;CALL
                                        ;       RETURN                  ;ADDRESS OF THE 1ST ASCIZ CHAR IS ON THE STACK
                                        ;
                                        ;NOTE:  THE PROGRAM REQUIRES THIS FORM OF '$SB2D , NOT THE VERSION ON
                                        ;       THE SYSMAC LIBRARY, REV C AND LATER

033412  016637  000002  033436  $SB2D:  MOV     2(SP),1$        ;SAVE THE BINARY NUMBER
033420  012746  033436                  MOV     #1$, (SP)       ;SET THE POINTER
033424  004737  037404                  JSR     PC,$DB2D        ;CALL THE DOUBLE LENGTH CONVERT
033430  012666  000002                  MOV     (SP)+,2(SP)     ;PICKUP THE POINTER
033434  000207                          RTS     PC              ;RETURN
033436  000000  000000          1$:     .WORD   0,0

                                        ;THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
                                        ;UNSIGNED OCTAL ASCIZ NUMBER.
                                        ;CALL
                                        ;       MOV     NUMBER, (SP)    ;PUT THE NUMBER ON THE STACK
                                        ;       JSR     PC,$SB2O        ;CALL
                                        ;       RETURN                  ;ADDRESS OF THE 1ST ASCIZ CHAR IS ON THE STACK
                                        ;
                                        ;NOTE:  THE PROGRAM REQUIRES THIS FORM OF '$SB2O', NOT THE VERSION ON
                                        ;       THE SYSMAC LIBRARY, REV C AND LATER

033442  016637  000002  033466  $SB2O:  MOV     2(SP),1$        ;SAVE THE BINARY NUMBER
033450  012746  033466                  MOV     #1$,-(SP)       ;SET THE POINTER
033454  004737  037600                  JSR     PC,$DB2O        ;CALL THE DOUBLE LENGTH CONVERT
033460  012666  000002                  MOV     (SP)+,2(SP)     ;PICKUP THE POINTER
033464  000207                          RTS     PC              ;RETURN
033466  000000  000000          1$:     .WORD   0,0
```

```
  1                                     .SBTTL  TTY INPUT ROUTINE

                                        ;;**********************************************************************
                                        .ENABL  LSB
033472  000000                          $TKCNT: .WORD   0               ;;NUMBER OF ITEMS IN QUEUE
033474  000000                          $TKQIN: .WORD   0               ;;INPUT POINTER
033476  000000                          $TKQOUT: .WORD  0               ;;OUTPUT POINTER
033500                                  $TKQSRT: .BLKB  7               ;;TTY KEYBOARD QUEUE
        033507                          $TKQEND=.
                                        .EVEN

                                        ;*TK INITIALIZE ROUTINE
                                        ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
                                        ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
                                        ;
                                        ;*CALL:
                                        ;*      JSR     PC,$TKINT
                                        ;*      RETURN
                                        ;
033510  005037  033472                  $TKINT: CLR     $TKCNT          ;;CLEAR COUNT OF ITEMS IN QUEUE
033514  012737  033500  033474                  MOV     #$TKQSRT,$TKQIN ;;MOVE THE STARTING ADDRESS OF THE
033522  013737  033474  033476                  MOV     $TKQIN,$TKQOUT  ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
033530  012737  033560  000060                  MOV     #$TKSRV,@#TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
033536  012737  000200  000062                  MOV     #200,@#TKVEC+2  ;;"BR" LEVEL 4
033544  005777  145412                        TST     @$TKB           ;;CLEAR DONE FLAG
033550  012777  000100  145402                MOV     #100,@$TKS      ;;ENABLE TTY KEYBOARD INTERRUPT
033556  000207                          RTS     PC              ;;RETURN TO CALLER

                                        ;*TK SERVICE ROUTINE
                                        ;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
                                        ;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
                                        ;*IT IN THE QUEUE.
                                        ;*IF THE CHARACTER IS A "CONTROL-C" (^C) $TKINT IS CALLED AND
                                        ;*UPON RETURN EXIT IS MADE TO THE "CONTROL-C" RESTART ADDRESS (CTRAP)
                                        ;
033560  117746  145376                  $TKSRV: MOVB    @$TKB,-(SP)     ;;PICKUP THE CHARACTER
033564  042716  177600                          BIC     #^C177,(SP)     ;;STRIP THE JUNK
033570  021627  000021                          CMP     (SP),#$XON      ;;IS IT A RANDOM XON?
033574  001002                                  BNE     30$             ;;BRANCH IF NO
033576  005726                                  TST     (SP)+           ;;CLEAN RANDOM XON OFF STACK
033600  000002                                  RTI                     ;;RETURN
033602                                  30$:
033602  021627  000003                          CMP     (SP),#3         ;;IS IT A CONTROL C?
033606  001007                                  BNE     1$              ;;BRANCH IF NO
033610  104401  034715                          TYPE    ,$CNTLC         ;;TYPE A CONTROL-C (^C)
033614  004737  033510                          JSR     PC,$TKINT       ;;INIT THE KEYBOARD
033620  005726                                  TST     (SP)+           ;;CLEAN UP STACK
033622  000137  034756                          JMP     CTRAP           ;;CONTROL C RESTART
033626  021627  000007                  1$:     CMP     (SP),#7         ;;IS IT A CONTROL G?
033632  001004                                  BNE     2$              ;;BRANCH IF NO
033634  022737  000176  001154                  CMP     #SWREG,SWR      ;;IS SOFT-SWR SELECTED?
033642  001500                                  BEQ     6$              ;;GO TO SWR CHANGE

033644                                  2$:
033644  022737  000007  033472                  CMP     #7,$TKCNT       ;;IS THE QUEUE FULL?
033652  001004                                  BNE     3$              ;;BRANCH IF NO
033654  104401  001176                          TYPE    ,$BELL          ;;RING THE TTY BELL
```

```
033660  005726                          TST     (SP)+           ;;CLEAN CHARACTER OFF OF STACK
033662  000451                          BR      5$              ;;EXIT
033664  021627  000023          3$:     CMP     (SP),#23        ;;IS IT A CONTROL-S?
033670  001021                          BNE     32$             ;;BRANCH IF NO
033672  005077  145262                  CLR     @#TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
033676  005726                          TST     (SP)+           ;;CLEAN CHAR OFF STACK
033700  105777  145254          31$:    TSTB    @#TKS           ;;WAIT FOR A CHAR
033704  100375                          BPL     31$             ;;LOOP UNTIL ITS THERE
033706  117746  145250                  MOVB    @#TKB,-(SP)     ;;GET THE CHARACTER
033712  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
033716  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
033722  001366                          BNE     31$             ;;BRANCH IF NO
033724  012777  000100  145226          MOV     #100,@#TKS      ;;REENABLE TTY KEYBOARD INTERRUPTS
033732  000002                          RTI                     ;;RETURN
033734  005237  033472          32$:    INC     $TKCNT          ;;COUNT THIS CHARACTER
033740  021627  000140                  CMP     (SP),#140       ;;IS IT UPPER CASE?
033744  002405                          BLT     4$              ;;BRANCH IF YES
033746  021627  000175                  CMP     (SP),#175       ;;IS IT A SPECIAL CHAR?
033752  003002                          BGT     4$              ;;BRANCH IF YES
033754  042716  000040                  BIC     #40,(SP)        ;;MAKE IT UPPER CASE
033760  112677  177510          4$:     MOVB    (SP)+,@#TKQIN   ;;AND PUT IT IN QUEUE
033764  005237  033474                  INC     $TKQIN          ;;UPDATE THE POINTER
033770  023727  033474  033507          CMP     $TKQIN,#$TKQEND ;;GO OFF THE END?
033776  001003                          BNE     5$              ;;BRANCH IF NO
034000  012737  033500  033474          MOV     #$TKQSRT,$TKQIN ;;RESET THE POINTER
034006  000002                  5$:     RTI                     ;;RETURN

                                ;;**********************************************************************
                                ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
                                ;*ROUTINE IS ENTERED  FROM THE TRAP HANDLER, AND WILL
                                ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
                                ;*CALL WHEN OPERATING IN TTY INTERRUPT MODE.
034010  022737  000176  001154  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED
034016  001124                          BNE     15$             ;;EXIT IF NOT
034020  105777  145134                  TSTB    @#TKS           ;;IS A CHAR WAITING?
034024  100121                          BPL     15$             ;;IF NOT, EXIT
034026  117746  145130                  MOVB    @#TKB,-(SP)     ;;YES
034032  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT  7-BIT ASCII
034036  021627  000007                  CMP     (SP),#7         ;;IS IT A CONTROL-G?
034042  001300                          BNE     2$              ;;IF NOT, PUT IT IN THE TTY QUEUE
                                                                ;;AND EXIT

                                ;;**********************************************************************
                                ;*CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
                                ;*ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
                                ;*CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
034044  123727  001150  000001  6$:     CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
034052  001674                          BEQ     2$              ;;BRANCH IF YES
034054  005726                          TST     (SP)+           ;;CLEAR CONTROL-G OFF STACK
034056  004737  033510                  JSR     PC,$TKINT       ;;FLUSH THE TTY INPUT QUEUE
034062  005077  145072                  CLR     @#TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS
034066  112737  000001  001151          MOVB    #1,$INTAG       ;;SET INTERRUPT MODE INDICATOR

034074  104401  034727                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
034100  104401  034734  $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
034104  013746  000176                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
034110  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
```

```
034112  104401  034745                  TYPE    .$MNEW          ;.PROMPT FOR NEW SWR
034116  005046                  19$:     CLR     (SP)            ;;CLEAR COUNTER
034120  005046                          CLR     (SP)            ;;THE NEW SWR
034122  105777  145032          7$:     TSTB    @$TKS           ;;CHAR THERE?
034126  100375                          BPL     7$              ;;IF NOT TRY AGAIN

034130  117746  145026                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
034134  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII

034140  021627  000003                  CMP     (SP),#3         ;;IS IT A CONTROL-C?
034144  001015                          BNE     9$              ;;BRANCH IF NOT
034146  104401  034715                  TYPE    .$CNTLC         ;;YES, ECHO CONTROL-C (^C)
034152  062706  000006                  ADD     #6,SP           ;;CLEAN UP STACK
034156  123727  001151  000001          CMPB    $INTAG,#1       ;;REENABLE TTY KEYBOARD INTERRUPTS?
034164  001003                          BNE     8$              ;;BRANCH IF NO
034166  012777  000100  144764          MOV     #100,@$TKS      ;;ALLOW TTY KEYBOARD INTERRUPTS
034174  000137  034756          8$:     JMP     CTRAP           ;;CONTROL-C RESTART


034200  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
034204  001005                          BNE     10$             ;;BRANCH IF NOT
034206  104401  034722                  TYPE    .$CNTLU         ;;YES, ECHO CONTROL U (^U)
034212  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
034216  000737                          BR      19$             ;;LET'S TRY IT AGAIN


034220  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
034224  001022                          BNE     16$             ;;BRANCH IF NO
034226  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
034232  001403                          BEQ     11$             ;;BRANCH IF YES
034234  016677  000002  144712          MOV     2(SP),@SWR      ;;SAVE NEW SWR
034242  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
034246  104401  001203          14$:    TYPE    .$CRLF          ;;ECHO <CR> AND <LF>
034252  123727  001151  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
034260  001003                          BNE     15$             ;;BRANCH IF NOT
034262  012777  000100  144670          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
034270  000002          15$:    RTI                             ;;RETURN
034272  004737  036144          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
034276  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
034302  002420                          BLT     18$             ;;BRANCH IF YES
034304  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
034310  003015                          BGT     18$             ;;BRANCH IF YES
034312  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
034316  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
034322  001?03                          BEQ     17$             ;;BRANCH IF YES
034324  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
034326  006316                          ASL     (SP)            ;;    CHAR OVER TO MAKE
034330  006316                          ASL     (SP)            ;;    ROOM FOR NEW ONE.
034332  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
034336  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
034342  000667                          BR      7$              ;;GET THE NEXT ONE
034344  104401  001?02          18$:    TYPE    .$QUES          ;;TYPE ?<CR><LF>
034350  000720                          BR      20$             ;;SIMULATE CONTROL-U
                        .DSABL  LSB

                        ;;*****************************************************************
```

```
                                ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
                                ;*CALL:
                                ;*      RDCHR                   ;;GET A CHARACTER FROM THE QUEUE
                                ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
                                ;*                              ;;WITH PARITY BIT STRIPPED OFF
                                ;
034352  011646                 $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC AND
034354  016666  000004  000002          MOV     4(SP),2(SP)     ;;THE PS
034362  005066  000004                  CLR     4(SP)           ;;GET READY FOR A CHARACTER
034366  005046                          CLR      (SP)           ;;PUT NEW PS ON STACK
034370  012746  034376                  MOV     #64$,-(SP)      ;;PUT NEW PC ON  STACK
034374  000002                          RTI                     ;;POP NEW PC AND PS
034376                          64$:
034376  005737  033472          1$:     TST     $TKCNT          ;;WAIT ON A CHARACTER
034402  001775                          BEQ     1$
034404  005337  033472                  DEC     $TKCNT          ;;DECREMENT THE COUNTER
034410  117766  177062  000004          MOVB    @$TKQOUT,4(SP)  ;;GET ONE CHARACTER
034416  005237  033476                  INC     $TKQOUT         ;;UPDATE THE POINTER
034422  023727  033476  03350.          CMP     $TKQOUT,#$TKQEND ;;DID IT GO OFF OF THE END?
034430  001003                          BNE     2$              ;;BRANCH IF NO
034432  012737  033500  033476          MOV     #$TKQSRT,$TKQOUT ;;RESET THE POINTER
034440  000002                  2$:     RTI                     ;;RETURN
                                ;;**************************************************************
                                ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
                                ;*CALL:
                                ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
                                ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
                                ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
034442  010346                 $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
034444  005046                          CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
034446  012703  034676          1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
034452  022703  034715          2$:     CMP     #$TTYIN+15.,R3  ;;BUFFER FULL?
034456  101456                          BLOS    4$              ;;BR IF YES
034460  104410                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
034462  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
034464  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
034470  001022                          BNE     5$              ;;BR IF NO
034472  005716                          TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
034474  001007                          BNE     6$              ;;BR IF NO
034476  112737  000134  034674          MOVB    #'\,9$          ;;TYPE A BACK SLASH
034504  104401  034674                  TYPE    ,9$
034510  012716  177777                  MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
034514  005303                  6$:     DEC     R3              ;;BACKUP BY ONE
034516  020327  034676                  CMP     R3,#$TTYIN      ;;STACK EMPTY?
034522  103434                          BLO     4$              ;;BR IF YES
034524  111337  034674                  MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
034530  104401  034674                  TYPE    ,9$             ;;GO TYPE
034534  000746                          BR      2$              ;;GO READ ANOTHER CHAR.
034536  005716                  5$:     TST     (SP)            ;;RUBOUT KEY SET?
034540  001406                          BEQ     7$              ;;BR IF NO
034542  112737  000134  034674          MOVB    #'\,9$          ;;TYPE A BACK SLASH
034550  104401  034674                  TYPE    ,9$
034554  005016                          CLR     (SP)            ;;CLEAR THE RUBOUT KEY
034556  122713  000025          7$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
034562  001003                          BNE     8$              ;;BR IF NO
```

```
034564  104401  034722                  TYPE    .$CNTLU         ;;TYPE A CONTROL "U"
034570  000726                          BR      1$              ;;GO START OVER
034572  122715  000022          8$:     CMPB    #22.,(R3)       ;;IS CHARACTER A "^R"?
034576  001011                          BNE     3$              ;;BRANCH IF NO
034600  105013                          CLRB    (R3)            ;;CLEAR THE CHARACTER
034602  104401  001203                  TYPE    .$CRLF          ;;TYPE A "CR" & "LF"
034606  104401  034676                  TYPE    .$TTYIN         ;;TYPE THE INPUT STRING
034612  000717                          BR      2$              ;;GO PICKUP ANOTHER CHACTER
034614  104401  001202          4$:     TYPE    .$QUES          ;;TYPE A '?'
034620  000712                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
034622  111337  034674          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
034626  104401  034674                  TYPE    ,9$
034632  122723  000015                  CMPB    #15.,(R3)+      ;;CHECK FOR RETURN
034636  001305                          BNE     2$              ;;LOOP IF NOT RETURN
034640  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
034644  104401  001204                  TYPE    ,$LF            ;;TYPE A LINE FEED
034650  005726                          TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
034652  012603                          MOV     (SP)+,R3        ;;RESTORE R3
034654  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
034656  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
034664  012766  034676  000004          MOV     #$TTYIN,4(SP)
034672  000002                          RTI                     ;;RETURN
034674     000          9$:     .BYTE   0                       ;;STORAGE FOR ASCII CHAR. TO TYPE
034675     000                  .BYTE   0                       ;;TERMINATOR
034676                  $TTYIN: .BLKB   15.                     ;;RESERVE 15. BYTES FOR TTY INPUT
034715     136     103     015  $CNTLC: .ASCIZ  /^C/<15><12>    ;;CONTROL "C"
034722     136     125     015  $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL "U"
034727     136     107     015  $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL "G"
034734     015     012     123  $MSWR:  .ASCIZ  <15><12>/SWR = /
034745     040     040     116  $MNEW:  .ASCIZ  /  NEW = /
2
3                               ;THIS ROUTINE WILL PROCESS THE (^C) CHARACTER
4
5  034756  012737  000001  001354 CTRAP:  MOV     #1,CFLAG        ;SET THE 'CONTROL C' FLAG
6  034764  005237  033472          INC     $TKCNT          ;COUNT THIS CHARACTER
7  034770  112737  000015  176476  MOVB    #15,@$TKQIN     ;PUT 'RETURN' CHARACTER IN QUEUE
8  034776  005237  033474          INC     $TKQIN          ;UPDATE THE POINTER
9  035002  023727  033474  033507  CMP     $TKQIN,#$TKQEND ;GO OFF THE END ?
10 035010  001003                  BNE     1$              ;BR IF YES
11 035012  012737  033500  033474  MOV     #$TKQSRT,$TKQIN ;RESET THE POINTER
12 035020  000002          1$:     RTI                     ;RETURN
```

```
1                                       .SBTTL  ERROR HANDLER ROUTINE

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

035022  105037  035410          $ERROR: CLRB    IBSAVE          ;;CLEAR THE ITEM BYTE SAVE LOCATION
035026  104407                          CPSWR                   ;;TEST FOR CHANGE IN SOFT SWR
035030  010337  001316                  MOV     R3,ATTN         ;SAVE THE ATTENTION REGISTER CONTENTS
035034  010137  001320                  MOV     R1,DRVNO        ;DRIVE NUMBER
035040  032777  020000  144106          BIT     #SW13,@SWR      ;INHIBIT PRINTOUTS ?
035046  001004                          BNE     .+12            ;BR IF YES
035050  104401  001203                  TYPE    ,$CRLF          ;CR-LF
035054  104401  001203                  TYPE    ,$CRLF          ;CR-LF
035060  004737  026226                  JSR     PC,$TIME        ;TYPE ELAPSED TIME
035064  105237  001117          70:     INCB    $ERFLG          ;;SET THE ERROR FLAG
035070  001775                          BEQ     70              ;;DON'T LET THE FLAG GO TO ZERO
035072  013777  001116  144056          MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
035100  032777  002000  144046          BIT     #BIT10,@SWR     ;;BELL ON ERROR?
035106  001402                          BEQ     10              ;;NO - SKIP
035110  104401  001176                  TYPE    ,$BELL          ;;RING BELL
035114  005237  001126          10:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
035120  011637  001132                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
035124  162737  000002  001132          SUB     #2,$ERRPC
035132  117737  143774  0011            MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
035140  032777  001000  144000          BIT     #BIT09,@SWR     ;;SEE IF LOOP ON ERROR IS SET
035146  001060                          BNE     10048           ;;BRANCH AROUND ROUTINE IF SO
035150  122737  000177  001130          CMPB    #177,$ITEMB     ;;SEE IF THIS IS THE POWER FAIL CALL
035156  001454                          BEQ     10048           ;;BRANCH AROUND ROUTINE IF IT IS
035160  105737  035410                  TSTB    IBSAVE          ;;SEE IF THIS IS THE 2ND ERROR CALL IN THIS ROUTINE
035164  001047                          BNE     10038           ;;BRANCH IF SO
035166  022737  177777  035406          CMP     #-1,CPSAVE      ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
035174  001445                          BEQ     10048           ;;BRANCH IF SO
035176  013746  000004                  MOV     ERRVEC,-(SP)    ;;SAVE CONTENTS OF ERROR VECTOR
035202  012737  035220  000004          MOV     #10008,ERRVEC   ;;SETUP 'TRAP' RETURN ADDRESS
035210  013737  177766  035406          MOV     177766,CPSAVE   ;;MOVE CPU ERROR REGISTER TO CPSAVE FOR TEST
035216  000406                          BR      10018
035220  012737  177777  035406  10008:  MOV     #-1,CPSAVE      ;;SET CPU ERROR REGISTER TIMEOUT INDICATOR
035226  012716  035234                  MOV     #10018,(SP)     ;;SETUP RETURN ADDRESS
035232  000002                          RTI
035234  012637  000004          10018:  MOV     (SP)+,ERRVEC    ;;RESTORE CONTENTS OF ERROR VECTOR

035240  022737  177777  035406  10028:  CMP     #-1,CPSAVE      ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
035246  001420                          BEQ     10048           ;;BRANCH IF SO
035250  032737  000001  035406          BIT     #BIT00,CPSAVE   ;;SEE IF POWER MONITOR BIT IS SET IN CPU ERR REG
035256  001414                          BEQ     10048           ;;BRANCH IF OK
035260  042737  000001  177766          BIC     #BIT00,177766   ;;CLEAR THE BIT FOUND SET
035266  113737  001130  035410          MOVB    $ITEMB,IBSAVE   ;;MAKE IBSAVE NON-ZERO FOR DUAL ERROR CALL
035274  112737  000177  001130          MOVB    #177,$ITEMB     ;;SET $ITEMB TO SPECIAL POWER FAIL POINTER
035302  000402                          BR      10048           ;;BRANCH OVER IBSAVE CLEARING
```

```
035306  105037  035410          10038:  CLRB    IBSAVE          ;;CLEAR IBSAVE SO 2ND TIME THROUGH EXITS
035310                          10048:
035310  032777  020000  149636          BIT     #BIT13,@#SWR    ;;SKIP TYPEOUT IF SET
035316  001008                          BNE     20$             ;;SKIP TYPEOUTS
035320  004737  035412                  JSR     PC,@ERRTYP      ;;GO TO USER ERROR ROUTINE
035324  104401  001203                  TYPE    ,#CRLF
035330                          20$:
035330  122737  000001  001226          CMPB    #APTFLW,#FLW    ;;RUNNING IN APT MODE
035336  001007                          BNE     21$             ;;NO,SKIP APT ERROR REPORT
035340  113737  001150  035352          MOVB    @ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
035346  004737  036756                  JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
035352     000                  21$:     .BYTE   0
035353     000                          .BYTE   0
035354  000777                  22$:     BR      22$             ;;APT ERROR LOOP
035356  105737  035410          21$:    TSTB    IBSAVE          ;;SEE IF IBSAVE IS LOADED
035362  001005                          BNE     31$             ;;BRANCH IF NOT - NO HALT ON PWR FDN BIT ERROR
035364  005777  143564                  TST     @#SWR           ;;HALT ON ERROR
035370  100002                          BPL     31$             ;;SKIP IF CONTINUE
035372  000000                          HALT                    ;;HALT ON ERROR.
035374  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT.SWR
035376                          31$:
035376  105737  035410                  TSTB    IBSAVE          ;;SEE IF ITEM BYTE SAVE LOCATION HAS AN ERROR CALL
035402  001230                          BNE     7$              ;;BRANCH BACK TO CALL ORIGINAL ERROR
035404  000002                          RTI                     ;;RETURN
035406  000000          CPSAVE: .WORD   0               ;;LOCATION TO SAVE CPU ERROR REG CONTENTS
035410  000000          IBSAVE: .WORD   0               ;;LOCATION TO SAVE ITEM BYTE
```

```
                                .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE

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

035412                          ERRTYP:
035412  104401  001203                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
035416  010046                          MOV     R0,-(SP)        ;;SAVE R0
035420  005000                          CLR     R0              ;;PICKUP THE ITEM INDEX
035422  153700  001130                  BISB    @#ITEMCB,R0
035426  001004                          BNE     10              ;;IF ITEM NUMBER IS ZERO, JUST
                                                                ;;TYPE THE PC OF THE ERROR
035430  013706  001132                  MOV     @#ERRPC,-(SP)   ;;SAVE ERRPC FOR TYPEOUT
                                                                ;;ERROR ADDRESS
035434  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
035436  000457                          BR      66              ;;GET OUT
035440  122700  000177          10:     CMPB    #177,R0         ;;SEE IF THIS ERROR CALL IS SPECIAL POWER FAIL CALL
035444  001006                          BNE     10000           ;;BRANCH IF NOT
035446  013737  001212  035730          MOV     @#TESTN,PFTSTN  ;;GET TEST NUMBER
035454  012700  035570                  MOV     #PFECH,R0       ;;MOVE POWER FAIL ERROR CALL TABLE TO R0
035460  000406                          BR      10010           ;;BRANCH TO CALL ERROR
035462  005300                  10000:  DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
035464  006300                          ASL     R0              ;;      WORK FOR THE ERROR TABLE
035466  006300                          ASL     R0
035470  006300                          ASL     R0
035472  062700  003560                  ADD     #ERRTB,R0       ;;FORM TABLE POINTER
035476  012037  035506          10010:  MOV     (R0)+,20        ;;PICKUP "ERROR MESSAGE" POINTER
03'502  001404                          BEQ     30              ;;SKIP TYPEOUT IF NO POINTER
03'504  104401                          TYPE                    ;;TYPE THE "ERROR MESSAGE"
03'506  000000                  20:     .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
03'510  104401  001203                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
03'514  012037  035524          30:     MOV     (R0)+,40        ;;PICKUP "DATA HEADER" POINTER
03'520  001404                          BEQ     50              ;;SKIP TYPEOUT IF 0
03'522  104401                          TYPE                    ;;TYPE THE "DATA HEADER"
03'524  000000                  40:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
03'526  104401  001203                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
035532  011000                  50:     MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
035534  001004                          BNE     70              ;;GO TYPE THE DATA
035536  012600                  60:     MOV     (SP)+,R0        ;;RESTORE R0
035540  104401  001203                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
035544  000207                          RTS     PC              ;;RETURN
035546                          70:
035546  013046                          MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
035550  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
035552  005710                          TST     (R0)            ;;IS THERE ANOTHER NUMBER?
035554  001770                          BEQ     60              ;;BR IF NO
035556  104401  035564                  TYPE    ,80             ;;TYPE TWO(2) SPACES
035562  000771                          BR      70              ;;LOOP
035564     040     040     000  80:     .ASCIZ  /  /            ;;TWO(2) SPACES
                                        .EVEN
035570  035600  035662  035714  PFECH:  PFECH1,PFECH2,PFECH3,PFECH4 ;;WORDS DEFINING TABLES BELOW
035600     120     117     127  PFECH1: .ASCIZ  ?POWER MONITOR BIT IN CPU ERROR REGISTER FOUND SET?
035662     124     105     123  PFECH2: .ASCIZ  ?TESTNO  ERR PC  CPUERREG?
                                        .EVEN
035714  035730  001132  035406  PFECH3: .WORD   PFTSTN,@ERRPC,CPSAVE,0
```

```
[illegible]  000     000     000  [illegible]  0.0.0.0
[illegible] 50  000000           [illegible]  WORD  0

;#CONTAINS TEST NUMBER FOR PF DIT ERROR
```

```
                                .SBTTL  TYPE ROUTINE

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

035732  105737  001173          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
035736  100002                          BPL     1$              ;;BR IF YES
035740  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
035742  000430                          BR      3$              ;;LEAVE
035744  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
035746  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
035752  122737  000001  001226          CMPB    @APTENV,$ENV    ;;RUNNING IN APT MODE
035760  001011                          BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
035762  132737  000100  001227          BITB    @APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
035770  001405                          BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
035772  010037  036002                  MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
035776  004737  036746                  JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
036002  000000                  61$:    .WORD   0               ;;MESSAGE ADDRESS
036004  132737  000040  001227  62$:    BITB    @APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
036012  001003                          BNE     60$             ;;YES,SKIP TYPE OUT
036014  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
036016  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
036020  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
036022  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
036024  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
036030  000002                          RTI                     ;;RETURN
036032  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
036036  001430                          BEQ     8$
036040  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
036044  001006                          BNE     5$
036046  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
036050  104401                          TYPE                    ;;TYPE A CR AND LF
036052  001203                          $CRLF
036054  105037  036262                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
036060  000755                          BR      2$              ;;GET NEXT CHARACTER
036062  004737  036144          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
036066  123726  001172          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
036072  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
036074  013746  001170                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
036100  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
036104  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
036106  004737  036144                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
036112  105337  036262                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
036116  000770                          BR      7$              ;;LOOP
```

```
                                ;HORIZONTAL TAB PROCESSOR

036120  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
036124  004737  036144          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
036130  132737  000007  036262          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
036136  001372                          BNE     9$              ;;TAB STOP
036140  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
036142  000724                          BR      2$              ;;GET NEXT CHARACTER
036144                          $TYPEC: .
036144  105777  143010                  TSTB    @$TKS           ;;CHAR IN KYBD BUFFER?
036150  100022                          BPL     10$             ;;BR IF NOT
036152  017746  143004                  MOV     @$TKB,-(SP)     ;;GET CHAR
036156  042716  177600                  BIC     #177600,(SP)    ;;STRIP EXTRANEOUS BITS
036162  122716  000023                  CMPB    #$XOFF,(SP)     ;;WAS CHAR XOFF
036166  001012                          BNE     102$            ;;BR IF NOT
036170                          101$:
036170  105777  142764                  TSTB    @$TKS           ;;WAIT FOR CHAR
036174  100375                          BPL     101$
036176  117716  142760                  MOVB    @$TKB,(SP)      ;;GET CHAR
036202  042716  177600                  BIC     #177600,(SP)    ;;STRIP IT
036206  122716  000021                  CMPB    #$XON,(SP)      ;;WAS IT XON?
036212  001366                          BNE     101$            ;;BR IF NOT
036214                          102$:
036214  005726                          TST     (SP)+           ;;FIX STACK
036216                          10$:
036216  105777  142742                  TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
036222  100375                          BPL     10$
036224  116677  000002  142734          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
036232  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
036240  001003                          BNE     1$              ;;BRANCH IF NO
036242  105037  036262                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
036246  000406                          BR      $TYPEX          ;;EXIT
036250  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
036256  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
036260  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
036262  000000          $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
036264  000207          $TYPEX: RTS     PC
```

```
                                .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE

                                ;;*****************************************************************
                                ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
                                ;*OCTAL (ASCII) NUMBER AND TYPE IT.
                                ;*$TYPOS  -ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
                                ;*CALL:
                                ;*      MOV     NUM, (SP)       ;;NUMBER TO BE TYPED
                                ;*      TYPOS                   ;;CALL FOR TYPEOUT
                                ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
                                ;*      .BYTE   M               ;;M=1 OR 0
                                ;*                                      ;;1=TYPE LEADING ZEROS
                                ;*                                      ;;0=SUPPRESS LEADING ZEROS
                                ;*
                                ;*$TYPON- --ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
                                ;*$TYPOS OR $TYPOC
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                ;*      TYPON                   ;;CALL FOR TYPEOUT
                                ;*
                                ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                ;*      TYPOC                   ;;CALL FOR TYPEOUT

036266  017646  000000         $TYPOS: MOV     @(SP), (SP)     ;;PICKUP THE MODE
036272  116637  000001 036511          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
036300  112637  036513                 MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
036304  062716  000002                 ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
036310  000406                         BR      $TYPON
036312  112737  000001 036511 $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
036320  112737  000006 036513         MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
036326  112737  000005 036510 $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
036334  010346                       MOV     R3,-(SP)        ;;SAVE R3
036336  010446                       MOV     R4,-(SP)        ;;SAVE R4
036340  010546                       MOV     R5,-(SP)        ;;SAVE R5
036342  113704  036513               MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
036346  005404                       NEG     R4
036350  062704  000006               ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
036354  110437  036512               MOVB    R4,$OMODE       ;;SAVE IT FOR USE
036360  113704  036511               MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
036364  016605  000012               MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
036370  005003                       CLR     R3              ;;CLEAR THE OUTPUT WORD
036372  006105                1$:    ROL     R5              ;;ROTATE MSB INTO "C"
036374  000404                       BR      3$              ;;GO DO MSB
036376  006105                2$:    ROL     R5              ;;FORM THIS DIGIT
036400  006105                       ROL     R5
036402  006105                       ROL     R5
036404  010503                       MOV     R5,R3
036406  006103                3$:    ROL     R3              ;;GET LSB OF THIS DIGIT
036410  105337  036512               DECB    $OMODE          ;;TYPE THIS DIGIT?
036414  100016                       BPL     7$              ;;BR IF NO
036416  042703  177770               BIC     #177770,R3      ;;GET RID OF JUNK
036422  001002                       BNE     4$              ;;TEST FOR 0
036424  005704                       TST     R4              ;;SUPPRESS THIS 0?
036426  001403                       BEQ     5$              ;;BR IF YES
036430  005204                4$:    INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
```

```
036432  052703  000060              BIS     # 0,R3          ;;MAKE THIS DIGIT ASCII
036436  052703  000040      5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
036442  110337  036506              MOVB    R3,8$           ;;SAVE FOR TYPING
036446  104401  036506              TYPE    ,8$             ;;GO TYPE THIS DIGIT
036452  105337  036510      7$:     DECB    $OCNT           ;;COUNT BY 1
036456  003347                      BGT     2$              ;;BR IF MORE TO DO
036460  002402                      BLT     6$              ;;BR IF DONE
036462  005204                      INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
036464  000744                      BR      2$              ;;GO DO THE LAST DIGIT
036466  012605              6$:     MOV     (SP)+,R5        ;;RESTORE R5
036470  012604                      MOV     (SP)+,R4        ;;RESTORE R4
036472  012603                      MOV     (SP)+,R3        ;;RESTORE R3
036474  016666  000002  000004      MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
036502  012616                      MOV     (SP)+,(SP)
036504  000002                      RTI                     ;;RETURN
036506     000              8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
036507     000                      .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
036510     000      $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
036511     000      $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
036512  000000      $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
```

```
                                .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

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

036514                          $TYPDS:
036514  010046                         MOV     R0,-(SP)        ;;PUSH R0 ON STACK
036516  010146                         MOV     R1,-(SP)        ;;PUSH R1 ON STACK
036520  010246                         MOV     R2,-(SP)        ;;PUSH R2 ON STACK
036522  010346                         MOV     R3,-(SP)        ;;PUSH R3 ON STACK
036524  010546                         MOV     R5,-(SP)        ;;PUSH R5 ON STACK
036526  012746  020200                 MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
036532  016605  000020                 MOV     20(SP),R5       ;;GET THE INPUT NUMBER
036536  100004                         BPL     1$              ;;BR IF INPUT IS POS.
036540  005405                         NEG     R5              ;;MAKE THE BINARY NUMBER POS.
036542  112766  000055  000001         MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
036550  005000                 1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
036552  012703  036730                 MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
036556  112723  000040                 MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
036562  005002                 2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
036564  016001  036720                 MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
036570  160105                 3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
036572  002402                         BLT     4$              ;;BR IF DONE
036574  005202                         INC     R2              ;;INCREASE THE BCD DIGIT BY 1
036576  000774                         BR      3$
036600  060105                 4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
036602  005702                         TST     R2              ;;CHECK IF BCD DIGIT=0
036604  001002                         BNE     5$              ;;FALL THROUGH IF 0
036606  105716                         TSTB    (SP)            ;;STILL DOING LEADING 0'S?
036610  100407                         BMI     7$              ;;BR IF YES
036612  106316                 5$:     ASLB    (SP)            ;;MSD?
036614  103003                         BCC     6$              ;;BR IF NO
036616  116663  000001  177777         MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
036624  052702  000060         6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
036630  052702  000040         7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
036634  110223                         MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
036636  005720                         TST     (R0)+           ;;JUST INCREMENTING
036640  020027  000010                 CMP     R0,#10          ;;CHECK THE TABLE INDEX
036644  002746                         BLT     2$              ;;GO DO THE NEXT DIGIT
036646  003002                         BGT     8$              ;;GO TO EXIT
036650  010502                         MOV     R5,R2           ;;GET THE LSD
036652  000764                         BR      6$              ;;GO CHANGE TO ASCII
036654  105726                 8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
036656  100003                         BPL     9$              ;;BR IF NO
036660  116663  177777  177776         MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
036666  105013                 9$:     CLRB    (R3)            ;;SET THE TERMINATOR
036670  012605                         MOV     (SP)+,R5        ;;POP STACK INTO R5
036672  012603                         MOV     (SP)+,R3        ;;POP STACK INTO R3
036674  012602                         MOV     (SP)+,R2        ;;POP STACK INTO R2
036676  012601                         MOV     (SP)+,R1        ;;POP STACK INTO R1
```

CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```
036700  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
036702  104401  036730                  TYPE    .$DBLK          ;;NOW TYPE THE NUMBER
036706  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
036714  012616                          MOV     (SP)+,(SP)
036716  000002                          RTI                     ;;RETURN TO USER
036720  023420                  $DTBL:  10000.
036722  001750                          1000.
036724  000144                          100.
036726  000012                          10.
036730                          $DBLK:  .BLKW   4
```

```
                                .SBTTL  APT COMMUNICATIONS ROUTINE

                                ;;*************************************************************
036740  112737  000001  037204  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
036746  112737  000001  037202  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
036754  000403                        BR      $ATYC
036756  112737  000001  037204  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
036764                          $ATYC:
036764  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
036766  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
036770  105737  037202                TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
036774  001450                        BEQ     5$              ;;IF NOT:  BR
036776  122737  000001  001226        CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
037004  001031                        BNE     3$              ;;IF NOT:  BR
037006  132737  000100  001227        BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
037014  001425                        BEQ     3$              ;;IF NOT:  BR
037016  017600  000004                MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
037022  062766  000002  000004        ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
037030  005737  001206          1$:   TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
037034  001375                        BNE     1$              ;;IF NOT:  WAIT
037036  010037  001222                MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
037042  105720                  2$:   TSTB    (R0)+           ;;FIND END OF MESSAGE
037044  001376                        BNE     2$
037046  163700  001222                SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
037052  006200                        ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
037054  010037  001224                MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
037060  012737  000004  001206        MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
037066  000413                        BR      5$
037070  017637  000004  037114  3$:   MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
037076  062766  000002  000004        ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
037104  013746  177776                MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
037110  004737  035732                JSR     PC,$TYPE        ;;CALL TYPE MACRO
037114  000000                  4$:   .WORD   0
037116                          5$:
037116  105737  037204          10$:  TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
037122  001416                        BEQ     12$             ;;IF NOT:  BR
037124  005737  001226                TST     $ENV            ;;RUNNING UNDER APT?
037130  001413                        BEQ     12$             ;;IF NOT:  BR
037132  005737  001206          11$:  TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
037136  001375                        BNE     11$             ;;IF NOT:  WAIT
037140  017637  000004  001210        MOV     @4(SP),$FATAL   ;;GET ERROR #
037146  062766  000002  000004        ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
037154  005237  001206                INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
037160  105037  037204          12$:  CLRB    $FFLG           ;;CLEAR FATAL FLAG
037164  105037  037203                CLRB    $LFLG           ;;CLEAR LOG FLAG
037170  105037  037202                CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
037174  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
037176  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
037200  000207                        RTS     PC              ;;RETURN
037202     000          $MFLG:  .BYTE   0               ;;MESSG. FLAG
037203     000          $LFLG:  .BYTE   0               ;;LOG FLAG
037204     000          $FFLG:  .BYTE   0               ;;FATAL FLAG
                                .EVEN
        000200          APTSIZE = 200
        000001          APTENV  = 001
        000100          APTSPOOL= 100
        000040          APTCSUP = 040
```

```
                                .SBTTL  RANDOM NUMBER GENERATOR ROUTINE

                                ;;*****************************************************************
                                ;*THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
                                ;*WITH A RANGE OF 0 TO 2(+33)-1.
                                ;*CALL:
                                ;*      JSR     PC,$RAND        ;;CALL THE ROUTINE
                                ;*      RETURN                  ;;RETURN HERE THE RANDOM
                                ;*                              ;;NUMBER WILL BE IN
                                ;*                              ;;$HINUM,$LONUM

037206                          $RAND:
037206  010046                         MOV     R0,-(SP)        ;;PUSH R0 ON STACK
037210  010146                         MOV     R1,-(SP)        ;;PUSH R1 ON STACK
037212  010246                         MOV     R2,-(SP)        ;;PUSH R2 ON STACK
037214  013700  037306                 MOV     $LONUM,R0       ;;SET R0 WITH LOW
037220  013701  037304                 MOV     $HINUM,R1       ;;SET R1 WITH HIGH
037224  012702  177771                 MOV     #-7,R2          ;;SET SHIFT COUNT
037230  006300                  1$:    ASL     R0              ;;SHIFT R0 LEFT AND
037232  006101                         ROL     R1              ;;ROTATE CARRY INTO R1 AND
037234  005202                         INC     R2              ;;CHECK FOR DONE
037236  001374                         BNE     1$              ;;CONTINUE SHIFT LOOP
037240  063700  037306                 ADD     $LONUM,R0       ;;ADD NUMBER TO MAKE X 129
037244  005501                         ADC     R1              ;;PROPOGATE CARRY
037246  063701  037304                 ADD     $HINUM,R1       ;;ADD NUMBER TO MAKE X 129
037252  062700  001057                 ADD     #1057,R0        ;;ADD LOW CONSTANT
037256  005501                         ADC     R1              ;;PROPOGATE CARRY
037260  062701  047401                 ADD     #47401,R1       ;;ADD HIGH CONSTANT
037264  010037  037306                 MOV     R0,$LONUM       ;;SAVE R0
037270  010137  037304                 MOV     R1,$HINUM       ;;SAVE R1
037274  012602                         MOV     (SP)+,R2        ;;POP STACK INTO R2
037276  012601                         MOV     (SP)+,R1        ;;POP STACK INTO R1
037300  012600                         MOV     (SP)+,R0        ;;POP STACK INTO R0
037302  000207                         RTS     PC              ;;RETURN
037304  176543                  $HINUM: .WORD   176543
037306  123456                  $LONUM: .WORD   123456
```

```
                                .SBTTL  SAVE AND RESTORE RO R5 ROUTINES

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

037310                          $SAVREG:
037310  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
037312  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
037314  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
037316  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
037320  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
037322  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
037324  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
037330  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
037334  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
037340  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
037344  000002                          RTI

                                ;*RESTORE RO-R5
                                ;*CALL:
                                ;*      RESREG

037346                          $RESREG:
037346  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
037352  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
037356  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
037362  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
037366  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
037370  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
037372  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
037374  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
037376  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
037400  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
037402  000002                          RTI
```

```
                                .SBTTL  DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

                                ;;*****************************************************************
                                ;*THIS ROUTINE WILL CONVERT A 32 BIT BINARY NUMBER TO AN UNSIGNED
                                ;*DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
                                ;*POSITIVE.
                                ;*CALL
                                ;*      MOV     @PNTR,-(SP)     ;;POINTER TO LOW WORD OF BINARY NUMBER
                                ;*      JSR     PC,@#DB2D
                                ;*      RETURN                  ;;THE FIRST ADDRESS OF ASCIZ
                                ;;                              ;;IS ON THE STACK

037404  104412                  $DB2D:  SAVREG                  ;;SAVE REGISTERS
037406  016602  000002                  MOV     2(SP),R2        ;;PICKUP THE DATA POINTER
037412  012700  037564                  MOV     #$DECVL,R0      ;;GET ADDRESS OF "$DECVL" STRING
037416  010066  000002                  MOV     R0,2(SP)        ;;PUT ADDRESS OF ASCIZ STRING ON STACK
037422  012201                          MOV     (R2)+,R1        ;;PICKUP THE BINARY NUMBER
037424  012202                          MOV     (R2)+,R2
037426  012737  000012  037502          MOV     #10.,4$         ;;SET UP TO DO 10 CONVERSIONS
037434  012704  037514                  MOV     #$TNPWR,R4      ;;ADDRESS OF TEN POWER
037440  012705  037516                  MOV     #$TNPWR+2,R5
037444  005003                  1$:     CLR     R3              ;;CLEAR PARTIAL
037446  161401                  2$:     SUB     (R4),R1         ;;SUBTRACT TEN POWER
037450  005602                          SBC     R2
037452  161502                          SUB     (R5),R2
037456  002402                          BLT     3$              ;;BR IF TEN POWER TO LARGE
037460  005203                          INC     R3              ;;ADD 1 TO PARTIAL
037462  000772                          BR      2$              ;;LOOP
037464  062401                  3$:     ADD     (R4)+,R1        ;;RESTORE SUBTRACTED VALUE
037466  005502                          ADC     R2
037470  062402                          ADD     (R4)+,R2
037472  022525                          CMP     (R5)+,(R5)+     ;;MOVE TO NEXT TEN POWER
037474  052703  000060                  BIS     #'0,R3          ;;CHANGE PARTIAL TO ASCII
037500  110320                          MOVB    R3,(R0)+        ;;SAVE IT
037502  005327                          DEC     (PC)+           ;;DONE?
037504  000000                  4$:     .WORD   0
037506  001357                          BNE     1$              ;;BR IF NO
037510  105020                          CLRB    (R0)+           ;;TERMINATOR
037512  104413                          RESREG                  ;;RESTORE REGISTERS
037514  000207                          RTS     PC              ;;RETURN
037516  145000                  $TNPWR: 145000                  ;;1.0E09
037520  035632                          35632
037522  160400                          160400                  ;;1.0E08
037524  002765                          2765
037526  113200                          113200                  ;;1.0E07
037530  000230                          230
037532  041100                          041100                  ;;1.0E06
037534  000017                          17
037536  103240                          103240                  ;;1.0E05
037540  000001                          1
037542  023420                          23420                   ;;1.0E04
037544  000000                          0
037546  001750                          1750                    ;;1.0E03
037550  000000                          0
037552  000144                          144                     ;;1.0E02
037554  000000                          0
```

```
0?7?54  000012                          12                      ;;1.0E01
0?7?56  000000                          0
0?7?60  000001                          1                       ;;1.0E00
0?7?62  000000                          0
0?7?64                          $DECVL: .BLKB   12.             ;;RESERVE STORAGE FOR ASCIZ STRING
```

```
                                .SBTTL  DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE

                                ;;*****************************************************************
                                ;*THIS ROUTINE WILL CONVERT A 32 BIT UNSIGNED BINARY NUMBER TO AN
                                ;*UNSIGNED OCTAL ASCII NUMBER.
                                ;*CALL
                                ;*      MOV     @PNTR,-(SP)     ;;POINTER TO LOW WORD OF BINARY NUMBER
                                ;*      JSR     PC,@#OBJ20      ;;CALL THE ROUTINE
                                ;*      RETURN                  ;;THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK

037600  104412                  #OBJ20: SAVREG                  ;;SAVE ALL REGISTERS
037602  016601  000002                  MOV     2(SP),R1        ;;PICKUP THE POINTER TO LOW WORD
037606  012705  037717                  MOV     #OCTVL+13.,R5   ;;POINTER TO DATA TABLE
037612  012704  000014                  MOV     #12.,R4         ;;DO ELEVEN CHARACTERS
037616  012703  177770                  MOV     #^C7,R3         ;;MASK
037622  012100                          MOV     (R1)+,R0        ;;LOWER WORD
037624  012101                          MOV     (R1)+,R1        ;;HIGH WORD
037626  005002                          CLR     R2                TERMINATOR
037630  110245                  1$:     MOVB    R2,-(R5)        ;;PUT CHARACTER IN DATA TABLE
037632  010002                          MOV     R0,R2           ;;GET THIS DIGIT
037634  005304                          DEC     R4              ;;COUNT THIS CHARACTER
037636  003007                          BGT     3$              ;;BR IF NOT THE LAST DIGIT
037640  001405                          BEQ     2$              ;;BR IF IT IS THE LAST DIGIT
037642  005205                          INC     R5              ;;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
037644  010566  000002                  MOV     R5,2(SP)        ;;ASCII CHAR. & PUT IT ON THE STACK
037650  104413                          RESREG                  ;;RESTORE ALL REGISTERS
037652  000207                          RTS     PC              ;;RETURN TO USER
037654  006203                  2$:     ASR     R3              ;;POSITION THE MASK FOR THE LAST DIGIT
037656  006001                  3$:     ROR     R1              ;;POSITION THE BINARY NUMBER FOR
037660  006000                          ROR     R0              ;;      THE NEXT OCTAL DIGIT
037662  006001                          ROR     R1
037664  006000                          ROR     R0
037666  006001                          ROR     R1
037670  006000                          ROR     R0
037672  040302                          BIC     R3,R2           ;;MASK OUT ALL JUNK
037674  062702  000060                  ADD     #'0,R2          ;;MAKE THIS CHAR. ASCII
037700  000755                          BR      1$              ;;GO PUT IT IN THE DATA TABLE
037702                          #OCTVL: .BLKB   14.             ;;RESERVE DATA TABLE
```

```
                                .SBTTL  POWER DOWN AND UP ROUTINES

                                ;;*****************************************************************
                                ;POWER DOWN ROUTINE
037720  012737  040072  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
037726  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
037734  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
037736  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
037740  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
037742  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
037744  010446                        MOV     R4,-(SP)        ;;PUSH R4 ON STACK
037746  010546                        MOV     R5,-(SP)        ;;PUSH R5 ON STACK
037750  017746  141200                MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
037754  010637  040076                MOV     SP,$SAVR6       ;;SAVE SP
037760  012737  037772  000024        MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
037766  000000                        HALT
037770  000776                        BR      .-2             ;;HANG UP

                                ;;*****************************************************************
                                ;POWER UP ROUTINE
037772  012737  040072  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
040000  013706  040076                MOV     $SAVR6,SP       ;;GET SP
040004  005037  040076                CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
040010  005237  040076        1$:     INC     $SAVR6          ;;WAIT FOR THE INC
040014  001375                        BNE     1$              ;;OF  WORD
040016  005337  040100        2$:     DEC     PWRFLG          ;TTY WAIT LOOP & SET POWER FAIL FLAG
040022  003775                        BLE     2$              ;WAIT FOR FLAG= 1
040024  012677  141124                MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
040030  012605                        MOV     (SP)+,R5        ;;POP STACK INTO R5
040032  012604                        MOV     (SP)+,R4        ;;POP STACK INTO R4
040034  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
040036  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
040040  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
040042  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
040044  012737  037720  000024        MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
040052  012737  000340  000026        MOV     #340,@#PWRVEC+2 ;;PRIO:7
040060  104401                        TYPE                    ;;REPORT THE POWER FAILURE
040062  040102                $PWRMG: .WORD   MSGPWR          ;;POWER FAIL MESSAGE POINTER
040064  012716                        MOV     (PC)+,(SP)      ;;RESTART AT PWRUP
040066  040124                $PWRAD: .WORD   PWRUP           ;;RESTART ADDRESS
040070  000002                        RTI
040072  000000                $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
040074  000776                        BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
040076  000000                $SAVR6: 0                       ;;PUT THE SP HERE
 2
 3 040100  000000              PWRFLG: .WORD   0               ;POWER FAIL FLAG GOES HERE; FAILED= 1
 4
 5 040102      200      012      042  MSGPWR: .ASCIZ  <CRLF><LF>/"POWER UP" AT /
 6                                    .EVEN
 7
 8                                    ;THIS ROUTINE HANDLES THE RETURN ON POWER UP. WAIT THREE (3) MINUTES AND
 9                                    ;DO AN AUTO RE-START TO 'SIZMEM'.
10
11 040124  005227  000000        PWRUP:  INC     #0              ;TTY LOOP, WAIT FOR INCREMENT
12 040130  001375                        BNE     .-4             ;OF WORD
13 040132  000005                        RESET                   ;CLEAR THE WORLD
14 040134  005037  177776                CLR     PS              ;CLEAR PSW
```

POWER DOWN AND UP ROUTINES

```
 15 040140  012706  001100                 MOV     #STACK,SP       ;SETUP STACK POINTER
 16 040144  005037  001344                 CLR     SECOND          ;RESET SECOND COUNT
 17 040150  005037  001472                 CLR     INTRVL+2        ;RESET THE INTERVAL COUNT
 18 040154  004737  033510                 JSR     PC,$TKINT       ;MAKE SURE KEYBOARD INTERRUPT AND
 19 040160  004737  024574                 JSR     PC,CKCLK        ;SYSTEM CLOCK ARE STILL ON.
 20 040164  004737  026226                 JSR     PC,$TIME        ;TYPE ELAPSED TIME
 21 040170  104401  001203                 TYPE    ,$CRLF          ;CR-LF
 22 040174  005037  001334                 CLR     CFLAG           ;CLEAR (^C) FLAG
 23 040200  105737  001542                 TSTB    ASMLST          ;ANY DRIVES ASSIGNED ?
 24 040204  001414                        BEQ     2$              ;BR IF NO
 25 040206  104401  040256                 TYPE    ,MSWAIT         ;TYPE 'WAITING 3 MINUTES...TO START'
 26 040212  005737  001334           1$:   TST     CFLAG           ;WAS (^C) TYPED?
 27 040216  001007                        BNE     2$              ;BR IF YES
 28 040220  023727  001472  000003        CMP     INTRVL+2,#3     ;THREE MINUTES YET ?
 29 040226  002771                        BLT     1$              ;WAIT IF NOT
 30 040230  012737  000400  001332        MOV     #400,CHGADR     ;FUDGE 200 START
 31 040236  012705  001520           2$:   MOV     #ORDERQ,R5      ;CLEAR UP THE QUE AND BUFFER
 32 040242  005025                   3$:   CLR     (R5)+
 33 040244  022705  002056                 CMP     #BLKADR,R5      ;ALL DONE ?
 34 040250  001374                        BNE     3$              ;BRANCH IF NOT .
 35 040252  000137  005112                 JMP     SIZMEM          ;LOOP BACK
 36
 37 040256     200     124     131  MSWAIT: .ASCII  <CRLF>/TYPE ^C TO ABORT/
 38 040277     200     127     101          .ASCIZ  <CRLF>/WAITING 3 MINUTES...TO START/<CRLF>
 39                                 .EVEN
```

```
                                .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.

040336  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
040340  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
040344  005740                          TST     (R0)            ;;BACKUP BY 2
040346  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
040350  006300                          ASL     R0              ;;POSITION FOR INDEXING
040352  016000  040372                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
040356  000200                          RTS     R0              ;;GO TO ROUTINE


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

040360  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
040362  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
040370  000002                          RTI                     ;;RESTORE THE PSW

                                .SBTTL  TRAP TABLE

                                ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
                                ;*BY THE "TRAP" INSTRUCTION.
                                ;       ROUTINE
                                ;       -------
040372  040360                  $TRPAD: .WORD   $TRAP2
040374  035732                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
040376  036312                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
040400  036266                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
040402  036326                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
040404  036514                          $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)

040406  034100                          $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING

040410  034010                          $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
040412  034352                          $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
040414  034442                          $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
040416  037310                          $SAVREG ;;CALL=SAVREG   TRAP+12(104412) SAVE R0-R5 ROUTINE
040420  037346                          $RESREG ;;CALL=RESREG   TRAP+13(104413) RESTORE R0-R5 ROUTINE
2 040422  033100                        $DSPLY  ;;CALL=DISPLY   TRAP+14(104414) ROUTINE TO TYPE ERROR MESSAGES
3         000032                $TERM=.-$TRPAD
```

```
                                .SBTTL  RPO7 DRIVER

                                ;STORAGE FOR RPDS, RPER1, RPER2, AND RPER3

040424  000000  000000  000000  RPSTU0: .WORD   0,0,0,0        ;DS, ER1, ER2 & ER3 STORAGE FOR DRIVE 0
040434  000000  000000  000000  RPSTU1: .WORD   0,0,0,0        ;DS, ER1, ER2 & ER3 STORAGE FOR DRIVE 1
040444  000000  000000  000000  RPSTU2: .WORD   0,0,0,0        ;DS, ER1, ER2 & ER3 STORAGE FOR DRIVE 2
040454  000000  000000  000000  RPSTU3: .WORD   0,0,0,0        ;DS, ER1, ER2 & ER3 STORAGE FOR DRIVE 3
040464  000000  000000  000000  RPSTU4: .WORD   0,0,0,0        ;DS, ER1, ER2 & ER3 STORAGE FOR DRIVE 4
040474  000000  000000  000000  RPSTU5: .WORD   0,0,0,0        ;DS, ER1, ER2 & ER3 STORAGE FOR DRIVE 5
040504  000000  000000  000000  RPSTU6: .WORD   0,0,0,0        ;DS, ER1, ER2 & ER3 STORAGE FOR DRIVE 6
040514  000000  000000  000000  RPSTU7: .WORD   0,0,0,0        ;DS, ER1, ER2 & ER3 STORAGE FOR DRIVE 7

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

040524      000                 DRVACT: .BYTE   0               ;DRIVE 0
040525      000                         .BYTE   0               ;DRIVE 1
040526      000                         .BYTE   0               ;DRIVE 2
040527      000                         .BYTE   0               ;DRIVE 3
040530      000                         .BYTE   0               ;DRIVE 4
040531      000                         .BYTE   0               ;DRIVE 5
040532      000                         .BYTE   0               ;DRIVE 6
040533      000                         .BYTE   0               ;DRIVE 7

                                ;TABLE OF DRIVE STATUS INDICATORS (DRVSTA=8 BYTES)
                                        ;DRVSTA=0 IF DRIVE IS OFFLINE OR NONEXSITENT
                                        ;DRVSTA>0 IF DRIVE IS ONLINE
                                        ;DRVSTA<0 IF DRIVE IS UNSAFE

040534      000                 DRVSTA: .BYTE   0               ;DRIVE 0
040535      000                         .BYTE   0               ;DRIVE 1
040536      000                         .BYTE   0               ;DRIVE 2
040537      000                         .BYTE   0               ;DRIVE 3
040540      000                         .BYTE   0               ;DRIVE 4
040541      000                         .BYTE   0               ;DRIVE 5
040542      000                         .BYTE   0               ;DRIVE 6
040543      000                         .BYTE   0               ;DRIVE 7

                                ;TABLE OF DRIVE TYPES (DRVTYP=8 BYTES)
                                        ;DRVTYP= 0 IF DRIVE IS NONEXISTENT (DRVSTA=0, ALSO)
                                        ;DRVTYP= 4 IF DRIVE IS RP07.
                                        ;DRVTYP= 5 IF DRIVE IS RP07  MOVING HEAD OPTION
                                        ;DRVTYP=-1 IF NOT RP07

040544      000                 DRVTYP: .BYTE   0               ;DRIVE 0
040545      000                         .BYTE   0               ;DRIVE 1
040546      000                         .BYTE   0               ;DRIVE 2
040547      000                         .BYTE   0               ;DRIVE 3
040550      000                         .BYTE   0               ;DRIVE 4
040551      000                         .BYTE   0               ;DRIVE 5
040552      000                         .BYTE   0               ;DRIVE 6
040553      000                         .BYTE   0               ;DRIVE 7
```

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

040554  000             DPINT:  .BYTE   0               ;DRIVE 0
040555  000                     .BYTE   0               ;DRIVE 1
040556  000                     .BYTE   0               ;DRIVE 2
040557  000                     .BYTE   0               ;DRIVE 3
040560  000                     .BYTE   0               ;DRIVE 4
040561  000                     .BYTE   0               ;DRIVE 5
040562  000                     .BYTE   0               ;DRIVE 6
040563  000                     .BYTE   0               ;DRIVE 7

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

040564  000             DPRQS:  .BYTE   0               ;DRIVE 0
040565  000                     .BYTE   0               ;DRIVE 1
040566  000                     .BYTE   0               ;DRIVE 2
040567  000                     .BYTE   0               ;DRIVE 3
040570  000                     .BYTE   0               ;DRIVE 4
040571  000                     .BYTE   0               ;DRIVE 5
040572  000                     .BYTE   0               ;DRIVE 6
040573  000                     .BYTE   0               ;DRIVE 7

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

040574  000000          TRNSWT: .WORD   0

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

040576  000000          SRCHWT: .WORD   0

                                ;RPO7 DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
                                        ;ACTDRV=0 IF DRIVER IS INACTIVE
                                        ;ACTDRV>0 IF DRIVER IS ACTIVE

040600  000             ACTDRV: .BYTE   0

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

040601  000             ACTSTR: .BYTE   0
```

```
                                ;SAVE REGISTERS FLAG (SAVEFG=1 WORD)
                                        ;SAVEFG <0 IF SAVE THE RHXX/RP07 REGISTERS WHEN THE
                                        ;OPERATION IS COMPLETED AS PER (DPB=14).
                                        ;SAVEFG=0 IF SAVE THE RHXX/RP07 REGISTERS, AS PER
                                        ;(DPB=14), AFTER AN ERROR.

040602  000000                  SAVEFG: .WORD   0

                                ;SEEK FLAG (SEEKFG=1 WORD)
                                        ;SEEKFG=0 IF WHEN THE DISK ADDRESS ISN'T IN THE WINDOW
                                        ;FOR A DATA TRANSFER START A SEARCH COMMAND
                                        ;SEEKFG<0 IF DATA TRANSFER WILL DO IMPLIED SEEKS,
                                        ;DISREGARD THE WINDOW

040604  177777                  SEEKFG: .WORD   -1

                                ;TIMEOUT TABLE (TIMER=8 WORDS)
                                        ;THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION

040606  177777                  TIMER:  .WORD   -1       ;DRIVE 0
040610  177777                          .WORD   -1       ;DRIVE 1
040612  177777                          .WORD   -1       ;DRIVE 2
040614  177777                          .WORD   -1       ;DRIVE 3
040616  177777                          .WORD   -1       ;DRIVE 4
040620  177777                          .WORD   -1       ;DRIVE 5
040622  177777                          .WORD   -1       ;DRIVE 6
040624  177777                          .WORD   -1       ;DRIVE 7

                                ;DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)
                                        ;DTUW<0 IF NO DATA TRANSFER UNDERWAY
                                        ;DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE N

040626  177777                  DTUW:   .WORD   -1

                                ;ATTENTION BITS TABLE (ATABIT=8 BYTES)
                                        ;THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES
                                        ;ATTENTION BIT

040630      001                 ATABIT: .BYTE   1        ;DRIVE 0
040631      002                         .BYTE   2        ;DRIVE 1
040632      004                         .BYTE   4        ;DRIVE 2
040633      010                         .BYTE   10       ;DRIVE 3
040634      020                         .BYTE   20       ;DRIVE 4
040635      040                         .BYTE   40       ;DRIVE 5
040636      100                         .BYTE   100      ;DRIVE 6
040637      200                         .BYTE   200      ;DRIVE 7
```

```
                                ;STORAGE FOR RPADR (THE FIRST ADDRESS (776700) OF THE RHXX/RP07),
                                ;RPVEC (THE VECTOR ADDRESS (254)), AND RPVEC+2 (THE BR LEVEL (5)).
                                
040640  176700                  RPADR:  .WORD   176700          ;RHXX/RP07 CONTROL STATUS REGISTER
040642  000254  000240          RPVEC:  .WORD   254.+32.        ;VECTOR ADDRESS & BR LEVEL 5
040646  000050                  RHEXT:  .WORD   50              ;OFFSET TO RPBAE
                                
                                ;SEARCH DIFFERENCE IS 1 SECTOR
                                
040650  000001                  MXWNDW: .WORD   1
                                
                                ;DEFINITIONS OF THE RHXX/RP07 ADDRESS INDEXES
                                
        000000                  RPCS1   = 0                     ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00)
        000002                  RPWC    = 2                     ;WORD COUNT REGISTER (NOT A DRIVE REG)
        000004                  RPBA    = 4                     ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
        000006                  RPDA    = 6                     ;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 05)
        000010                  RPCS2   = 10                    ;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
        000012                  RPDS    = 12                    ;DRIVE STATUS REGISTER (DRIVE REG 01)
        000014                  RPER1   = 14                    ;ERROR REGISTER #1 (DRIVE REG. 02)
        000016                  RPAS    = 16                    ;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 04)
        000020                  RPLA    = 20                    ;LOOK AHEAD REGISTER (DRIVE REG. 07)
        000022                  RPDB    = 22                    ;DATA BUFFER REGISTER (NOT A DRIVE REG.)
        000024                  RPMR1   = 24                    ;MAINTAINABILITY REGISTER (DRIVE REG. 03)
        000026                  RPDT    = 26                    ;DRIVE TYPE REGISTER (DRIVE REG. 06)
        000030                  RPSN    = 30                    ;SERIAL NUMBER REGISTER (DRIVE REG. 10)
        000032                  RPOF    = 32                    ;OFFSET REGISTER (DRIVE REG. 11)
        000034                  RPDC    = 34                    ;DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
        000036                  RPCC    = 36                    ;DUMMY  ADDRESS REGISTER (DRIVE REG. 13)
        000040                  RPER2   = 40                    ;ERROR REGISTER #2
        000042                  RPER3   = 42                    ;ERROR REGISTER #2 (DRIVE REG. 15)
        000044                  RPEC1   = 44                    ;ECC POSITION REGISTER (DRIVE REG. 16)
        000046                  RPEC2   = 46                    ;ECC PATTERN REGISTER (DRIVE REG.  17)
        000050                  RPBAE   = 50                    ;BUS ADDRESS EXTENTION REGISTER
        000052                  RPCS3   = 52                    ;CONTROL AND STATUS REGISTER #3
```

```
                                ;RHXX/RP07 DRIVER INITIALIZATION CODE
                                        ;THIS ROUTINE WILL DETERMINE WHICH RP07 DRIVES ARE
                                        ;AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
                                        ;TO THE PROPER STATE FOR EACH DRIVE.
                                        ;NOTE: THIS ROUTINE CALLS DRVINT
                                ;
                                ;CALL
                                ;
                                ;       JSR     PC,RPINIT
                                ;       RETURN
                                ;
                                ;NOTE: THE 'P' OR 'L' CLOCK MUST BE STARTED
                                ;
040652  104412                  RPINIT: SAVREG                  ;SAVE R0 - R5
040654  013746  177776                  MOV     @#PS,-(SP)      ;SAVE THE PRESENT PROCESSOR STATUS
040660  012737  000240  177776          MOV     #<5*32.>,@#PS   ;CHANGE THE PRIORITY TO 5
040666  004737  046052                  JSR     PC,CLRQUE       ;CLEAR ALL REQUEST QUEUES
040672  012701  040424                  MOV     #RPSTUO,R1      ;FIRST ADDRESS TO BE CLEARED
040676  012702  040604                  MOV     #SEEKFG,R2      ;LAST ADDRESS TO BE CLEARED
040702  005021                  1$:     CLR     (R1)+           ;CLEAR
040704  020102                          CMP     R1,R2           ;ARE WE DONE?
040706  103775                          BLO     1$              ;BRANCH IF NO
                                
040710  012702  040626                  MOV     #DTUW,R2        ;LAST ADDRESS
040714  012721  177777          2$:     MOV     #-1,(R1)+       ;INITIALIZE
040720  020102                          CMP     R1,R2           ;DONE?
040722  101774                          BLOS    2$              ;LOOP IF NO
                                
040724  005037  040534                  CLR     DRVSTA          ;SET ALL DRIVES TO OFFLINE
040730  005037  040536                  CLR     DRVSTA+2
040734  005037  040540                  CLR     DRVSTA+4
040740  005037  040542                  CLR     DRVSTA+6
040744  013703  040642                  MOV     RPVEC,R3        ;SETUP THE RHXX/RP07 VECTOR
040750  012723  043254                  MOV     #ISR,(R3)+
040754  013713  040644                  MOV     RPVEC+2,(R3)
040760  013704  040640                  MOV     RPADR,R4        ;FIRST ADDRESS OF RHXX/RP07
040764  012764  000040  000010          MOV     #BIT05,RPCS2(R4)  ;MASSBUS INIT
040772  005001                          CLR     R1              ;START WITH DRIVE 0
040774  004037  041030          3$:     JSR     R0,DRVINT       ;INIT THE DRIVE
041000  000401                          BR      4$              ;'DVA' NOT SET OR PARITY ERROR
041002  000402                          BR      5$              ;NORMAL RETURN
                                
041004  105061  040534          4$:     CLRB    DRVSTA(R1)      ;SET DRIVE STATUS TO OFFLINE
041010  005201                  5$:     INC     R1              ;GO TO NEXT DRIVE
041012  042701  177770                  BIC     #^C7,R1         ;MASK OUT UNUSED BITS
041016  001366                          BNE     3$              ;BR IF MORE DRIVES TO GO
041020  012637  177776                  MOV     (SP)+,@#PS      ;RESTORE THE PROCESSOR STATUS
041024  104413                          RESREG                  ;RESTORE R0 - R5
041026  000207                          RTS     PC              ;BYE-BYE
```

```
                                        ;DRIVE INITILIZATION ROUTINE
                                        ;       ;THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
                                        ;       ;AN RP07. IF IT IS, A "READ IN PRESET" IS ISSUED AND FMT22
                                        ;       ;IS SET TO A "1". THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
                                        ;       ;INSURE THEY ARE ALL ON A "1". AND DEPENDING ON THEIR STATE,
                                        ;       ;DRVSTA IS SET TO THE PROPER CONDITION.
                                        ;CALL
                                        ;       MOV     RPADR,R4        ;UNIBUS ADDRESS OF RHXX/RP07 (RPCS1)
                                        ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER
                                        ;       JSR     R0,DRVINT       ;CALLED BY A JSR
                                        ;       RETURN1                 ;ERROR OCCURRED (PARITY)
                                        ;       RETURN2                 ;NORPAL RETURN
                                        ;
041030  010546                  DRVINT: MOV     R5,-(SP)        ;SAVE R5
041032  112761  177777  040554          MOVB    #-1,DPINT(R1)   ;SET THE DUAL PORT INITIALIZE FLAG
041040  006301                          ASL     R1
041042  012761  023420  040606          MOV     #10000.,TIMER(R1)       ;START 10. SECOND TIMER
041050  006201                          ASR     R1              ;DRIVE ADDRESS

041052  105061  040534          1$:     CLRB    DRVSTA(R1)      ;START DRIVE STATUS AS OFFLINE
041056  105061  040544                  CLRB    DRVTYP(R1)      ;CLEAR THE DRIVE TYPE INDICATOR
041062  010164  000010                  MOV     R1,RPCS2(R4)    ;SELECT A DRIVE
041066  112764  000111  000000          MOVB    #111,RPCS1(R4)  ;DO A DRIVE CLEAR COMMAND (& SEIZE DRIVE)
041074  032764  010000  000010          BIT     #BIT12,RPCS2(R4) ;NONEXISTENT DRIVE?
041102  001403                          BEQ     2$              ;NO---BRANCH
041104  004737  045506                  JSR     PC,SET.IE       ;GO SET "IE" WITHOUT A "TRE"
041110  000513                          BR      6$              ;LEAVE THIS ROUTINE

041112  105061  040534          2$:     CLRB    DRVSTA(R1)      ;SET DRIVE STATUS TO OFFLINE
041116  032764  004000  000000          BIT     #BIT11,RPCS1(R4)  ;SEE IF DRIVE AVAILABLE
041124  001004                          BNE     3$              ;BRANCH IF DVA SET
041126  105761  040554                  TSTB    DPINT(R1)       ;SOFTWARE TIMEOUT ON DUAL PORT INITIALIZE ?
041132  001347                          BNE     1$              ;BRANCH IF NOT
041134  000501                          BR      6$              ;OTHERWISE EXIT

041136  004037  045116          3$:     JSR     R0,RD.RP        ;READ THE DRIVE TYPE REG.
041142  000026                          RPDT
041144  041342                          8$                      ;ERROR RETURN ADDRESS
041146  012605                          MOV     (SP)+,R5        ;PUT DRIVE TYPE IN R5
041150  112761  000005  040544          MOVB    #5,DRVTYP(R1)   ;SET RP07 INDICATOR
041156  022705  020040                  CMP     #20040,R5       ;SINGLE PORT RP07
041162  001420                          BEQ     4$              ;BR IF YES
041164  022705  024040                  CMP     #24040,R5       ;DUAL PORT RP07
041170  001415                          BEQ     4$              ;BR IF YES
041172  112761  000004  040544          MOVB    #4,DRVTYP(R1)   ;SET RP07+ INDICATOR
041200  022705  020042                  CMP     #20042,R5       ;SINGLE PORT RP07+
041204  001407                          BEQ     4$              ;BRANCH IF SO
041206  022705  024042                  CMP     #24042,R5       ;DUAL PORT RP07+
041212  001404                          BEQ     4$              ;BRANCH IF SO
041214  112761  177777  040544          MOVB    #-1,DRVTYP(R1)  ;SET INDICATOR TO 'OTHER'
041222  000446                          BR      6$              ;EXIT

041224  012746  000121          4$:     MOV     #121,-(SP)      ;DO A "READ-IN PRESET"
041230  004037  045210                  JSR     R0,WRT.RP
041234  000000                          RPCS1
041236  041342                          8$
```

RPO7 DRIVER

```
     58 041240  012746  010000                    MOV     #BIT12,(SP)     ;SET FMT16=1
     59 041244  004037  045210                    JSR     R0,WRT.RP
     60 041250  000032                            RPOF
     61 041252  041342                            8$
     62 041254  004037  045116                    JSR     R0,RD.RP        ;READ RPDS
     63 041260  000012                            RPDS
     64 041262  041342                            8$
     65 041264  012605                            MOV     (SP)+,R5        ;AND SAVE IT IN R5
     66 041266  100015                            BPL     5$              ;BRANCH IF ATA=0
     67 041270  116164  040630  000016            MOVB    ATABIT(R1),RPAS(R4) ;CLEAR ATTENTION BIT
     68 041276  004037  045116                    JSR     R0,RD.RP        ;FIND OUT WHY ATA=1
     69 041302  000014                            RPER1
     70 041304  041342                            8$
     71 041306  006126                            ROL     (SP)+           ;IS IT UNSAFE?
     72 041310  100004                            BPL     5$              ;BR IF NOT
     73 041312  112761  177777  040534            MOVB    #-1,DRVSTA(R1)  ;SET UNSAFE INDICATOR
     74 041320  000407                            BR      6$              ;EXIT
     75
     76 041322  005105                    5$:     COM     R5              ;CHECK MOL, DPR, DRY, AND VV
     77 041324  042705  167077                    BIC     #^C<BIT12!BIT08!BIT07!BIT06>,R5
     78 041330  001003                            BNE     6$              ;BRANCH IF MOL, DPR, DRY, OR VV IS CLEAR
     79 041332  112761  000001  040534            MOVB    #1,DRVSTA(R1)   ;SET DRIVE STATUS TO ONLINE
     80 041340  005720                    6$:     TST     (R0)+           ;STEP OVER THE ERROR RETURN
     81 041342                            7$:
     82 041342  006301                    8$:     ASL     R1              ;WORD INDEX
     83 041344  012761  177777  040606            MOV     #-1,TIMER(R1)   ;STOP THE CLOCK
     84 041352  006201                            ASR     R1              ;DRIVE ADDRESS
     85 041354  105061  040554                    CLRB    DPINT(R1)
     86 041360  012605                            MOV     (SP)+,R5        ;RESTORE R5
     87 041362  000200                            RTS     R0              ;EXIT
```

```
                                        ;REQUEST PRE PROCESSOR HANDLES SUBSYSTEM REQUEST
                                        ;
                                        ;CALL
                                        ;
                                        ;       JSR     R0,RP07         ;CALL THE RP07 DRIVER
                                        ;       PNTADR                  ;ADDRESS OF POINTER OF DRIVES PARAMETER BLOCK
                                        ;       RETURN1                 ;RETURN HERE IF QUEUE IS FULL
                                        ;       RETURN2                 ;RETURN HERE IF REQUEST IS IN QUEUE OR THERE
                                                                        ;IS AN ERROR CONDITION

041364  013746  177776          RP07:   MOV     @#PS,-(SP)      ;SAVE THE CALLING STATUS
041370  013737  040644  177776          MOV     RPVEC+2,@#PS    ;DON'T ALLOW ANY RP07 INTERRUPTS
041376  112737  000001  040600          MOVB    #1,ACTDRV       ;SET "ACTIVE DRIVER" FLAG
041404  104412                          SAVREG                  ;SAVE R0 - R5
041406  011002                          MOV     (R0),R2         ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
041410  005062  000016                  CLR     16(R2)          ;CLEAR THE STATUS/ERROR INDICATOR
041414  111201                          MOVB    (R2),R1         ;PICKUP THE DRIVE NUMBER
041416  013704  040640                  MOV     RPADR,R4        ;UNIBUS ADDRESS OF RPCS1
041422  105761  040534                  TSTB    DRVSTA(R1)      ;CHECK DRIVES STATUS
041426  003006                          BGT     1$              ;BRANCH IF ONLINE
041430  004037  041030                  JSR     R0,DRVINT       ;GO INIT. THE DRIVE
041434  000421                          BR      3$              ;ERROR RETURN

041436  105761  040534                  TSTB    DRVSTA(R1)      ;IS DRIVE STATUS ONLINE?
041442  003435                          BLE     5$              ;BR IF NOT
041444  105761  040564          1$:     TSTB    DPRQS(R1)       ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
041450  001016                          BNE     4$              ;BR IF YES
041452  010164  000010                  MOV     R1,RPCS2(R4)    ;SELECT THE DRIVE
041456  004037  046150                  JSR     R0,DRVQUE       ;PUT THIS REQUEST IN QUEUE
041462  000450                          BR      8$              ;QUEUE IS FULL

041464  105761  040524          2$:     TSTB    DRVACT(R1)      ;IS THIS DRIVE ACTIVE?
041470  001042                          BNE     7$              ;BR IF YES
041472  004737  041622                  JSR     PC,OPT          ;CALL THE OPTIMIZER
041476  000437                          BR      7$

041500  004737  042750          3$:     JSR     PC,CI7          ;GO HANDLE THE PARITY ERROR
041504  000434                          BR      7$

041506  004037  046150          4$:     JSR     R0,DRVQUE       ;PUT REQUEST IN QUEUE
041512  000434                          BR      8$              ;QUEUE IS FULL

041514  012764  000000  000036          MOV     #0,RPCC(R4)     ;WRITE THE CURRENT CYL REG
041522  032714  000100                  BIT     #BIT06,(R4)     ;IE BIT SET ?
041526  001023                          BNE     7$              ;YES
041530  004737  045506                  JSR     PC,SET.IE       ;SET THE INTERRUPT
041534  000420                          BR      7$              ;RETURN

041536  105761  040534          5$:     TSTB    DRVSTA(R1)      ;SEE IF DRIVE OFFLINE OR UNSAFE
041542  002412                          BLT     6$              ;BR IF UNSAFE
041544  012762  140000  000016          MOV     #BIT15!BIT14,16(R2)  ;SET OFFLINE ERROR INDICATOR
041552  105761  040544                  TSTB    DRVTYP(R1)      ;SEE IF OFFLINE OR NONEXISTENT
041556  001007                          BNE     7$              ;BR IF OFFLINE
041560  012762  100002  000016          MOV     #BIT15!BIT01,16(R2)  ;REPORT DRIVE NONEXISTENT
041566  000403                          BR      7$              ;GO TO EXIT

041570  012762  110000  000016  6$:     MOV     #BIT15!BIT12,16(R2)  ;DRIVE IS UNSAFE
```

```
     61 041576  104413                 7$:     RESREG                  ;RESTORE R0   R5
     62 041600  005720                         TST     (R0)+           ;SETUP FOR NORMAL RETURN
     63 041602  000401                         BR      9$              ;FINISH UP, THEN EXIT
     64
     65 041604  104413                 8$:     RESREG                  ;RESTORE R0 - R5
     66 041606  005720                 9$:     TST     (R0)+           ;CORRECT THE RETURN ADDRESS
     67 041610  105037  040600                 CLRB    ACTDRV          ;CLEAR "ACTIVE DRIVER" FLAG
     68 041614  012637  177776                 MOV     (SP)+,@#PS      ;RETURN "PS" TO USER LEVEL
     69 041620  000200                         RTS     R0              ;RETURN TO CALLER
```

```
                                        ;OPTIMIZER CALLED FOR A PARTICULAR DRIVE
                                        ;
                                        ;CALL
                                        ;       MOV     #DPB,R2         ;ADDRESS OF DRIVE PARAMETER BLOCK
                                        ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER
                                        ;       JSR     PC,OPT          ;SETUP A COMMAND
                                        ;
041622  104412                  OPT:    SAVREG                          ;SAVE R0 - R5
041624  013746  177776                  MOV     @#PS, (SP)              ;SAVE PROC. STATUS
041630  146137  040630  040576          BICB    ATABIT(R1),SRCHWT       ;CLEAR SEARCH FLAG
041636  105061  040564                  CLRB    DPRQS(R1)               ;RESET THE PORT REQ FLAG
041642  004737  046224                  JSR     PC,GETREQ               ;GET "DPB" POINTER OF REQUEST
041646  005702                          TST     R2                      ;IS THERE A REQUEST IN QUEUE?
041650  001475                          BEQ     8$                      ;NO--BRANCH TO EXIT
041652  010164  000010                  MOV     R1,RPCS2(R4)            ;LOAD THE DRIVE ADDRESS
041656  012764  000111  000000          MOV     #111,RPCS1(R4)          ;CLEAR THE DRIVE
041664  032764  004000  000000          BIT     #BIT11,RPCS1(R4)        ;DVA SET ?
041672  001446                          BEQ     6$                      ;TO PROT REQUEST ,IF NOT
041674  105761  040534          1$:     TSTB    DRVSTA(R1)              ;IS DRIVE ONLINE?
041700  003014                          BGT     2$                      ;YES--BRANCH
041702  004737  046246                  JSR     PC,POPQUE               ;NO--REMOVE REQUEST FROM QUEUE
041706  012762  140000  000016          MOV     #BIT15!BIT14,16(R2)  ;SET OFFLINE STATUS/ERROR INDICATOR
041714  105761  040534                  TSTB    DRVSTA(R1)              ;IS DRIVE UNSAFE ?
041720  100056                          BPL     9$                      ;BR TO EXIT IF NOT
041722  012762  110000  000016          MOV     #BIT15!BIT12,16(R2)  ;SET UNSAFE STATUS/ERROR INDICATOR
041730  000452                          BR      9$                      ;BRANCH TO EXIT

041732  122762  000150  000002  2$:     CMPB    #150,2(R2)              ;IS THE REQUEST FOR I/O?
041740  002407                          BLT     3$                      ;YES- BRANCH
041742  122762  000135  000002          CMPB    #135,2(R2)              ;IS IT A DIAGNOSTIC COMMAND ?
041750  001403                          BEQ     3$                      ;BRANCH IF SO
041752  004737  042372                  JSR     PC,CI4                  ;CALL THE COMMAND INITIATOR
041756  000437                          BR      9$                      ;BRANCH TO EXIT

041760  005737  040626          3$:     TST     DTUW                    ;DATA TRANSFER UNDERWAY?
041764  002006                          BGE     5$                      ;BR IF YES
041766  005737  040604                  TST     SEEKFG                  ;DO IMPLIED SEEKS ?
041772  100003                          BPL     5$                      ;NO, DO SEARCH
041774  004737  042066          4$:     JSR     PC,CI1                  ;START A DATA TRANSFER
042000  000426                          BR      9$

042002  004737  042260          5$:     JSR     PC,CI3                  ;START A SEARCH ON TARGET SECTOR -1
042006  000423                          BR      9$                      ;GO TO THE EXIT

042010  112761  177777  040564  6$:     MOVB    #-1,DPRQS(R1)           ;SET PORT REQUEST INDICATOR
042016  010103                          MOV     R1,R3                   ;SET UP TO ADDRESS WORDS
042020  006303                          ASL     R3                      ;CONVERT TO WORD INDEX
042022  012763  035230  040606          MOV     #15000.,TIMER(R3)    ;START 15. SECOND TIMER
042030  012764  000000  000036          MOV     #0,RPCC(R4)             ;SET PORT REQUEST
042036  000402                          BR      8$                      ;EXIT

042040  004737  042750          7$:     JSR     PC,CI7                  ;PROCESS THE PARITY ERROR
042044  032714  000100          8$:     BIT     #BIT06,(R4)             ;SEE IF 'IE' ALREADY SET
042050  001002                          BNE     9$                      ;BR IF SET
042052  004737  045506                  JSR     PC,SET.IE               ;SET "IE" WITHOUT A "TRE"
042056  012637  177776          9$:     MOV     (SP)+,@#PS              ;RESTORE PROC. STATUS
042062  104413                          RESREG                          ;RESTORE R0 - R5
```

```
CZRJOBO RP07 PERF EXER MACRO V04.00 1 DEC 83 10:32:28 PAGE 105 1                SEQ 0205
RP07 DRIVER

     58 042064  00020*                          HTS     PC
```

M16


```
                                        ;COMMAND INITIATOR
                                        ;
                                        ;CALL
                                        ;       MOV     #DPB,R2         ;ADDRESS OF DRIVE PARAMETER BLOCK
                                        ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER
                                        ;       JSR     PC,CI????       ;CI???? = CI1, CI3, OR CI4
                                        ;                               ;WHERE:
                                        ;                               ; CI1 = DATA TRANSFER
                                        ;                               ; CI3 = SEARCH REQUESTED BY DATA XFER
                                        ;                               ; CI4 = NO DATA TRANSFER

042066  004737  046246          CI1:    JSR     PC,POPQUE       ;REMOVE REQUEST FROM "DRIVES WAIT" QUEUE
042072  010237  040574                  MOV     R2,TRNSWT       ;PUT REQ. IN TRANSFER WAIT QUEUE
042076  010203                          MOV     R2,R3           ;DPB ADDRESS TO R3
042100  013704  040640                  MOV     RPADR,R4        ;RPCS1 ADDRESS
042104  010164  000010                  MOV     R1,RPCS2(R4)    ;SELECT DRIVE
042110  122762  000135  000002          CMPB    #135,2(R2)      ;IS IT A DIAGNOSTIC COMMAND ?
042116  001011                          BNE     1$              ;BRANCH IF NOT
042120  016246  000004                  MOV     4(R2),-(SP)     ;GET THE ROUTINE NUMBER, PARAMETERS
042124  052716  100000                  BIS     #BIT15,(SP)     ;SET THE DIAGNOSTIC MODE BIT
042130  004037  045210                  JSR     R0,WRT.RP       ;WRITE THE RPMR1 REG
042134  000024                          RPMR1
042136  042750                          CI7
042140  000422                          BR      2$              ;LOAD THE COMMAND AND EXIT

042142  062703  000004          1$:     ADD     #4,R3           ;DESIRED WORD COUNT
042146  062704  000002                  ADD     #2,R4           ;RPWC ADDRESS
042152  012324                          MOV     (R3)+,(R4)+     ;LOAD WORD COUNT
042154  012324                          MOV     (R3)+,(R4)+     ;LOAD BUFFER ADDRESS
042156  012346                          MOV     (R3)+,-(SP)     ;LOAD SECTOR AND TRACK
042160  004037  045210                  JSR     R0,WRT.RP       ;CALL THE LOAD(WRITE) ROUTINE
042164  000006                          RPDA                    ;INDEX OF REGISTER TO LOAD
042166  042750                          CI7                     ;ERROR RETURN ADDRESS
042170  012346                          MOV     (R3)+,-(SP)     ;LOAD CYLINDER ADDRESS
042172  004037  045210                  JSR     R0,WRT.RP
042176  000034                          RPDC
042200  042750                          CI7
042202  004737  042232                  JSR     PC,CI2          ;SEE IF BIT15 SHOULD BE SET IN RPMR1

042206  016246  000002          2$:     MOV     2(R2),-(SP)     ;LOAD 'COMMAND+GO", "A17&A16", AND "PSEL"
042212  004037  045210                  JSR     R0,WRT.RP
042216  000000                          RPCS1
042220  042750                          CI7
042222  010137  040626                  MOV     R1,DTUW         ;SET "DATA TRANSFER UNDERWAY"
042226  000137  042712                  JMP     CI5

042232  026262  000012  000156  CI2:    CMP     12(R2),FE1(R2)  ;DATA XFER TO FE CYLINDERS ?
042240  002406                          BLT     1$              ;BR IF NO
042242  012746  100000                  MOV     #BIT15,-(SP)    ;SET THE DIAGNOSTIC MODE BIT
042246  004037  045210                  JSR     R0,WRT.RP       ;WRITE THE RPMR1 REG
042252  000024                          RPMR1
042254  042750                          CI7
042256  000207                  1$:     RTS     PC

042260  013704  040640          CI3:    MOV     RPADR,R4        ;RPCS1 ADDRESS
042264  010164  000010                  MOV     R1,RPCS2(R4)    ;SELECT DRIVE
042270  016246  000012                  MOV     12(R2),-(SP)    ;DESIRED CYLINDER ADDRESS
```

```
 64 042274  004037  04521C                JSR     RO,WR?.RP
 65 042300  000034                        RPDC
 66 042302  042750                        CI?
 67 042304  004737  042232                JSR     PC,CI2          ;SEE IF BIT15 SHOULD BE SET IN RPM 1
 68
 69 042310  116203  000010                MOVB    10(R2),R3       ;PICKUP SECTOR ADDRESS
 70 042314  163703  04065C                SUB     MXWNDW,R3       ;BACKUP BY MAX. SEARCH FOR I/O WINDOW
 71 042320  002002                        JGE     1$              ;BR IF >= 0
 72 042322  062703  000062                ADD     #50.,R3         ;ADD MAXIMUM SECTOR COUNT BACK
 73 042326  010346                1$:     MOV     R3,-(SP)        ;COMBINE THE ADJUSTED SECTOR WITH
 74 042330  116266  000011  000001        MOVB    11(R2),1(SP)    ;THE DESIRED TRACK
 75 042336  004037  045210                JSR     RO,WRT.RP       ;LOAD DESIRED TRACK & SECTOR
 76 042342  000006                        RPDA
 77 042344  042750                        CI7
 78
 79 042346  012746  000131                MOV     #131, (SP)      ;START A SEARCH
 80 042352  004037  045210                JSR     RO,WRT.RP
 81 042356  000000                        RPCS1
 82 042360  042750                        CI7
 83 042362  156137  040630  040576        BISB    ATABIT(R1),SRCH+1T      ;SET "SEARCH WAIT" KEY
 84 042370  000550                        BR      CI5
 85
 86 042372  013704  040640        CI4:    MOV     RPADR,R4        ;RPCS1 ADDRESS
 87 042376  010164  000010                MOV     R1,RPCS2(R4)    ;SELECT DRIVE
 88 042402  116203  000002                MOVB    2(R2),R3        ;PICKUP THE REQUESTED COMMAND
 89 042406  122703  000131                CMPB    #131,R3         ;IS IT A SEARCH COMMAND?
 90 042412  001007                        BNE     1$              ;BRANCH IF NO
 91 042414  016246  000010                MOV     10(R2),-(SP)    ;LOAD DESIRED TRACK & SECTOR
 92 042420  004037  045210                JSR     RO,WRT.RP
 93 042424  000006                        RPDA
 94 042426  042750                        CI7
 95 042430  000403                        BR      2$              ;GO LOAD CYLINDER
 96
 97 042432  122703  000105        1$:     CMPB    #105,R3         ;IS IT A SEEK COMMAND
 98 042436  001011                        BNE     3$              ;BRANCH IF NO
 99 042440  016246  000012        2$:     MOV     12(R2),-(SP)    ;LOAD DESIRED CYLINDER
100 042444  004037  045210                JSR     RO,WRT.RP
101 042450  000034                        RPDC
102 042452  042750                        CI7
103 042454  004737  042232                JSR     PC,CI2          ;SEE IF BIT15 SHOULD BE SET IN RPMR1
104 042460  000525                        BR      CI6
105
106 042462  122703  000115        3$:     CMPB    #115,R3         ;IS IT AN "OFFSET" COMMAND ?
107 042466  001013                        BNE     4$              ;BR IF NO
108 042470  004037  045116                JSR     RO,RD.RP        ;MERGE THE OFFSET VALUE INTO RPOF
109 042474  000032                        RPOF                    ;BUT DON'T CHANGE THE UPPER
110 042476  042750                        CI7
111 042500  116216  000001                MOVB    1(R2),(SP)      ;BYTE WHEN LOADING THE
112 042504  004037  045210                JSR     RO,WRT.RP       ;REGISTER (RPOF)
113 042510  000032                        RPOF
114 042512  042750                        CI7
115 042514  000507                        BR      CI6             ;GO START THE COMMAND
116
117 042516  122703  000107        4$:     CMPB    #107,R3         ;IS IT A "RECALIBRATE" COMMAND?
118 042522  001504                        BEQ     CI6             ;BRANCH IF YES
119 042524  122703  000117                CMPB    #117,R3         ;IS IT A RETURN TO CENTER?
120 042530  001501                        BEQ     CI6             ;BRANCH IF YES
```

```
121 042532  122703  000143          5$:     CMPB    #143,R3         ;IS IT A "SET FORMAT' COMMAND?
122 042536  001014                          BNE     6$              ;BRANCH IF NO
123 042540  004037  045116                  JSR     R0,RD.RP        ;READ THE OFFSET REGISTER
124 042544  000032                          RPOF
125 042546  042750                          CI7
126 042550  116266  000001  000001          MOVB    1(R2),1(SP)     ;COMBINE "FMT16","ECI", AND  HCI
127 042556  004037  045210                  JSR     R0,WRT.RP
128 042562  000032                          RPOF
129 042564  042750                          CI7
130 042566  000436                          BR      12$
131
132 042570  122703  000141          6$:     CMPB    #141,R3         ;IS IT A "GET REGISTER' COMMAND?
133 042574  001023                          BNE     10$             ;BRANCH IF NO
134 042576  016203  000006          7$:     MOV     6(R2),R3        ;POINTS TO 1ST ADDRESS OF WHERE
135                                                                 ;TO PUT THE REGISTER(S)
136 042602  116237  000010  042620          MOVB    10(R2),9$       ;INIT. THE INDEX FOR THE FIRST REG.
137 042610  116205  000011                  MOVB    11(R2),R5       ;INDEX OF LAST REG. TO MOVE
138 042614  004037  045116          8$:     JSR     R0,RD.RP        ;READ RHXX/RP07 REGISTER
139 042620  000000          9$:     RPCS1                           ;INDEX OF REG. TO READ
140 042622  042750                          CI7
141 042624  012623                          MOV     (SP)+,(R3)+     ;GET THE CONTENTS OF RHXX//RP07 REG.
142 042626  023705  042620                  CMP     9$,R5           ;LAST REG. BEEN READ?
143 042632  001414                          BEQ     12$             ;GET OUT IF YES
144 042634  062737  000002  042620          ADD     #2,9$           ;INCREASE THE INDEX BY 2
145 042642  000764                          BR      8$              ;LOOP--MORE TO READ
146
147 042644  122703  000145          10$:    CMPB    #145,R3         ;IS IT A "SELECT DRIVE' COMMAND?
148 042650  001405                          BEQ     12$             ;BRANCH IF YES
149 042652  010346                  11$:    MOV     R3,(SP)         ;LOAD THE COMMAND
150 042654  004037  045210                  JSR     R0,WRT.RP
151 042660  000000                          RPCS1
152 042662  042750                          CI7
153 042664  004737  046246          12$:    JSR     PC,POPQUE       ;REMOVE REQ. FROM QUEUE
154 042670  052762  000200  000016          BIS     #BIT07,16(R2)   ;SET THE "DONE" BIT
155 042676  005737  040602                  TST     SAVEFG          ;SAVE RHXX/RP07 REGISTERS ?
156 042702  100002                          BPL     13$             ;BR IF NO
157 042704  004737  045342                  JSR     PC,SVRHXX       ;YES--GO SAVE THE REGISTERS
158 042710  000207                  13$:    RTS     PC              ;RETURN TO USER
159
160 042712  006301                  CI5:    ASL     R1
161 042714  012761  023420  040606          MOV     #10000.,TIMER(R1)       ;START 10. SECOND TIMER
162 042722  006201                          ASR     R1
163 042724  112761  000001  040524          MOVB    #1,DRVACT(R1)   ;SET THE DRIVE ACTIVE
164 042732  000207                          RTS     PC              ;RETURN TO THE USER
165
166 042734  010346                  CI6:    MOV     R3,-(SP)        ;LOAD THE COMMAND
167 042736  004037  045210                  JSR     R0,WRT.RP
168 042742  000000                          RPCS1
169 042744  042750                          CI7
170 042746  000761                          BR      CI5
171
172 042750  005702                  CI7:    TST     R2              ;ANYTHING IN QUEUE ?
173 042752  001001                          BNE     1$              ;BRANCH IF QUEUE IS THERE
174 042754  000207                          RTS     PC              ;OTHERWISE EXIT
175 042756  012762  104000  000016  1$:     MOV     #BIT15!BIT11,16(R2)     ;SET "PARITY" ERROR INDICATOR
176
177 042764  012746  000111          CI7B:   MOV     #111,-(SP)      ;DO A "DRIVE CLEAR"
```

```
178 042770  004037  045210                  JSR     R0,WRT.RP
179 042774  000000                          RPCS1
180 042776  043036                          CI8
181 043000  004737  046130                  JSR     PC,EMPTYQ       ;EMPTY THE QUEUE
182 043004  105061  040564                  CLRB    DPRQS(R1)       ;CLEAR THE PORT REQUEST FLAG
183 043010  105061  040524                  CLRB    DRVACT(R1)      ;DRIVE IS IDLE
184 043014  020237  040574                  CMP     R2,TRNSWT       ;IF THIS DRIVE HAD AN I/O REQUEST
185 043020  001005                          BNE     1$              ;IN PROGRESS CLEAR ALL OF THE FLAGS
186 043022  005037  040574                  CLR     TRNSWT
187 043026  012737  177777  040626          MOV     #-1,DTUW
188 043034  000207                  1$:     RTS     PC
189
190 043036  104412                  CI8:    SAVREG                  ;SAVE R0 - R5
191 043040  005001                          CLR     R1
192 043042  005003                          CLR     R3
193 043044  105761  040524          1$:     TSTB    DRVACT(R1)      ;DRIVE ACTIVE?
194 043050  001003                          BNE     2$              ;BRANCH IF IN ACTIVE
195 043052  105761  040564                  TSTB    DPRQS(R1)       ;PORT REQUEST
196 043056  001443                          BEQ     6$              ;BRANCH IF NOT
197 043060  013702  040574          2$:     MOV     TRNSWT,R2       ;GET "DPB" FROM THE "TRANSFER WAIT" QUEUE
198 043064  020137  040626                  CMP     R1,DTUW         ;DID THIS DRIVE HAVE AN I/O IN PROGRESS?
199 043070  001402                          BEQ     3$              ;BRANCH IF YES
200 043072  004737  046224                  JSR     PC,GETREQ       ;GET THE DPB POINTER
201 043076  005702                  3$:     TST     R2              ;QUEUE ENTRY FOR DRIVE ?
202 043100  001413                          BEQ     5$              ;BR IF NOT
203 043102  032764  010000  000010          BIT     #BIT12,RPCS2(R4)  ;'NED' SET ?
204 043110  001404                          BEQ     4$              ;BR IF NOT
205 043112  012762  100002  000016          MOV     #BIT15!BIT01,16(R2)  ;SET 'DRIVE NON-EXISTENT' INDICATOR
206 043120  000403                          BR      5$              ;CONTINUE
207
208 043122  012762  102000  000016  4$:     MOV     #BIT15!BIT10,16(R2) ;SET "NON-CLEARABLE PARITY" ERROR INDICATOR
209 043130  012763  177777  040606  5$:     MOV     #-1,TIMER(R3)   ;STOP THE TIMER
210
214 043136  105061  040524                  CLRB    DRVACT(R1)      ;SET "DRIVE ACTIVE" TO IDLE
215 043142  105061  040564                  CLRB    DPRQS(R1)       ;CLEAR PORT REQUEST FLAG
216 043146  020137  040626                  CMP     R1,DTUW         ;IS THIS DRIVE SETUP FOR A TRANSFER
217 043152  001005                          BNE     6$              ;BR IF NOT
218 043154  012737  177777  040626          MOV     #-1,DTUW        ;RESET THE INDICATOR
219 043162  005037  040574                  CLR     TRNSWT          ;CLEAR THE TRANSFER QUEUE
220 043166  005201                  6$:     INC     R1              ;MOVE TO THE NEXT DRIVE
221 043170  062703  000002                  ADD     #2,R3
222 043174  042701  177770                  BIC     #^C7,R1
223 043200  001321                          BNE     1$              ;BRANCH IF MORE DRIVES
224 043202  012737  177777  040626          MOV     #-1,DTUW        ;NO DATA TRANSFERS UNDERWAY
225 043210  005037  040574                  CLR     TRNSWT          ;CLEAR THE 'TRANSFER WAIT' QUEUE
226 043214  004737  046052                  JSR     PC,CLRQUE       ;CLEAR ALL OF THE REQUEST QUEUES
227 043220  012764  000040  000010          MOV     #BIT05,RPCS2(R4) ;DO A MASSBUS INIT.
228 043226  000406                          BR      8$              ;CONTINUE
229
230 043230  004737  046130          7$:     JSR     PC,EMPTYQ       ;CLEAR THE DRIVE'S QUEUE
231 043234  105061  040534                  CLRB    DRVSTA(R1)      ;SET DRIVE TO OFFLINE
232 043240  105061  040544                  CLRB    DRVTYP(R1)      ;CLEAR THE DRIVE TYPE INDICATOR
233 043244  004737  045506          8$:     JSR     PC,SET.IE       ;SET "IE" WITHOUT "TRE"
234 043250  104413                          RESREG                  ;RESTORE R0 - R5
235 043252  000207                          RTS     PC              ;RETURN
```

```
                                        ;INTERRUPT SERVICE ROUTINE
                                        ;
043254  112737  000001  040600  ISR:    MOVB    #1,ACTDRV       ;SET "ACTIVE DRIVER' FLAG
043262  104412                          SAVREG                  ;SAVE R0 - R5
043264  013704  040640                  MOV     RPADR,R4        ;ADDRESS OF RPCS1
043270  013701  040626                  MOV     DTUW,R1         ;GET "DATA TRANSFER UNDERWAY" INDICATOR
043274  002402                          BLT     1$              ;BRANCH IF NO DATA TRANSFER UNDERWAY
043276  004737  043404                  JSR     PC,TD           ;CALL TRANSFER DONE
043302  004737  043630          1$:     JSR     PC,SC           ;CALL SPECIAL CONDITIONS
043306  104413                          RESREG                  ;RESTORE R0   R5
043310  105037  040600                  CLRB    ACTDRV          ;CLEAR "ACTIVE DRIVER" FLAG
043314  000002                          RTI                     ;RETURN
                                        ;
                                        ;FORCE WRITE CHECK ROUTINE
                                        ;
043316  005737  001506          WC.HK:  TST     WRTCHK          ;DO WRITE CHECK ?
043322  001427                          BEQ     1$              ;BR IF NO
043324  122762  000161  000002          CMPB    #WRTDAT,$COMND(R2)      ;LAST COMMAND A WRITE COMMAND ?
043332  001023                          BNE     1$              ;BRANCH IF NOT
043334  004037  046150                  JSR     R0,DRVQUE       ;PUT INTO THE QUEUE
043340  000420                          BR      1$              ;BRANCH IF QUEUE IS FULL
                                        ;
043342  005062  000016                  CLR     $TATUS(R2)      ;CLEAR 'DONE' BIT IN DPB
043346  116262  000166  000027          MOVB    $RPCS1(R2),$PREVO(R2)   ;PREVIOUS COMMAND
043354  016262  000012  000034          MOV     $CYL(R2),$PREVA+2(R2)   ;PREVIOUS CYLINDER ADDRESS
043362  016262  000010  000032          MOV     $SEC(R2),$PREVA(R2)     ;PREVIOUS SEC , TRK ADDRESSES
043370  105062  000024                  CLRB    $CODE(R2)       ;CHANGE WRITE DATA TO WRITE CHECK DATA
043374  112762  000151  000002          MOVB    #WCKD,$COMND(R2)        ;CHANGE FUNCTION CODE TO WRITE CHECK
043402  000207                  1$:     RTS     PC              ;EXIT
```

```
                                        ;TRANSFER DONE ROUTINE
                                        
043404  005037  043626          TD:     CLR     PERM            ;CLR PERMENANT ERROR FLAG
                                        
043410  105061  040524                  CLRB    DRVACT(R1)      ;SET DRIVE ACTIVE INDICATOR TO IDLE
043414  012737  177777  040626          MOV     #1,DTUW         ;NO DATA TRANSFERS UNDERWAY
043422  006301                          ASL     R1
043424  012761  177777  040606          MOV     #1,TIMER(R1)    ;CANCEL TIMEOUT
043432  006201                          ASR     R1
043434  013702  040574                  MOV     TRNSWT,R2       ;GET "DPB" FROM THE "TRANSFER WAIT" QUEUE
043440  005037  040574                  CLR     TRNSWT          ;TRANSFER WAIT QUEUE--CLEAR QUEUE
043444  052762  000200  000016          BIS     #BIT07,16(R2)   ;SET DONE
043452  010164  000010                  MOV     R1,RPCS2(R4)    ;SELECT THE DRIVE
043456  004037  045116                  JSR     R0,RD.RP        ;TRANSFER ERROR(TRE=1)?
043462  000000                          RPCS1
043464  042750                          CI7
043466  006126                          ROL     (SP)+
043470  100430                          BMI     4$              ;BR IF YES
043472  005737  040602                  TST     SAVEFG          ;SAVE RHXX/RPO7 REGISTERS ?
043476  100002                          BPL     1$              ;BR IF NO
043500  004737  045342                  JSR     PC,SVRHXX       ;YES--SAVE THE REGISTERS
043504  122762  000135  000002  1$:     CMPB    #135,2(R2)      ;IF FROM DIAGNOSTIC COMMAND ?
043512  001003                          BNE     2$              ;BRANCH IF NOT
043514  116164  040630  000016          MOVB    ATABIT(R1),RPAS(R4)     ;RESET THE ATA BIT
                                        
043522  004737  043316          2$:     JSR     PC,WC.MK        ;SEE IF WRITE CHECK TO BE PUT IN QUEUE
043526  004737  046224                  JSR     PC,GETREQ       ;GET DPB POINTER
043532  005702                          TST     R2              ;ENTRY FOR DRIVE ?
043534  001403                          BEQ     3$              ;BR IF NOT
043536  004737  041622                  JSR     PC,OPT          ;CALL OPTIMIZER
043542  000207                          RTS     PC              ;RETURN
043544  012714  000113          3$:     MOV     #113,(R4)       ;RELEASE THE DRIVE
043550  000207                          RTS     PC              ;RETURN
                                        
043552  052762  100100  000016  4$:     BIS     #BIT15!BIT06,16(R2) ;SET DATA ERROR FLAG
043560  004737  046130                  JSR     PC,EMPTYQ       ;EMPTY THE "DRIVE'S WAIT" QUEUE
043564  004737  045342                  JSR     PC,SVRHXX       ;SAVE THE RHXX/RPO7 REGISTERS
043570  012714  040111                  MOV     #40111,(R4)     ;ISSUE A "DRIVE CLEAR"
                                        
043574  032764  040000  000012          BIT     #BIT14,RPDS(R4) ;DID ERROR BIT CLEAR?
043602  001406                          BEQ     5$              ;BR IF YES
043604  005237  043626                  INC     PERM            ;SET PERM ERROR FLAG
043610  116164  040630  000016          MOVB    ATABIT(R1),RPAS(R4)     ;CLR ATA BIT
043616  000207                          RTS     PC              ;RETURN
                                        
043620  012714  000113          5$:     MOV     #113,(R4)       ;ISSUE A RELEASE TO THE DRIVE
043624  000207                          RTS     PC              ;RETURN
                                        
043626  000000                  PERM:   .WORD   0               ;PERMENANT ERROR FLAG
```

```
                                ;SPECIAL CONDITION ROUTINE
                                ;
043630  116403  000016  SC:     MOVB    RPAS(R4),R3     ;READ "RPAS" REGISTER
043634  001013                  BNE     2$              ;BRANCH IF ANY 'ATA' BITS SET
043636  004037  045116          JSR     R0,RD.RP        ;READ CONTROL AND STATUS REGISTER
043642  000000                  RPCS1
043644  043036                  CI8
043646  106126                  ROLB    (SP)+           ;IS "IE"=1?
043650  100404                  BMI     1$              ;YES, NO DRIVES TO CHECK
043652  000401                  BR      1$+4            ;BRANCH OVER ERROR, REPLACE BRANCH WITH 'NOP'
                                                        ;IF ERROR REPORT IS DESIRED
043654  104001                  EMT     1               ;REPORT ILLEGAL INTERRUPT
043656  004737  045506          JSR     PC,SET.IE       ;SET INTERRUPT ENABLE
043662  000207          1$:     RTS     PC              ;RETURN
                                ;
043664  005046          2$:     CLR     -(SP)           ;PROCESS ALL DRIVES THAT HAVE
043666  110316                  MOVB    R3,(SP)         ;STORE ATA BITS ON STACK
043670  005001                  CLR     R1              ;SETUP R1 & R3 TO CHECK DRIVE 0
043672  012703  000001          MOV     #1,R3
043676  030316          SC3:    BIT     R3,(SP)         ;IS THIS ATA BIT SET ?
043700  001005                  BNE     SC5             ;BR IF YES
043702  005201          SC4:    INC     R1              ;MOVE TO THE NEXT DRIVE
043704  106303                  ASLB    R3              ;ANY MORE DRIVES TO CHECK ?
043706  001373                  BNE     SC3             ;BR IF YES
043710  005726                  TST     (SP)+           ;CLEAN OFF THE STACK
043712  000207                  RTS     PC
                                ;
043714  105761  040564  SC5:     TSTB    DPRQS(R1)       ;IS THERE PORT REQUEST OUTSTANDING ?
043720  001402                  BEQ     1$              ;BR IF NO
043722  000137  044264          JMP     SC13            ;START THE OUTSTANDING COMMAND
                                ;
043726  105761  040534  1$:     TSTB    DRVSTA(R1)      ;IS THIS DRIVE ON-LINE ?
043732  003011                  BGT     2$              ;BR IF YES
043734  004737  046224          JSR     PC,GETREQ       ;GET DPB POINTER
043740  004737  045342          JSR     PC,SVRHXX       ;SAVE THE RHXX/RP07 REGISTERS
043744  004737  044200          JSR     PC,SC12         ;SAVE RPDS, RPER1, RPER2 AND RPER3
                                                        ;ALSO DO A DRIVE INIT (DRVINT)
043750  105761  040534          TSTB    DRVSTA(R1)      ;DID DRIVE COME ON-LINE ?
043754  003405                  BLE     3$              ;BR IF NO
043756  105761  040524  2$:     TSTB    DRVACT(R1)      ;DRIVE ACTIVE WITH COMMAND OR ERROR RECOVERY ?
043762  001020                  BNE     SC6             ;BR IF EITHER
043764  004737  044200          JSR     PC,SC12         ;SAVE RPDS, RPER1, RPER2 AND RPER3
                                                        ;ALSO DO A DRIVE INIT (DRVINT)
043770  105761  040534  3$:     TSTB    DRVSTA(R1)      ;CHECK ON DRIVE'S STATUS
043774  100412                  BMI     4$              ;BR IF UNSAFE
043776  012746  000111          MOV     #111,-(SP)      ;DRIVE CLEAR
044002  004037  045210          JSR     R0,WRT.RP       ;WRITE THE COMMAND INTO RPCS1
044006  000000                  RPCS1                   ;REGISTER INDEX
044010  044052                  SC8                     ;PARITY EXIT ADDRESS
044012  116164  040630  000016  MOVB    ATABIT(R1),RPAS(R4)     ;CLR ATTN BIT
044020  000240                  NOP
044022  000727          4$:     BR      SC4             ;CHECK FOR MORE DRIVES
                                ;
044024  006301          SC6:    ASL     R1              ;SETUP TO ADDRESS WORDS
044026  012761  177777  040606  MOV     #-1,TIMER(R1)   ;STOP THE TIMER
044034  006201                  ASR     R1              ;RESTORE THE DRIVE ADDRESS
044036  004737  046224          JSR     PC,GETREQ       ;GET THE DPB POINTER FROM THE QUEUE
```

```
 61 044042  010164  000010                    MOV     R1,RPCS2(R4)     ;SELECT DRIVE
 62 044046  000137  044102                    JMP     SC11             ;PROCESS THE SEARCH
 63
 64 044052  105761  040524          SC8:      TSTB    DRVACT(R1)       ;IS DRIVE IDLE?
 65 044056  001405                             BEQ     1$               ;YES--BRANCH
 66 044060  004737  046224                    JSR     PC,GETREQ        ;GET DPB POINTER
 67 044064  004737  042750                    JSR     PC,CI7           ;PROCESS THE PARITY ERROR
 68 044070  000402                             BR      2$               ;CONTINUE
 69
 70 044072  004737  042764          1$:       JSR     PC,CI7B          ;PROCESS THE UNCORRECTABLE PARITY ERROR
 71 044076  000137  043702          2$:       JMP     SC4              ;CHECK MORE DRIVES
 72
 73 044102  105061  040524          SC11:     CLRB    DRVACT(R1)       ;SET DRIVE IDLE
 74 044106  136137  040630  040576            BITB    ATABIT(R1),SRCHWT  ;DOING A SEARCH OPERATION FOR
 75                                                                     ;AN I/O COMMAND?
 76 044114  001012                             BNE     1$               ;BRANCH IF YES
 77 044116  004737  046246                    JSR     PC,POPQUE        ;REMOVE REQUEST FROM QUEUE
 78 044122  052762  000200  000016            BIS     #BIT07,16(R2)    ;SET "DONE" BIT
 79 044130  005737  040602                    TST     SAVEFG           ;SAVE THE REGISTERS?
 80 044134  100002                             BPL     1$               ;BR IF NO
 81 044136  004737  045342                    JSR     PC,SVRHXX        ;YES--SAVE ALL OF THE RHXX/RP07 REG'S
 82 044142  116164  040630  000016  1$:       MOVB    ATABIT(R1),RPAS(R4)  ;CLEAR ATTENTION BIT
 83 044150  146137  040630  040576            BICB    ATABIT(R1),SRCHWT      ;CLEAR IMPLIED SEEK SET
 84 044156  006301                             ASL     R1               ;WORD INDEX
 85 044160  012761  177777  040606            MOV     #-1,TIMER(R1)    ;STOP CLOCK
 86 044166  006201                             ASR     R1               ;RESTORE R1
 87 044170  004737  041622                    JSR     PC,OPT           ;START A REQUEST
 88 044174  000137  043702                    JMP     SC4              ;CHECK FOR MORE DRIVES
 89
 90 044200  010164  000010          SC12:     MOV     R1,RPCS2(R4)     ;SELECT DRIVE
 91 044204  006301                             ASL     R1
 92 044206  006301                             ASL     R1
 93 044210  006301                             ASL     R1
 94 044212  016461  000012  040424            MOV     RPDS(R4),RPSTU0(R1)      ;STORE DRIVE STATUS
 95 044220  016461  000014  040426            MOV     RPER1(R4),RPSTU0+2(R1)   ;STORE ERROR REG #1
 96 044226  016461  000040  040430            MOV     RPER2(R4),RPSTU0+4(R1)   ;STORE ERROR REG #2
 97 044234  016461  000042  040432            MOV     RPER3(R4),RPSTU0+6(R1)   ;STORE ERROR REG #3
 98 044242  006201                             ASR     R1
 99 044244  006201                             ASR     R1
100 044246  006201                             ASR     R1
101 044250  004037  041030                    JSR     R0,DRVINT        ;INIT. THE STATE OF THE DRIVE
102 044254  000401                             BR      1$               ;TAKE ERROR EXIT
103 044256  000207                             RTS     PC               ;RETURN
104 044260  005726                  1$:       TST     (SP)+            ;CLEAR THE STACK
105 044262  000673                             BR      SC8              ;PROCESS THE PARITY ERROR
106
112 044264  010164  000010          SC13:     MOV     R1,RPCS2(R4)     ;SELECT THE DRIVE
113 044270  116164  040630  000016            MOVB    ATABIT(R1),RPAS(R4)  ;CLEAR THE ATTENTION BIT
114 044276  105761  040554          1$:       TSTB    DPINT(R1)        ;INITIALIZING THE DRIVE ?
115 044302  001424                             BEQ     2$               ;BR IF NOT
116 044304  105061  040554                    CLRB    DPINT(R1)        ;CLEAR THE INIT INDICATOR
117 044310  004037  041030                    JSR     R0,DRVINT        ;GO INIT THE DRIVE
118 044314  000240                             NOP                      ;DUMMY PARITY ERROR RETURN
119 044316  105761  040534                    TSTB    DRVSTA(R1)       ;DRIVE ONLINE ?
120 044322  003014                             BGT     2$               ;BR IF YES -- START ORDER
121 044324  005702                             TST     R2               ;QUEUE ENTRY FOR THE DRIVE
122 044326  001423                             BEQ     4$               ;BR IF NOT
```

```
044330  004737  046224                  JSR     PC,GETREQ       ;GET DPB ADDRESS
044334  052762  140000  000016          BIS     #BIT15!BIT14,16(R2)  ;INFORM USER THAT DRIVE OFFLINE
044342  004737  045342                  JSR     PC,SVRHXX       ;SAVE THE REGISTERS
044346  004737  046246                  JSR     PC,POPQUE       ;REMOVE THE QUEUE
044352  000411                          BR      4$

044354  032764  004000  000000  2$:     BIT     #BIT11,RPCS1(R4)        ;DVA SET ?
044362  001003                          BNE     3$              ;SET THEN CALL OPT
044364  004737  045506                  JSR     PC,SET.IE
044370  000402                          BR      4$

044372  004737  041622          3$:     JSR     PC,OPT          ;START THE REQUEST
044376  000137  043702          4$:     JMP     SC4             ;PROCESS OTHER DRIVES
```

```
                                        ;RP07 TIMER ROUTINE
                                        ;CALL
                                        ;       MOV     #TIME,-(SP)     ;ELASPED TIME IN MILLISECONDS ON THE STACK
                                        ;       JSR     PC,RPTMR        ;CALL RP07 TIME ROUTINE
                                        
044402  005737  040600          RPTMR:  TST     ACTDRV          ;CHECK "ACTDRV & ACTSTR"
044406  001027                  4$      BNE     4$              ;IF NON ZERO EXIT
044410  112737  000001  040601          MOVB    #1,ACTSTR       ;SET "ACTSTR"
044416  104412                          SAVREG                  ;SAVE R0 - R5
044420  005001                          CLR     R1              ;START WITH DRIVE 0
044422  005003                          CLR     R3
044424  005763  040606          1$:     TST     TIMER(R3)       ;IS THE TIMER RUNNING?
044430  002406                          BLT     2$              ;BRANCH IF NO
044432  166663  000002  040606          SUB     2(SP),TIMER(R3) ;COUNT THE INTERVAL
044440  003002                          BGT     2$              ;BR IF NO SOFTWARE TIMEOUT
044442  004737  044472                  JSR     PC,STO          ;CALL SOFTWARE TIMEOUT ROUTINE
044446  005201                  2$:     INC     R1              ;MOVE TO NEXT DRIVE
044450  005723                          TST     (R3)+
044452  022701  000010                  CMP     #8.,R1          ;OUT OF DRIVES?
044456  003362                          BGT     1$              ;BRANCH IF NO
044460  104413                  3$:     RESREG                  ;RESTORE R0 - R5
044462  105037  040601                  CLRB    ACTSTR          ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
044466  012616                  4$:     MOV     (SP)+,(SP)      ;ADJUST THE STACK
044470  000207                          RTS     PC              ;RETURN
```

```
                                        ;SOFTWARE TIMEOUT ROUTINE
                                        ;
                                        ;NOTE: THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
                                        ;       OR GREATER
                                        ;
                                        ;CALL
                                        ;       MOV     @DRVNUM,R1      ;DRIVE NUMBER
                                        ;       JSR     PC,STO          ;SOFTWARE TIME ROUTINE
                                        ;       RETURN
044472  010146                  STO:    MOV     R1,-(SP)        ;SAVE R1
044474  010346                          MOV     R3,-(SP)        ;SAVE R3
044476  013704  040640                  MOV     RPADR,R4        ;GET ADDRESS OF "RPCS1"
044502  010164  000010                  MOV     R1,RPCS2(R4)    ;SELECT THE DRIVE
044506  004037  045116                  JSR     R0,RD.RP        ;READ THE DRIVE STATUS REGISTER
044512  000012                          RPDS
044514  045004                          9$
044516  105726                          TSTB    (SP)+           ;IS "DRY"=1?
044520  100473                          BMI     6$              ;BR IF YES
044522  105761  040554          1$:     TSTB    DPINT(R1)       ;TRYING TO INTIALIZE THE DRIVE ?
044526  001070                          BNE     6$              ;BR IF YES
044530  105761  040564                  TSTB    DPRQS(R1)       ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
044534  001065                          BNE     6$              ;BR IF YES
044536  013702  040574                  MOV     TRNSWT,R2       ;GET "DPB" FROM THE "TRANSFER WAIT" QUEUE
044542  020137  040626                  CMP     R1,DTUW         ;TRANSFER UNDERWAY ON THIS DRIVE?
044546  001402                          BEQ     2$              ;BRANCH IF YES
044550  004737  046224                  JSR     PC,GETREQ       ;GET DPB ADDRESS
044554  052762  101000  000016  2$:     BIS     @BIT15!BIT09,16(R2)     ;SET THE ERROR FLAGS
044562  004737  045342                  JSR     PC,SVRHXX       ;SAVE RHXX/RP07 REGISTERS
044566  012764  000040  000010          MOV     @BIT05,RPCS2(R4)        ;"INIT" THE MASS BUS
044574  105061  040524                  CLRB    DRVACT(R1)      ;DRIVE IS IDLE
044600  005001                          CLR     R1              ;START WITH DRIVE 0
044602  005003                          CLR     R3
044604  004037  041030          3$:     JSR     R0,DRVINT       ;INIT. THIS DRIVE
044610  000475                          BR      9$              ;PARITY ERROR RETURN

044612  105761  040524                  TSTB    DRVACT(R1)      ;DRIVE IDLE BEFORE THE INIT.?
044616  001414                          BEQ     5$              ;YES--BRANCH
044620  013702  040574                  MOV     TRNSWT,R2       ;GET "DPB" FROM THE  'TRANSFER WAIT" QUEUE
044624  023701  040626                  CMP     DTUW,R1         ;WAS A DATA TRANSFER UNDERWAY ON THIS DRIVE?
044630  001402                          BEQ     4$              ;YES--BRANCH
044632  004737  046224                  JSR     PC,GETREQ       ;GET THE DPB POINTER FROM QUEUE
044636  052762  100400  000016  4$:     BIS     @BIT15!BIT08,16(R2)     ;INFORM USER OF INIT.
044644  105061  040524                  CLRB    DRVACT(R1)      ;SET DRIVE ACTIVE TO IDLE
044650  012763  177777  040606  5$:     MOV     @-1,TIMER(R3)   ;STOP THE TIMER
044656  005723                          TST     (R3)+           ;UPDATE THE INDEX
044660  005201                          INC     R1              ;INCREMENT THE DRIVE NUMBER
044662  022701  000010                  CMP     @8.,R1          ;LAST DRIVE BEEN CHECKED?
044666  003346                          BGT     3$              ;NO--LOOP
044670  012737  177777  040626          MOV     @-1,DTUW        ;NO DATA TRANSFERS UNDERWAY
044676  005037  040574                  CLR     TRNSWT          ;CLEAR TRANSFER WAIT QUEUE
044702  004737  046052                  JSR     PC,CLRQUE       ;CLEAR ALL REQUEST QUEUES
044706  000500                          BR      13$             ;EXIT

044710  116405  000016          6$:     MOVB    RPAS(R4),R5     ;READ ATTENTION REG
044714  136105  040630                  BITB    ATABIT(R1),R5   ;IS ATTENTION FOR THIS DRIVE UP ?
044720  001017                          BNE     7$              ;YES--BRANCH
```

```
     58 044722  105761  040554                TSTB    DPINT(R1)          ;TRYING TO INTIALIZE THE DRIVE ?
     59 044726  001031                      BNE     10$                ;BR IF YES - DRIVE NOT ONLINE
     60 044730  105761  040564                TSTB    DPRQS(R1)          ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
     61 044734  001045                      BNE     11$                ;BR IF YES - NO RESPONSE TO REQUEST
     62 044736  020137  040626                CMP     R1,DTUW            ;DATA TRANSFER UNDERWAY FOR THIS DRIVE
     63 044742  001267                      BNE     1$                 ;BR IF NO
     64 044744  004037  045116                JSR     R0,RD.RP           ;READ CONTROL AND STATUS REGISTER
     65 044750  000000                      RPCS1
     66 044752  045004                      9$
     67 044754  105726                      TSTB    (SP)+              ;CHECK 'RDY'
     68 044756  100261                      BPL     1$                 ;BR IF "RDY"=0
     69 044760  105761  040554          7$: TSTB    DPINT(R1)          ;INITIALIZING THE DRIVE ?
     70 044764  001003                      BNE     8$                 ;BR IF INIT PENDING
     71 044766  105761  040564                TSTB    DPRQS(R1)          ;PORT REQUEST PENDING ?
     72 044772  001446                      BEQ     13$                ;BR IF NOT
     73 044774  012763  177777  040606  8$: MOV     #-1,TIMER(R3)      ;STOP THE TIMER
     74 045002  000442                      BR      13$                ;EXIT
     75
     76 045004  004737  043036          9$: JSR     PC,CI8             ;GO HANDLE THE 'NED'
     77 045010  000437                      BR      13$
     78
     79 045012  105061  040554         10$: CLRB    DPINT(R1)          ;CLEAR THE INITIALIZE INDICATOR
     80 045016  105061  040534                CLRB    DRVSTA(R1)         ;SET DRIVE OFFLINE
     81 045022  012763  177777  040606        MOV     #-1,TIMER(R3)      ;STOP THE TIMER
     82 045030  004737  046224                JSR     PC,GETREQ          ;GET THE DPB ADDRESS
     83 045034  005702                      TST     R2                 ;REQUEST IN QUEUE ?
     84 045036  001424                      BEQ     13$                ;BR IF NOT
     85 045040  052762  140000  000016        BIS     #BIT15!BIT14,16(R2)     ;INFORM THE USER DRIVE NOT AVAILABLE
     86 045046  000414                      BR      12$                ;FINISH
     87
     88 045050  012763  177777  040606 11$: MOV     #-1,TIMER(R3)      ;STOP THE TIMER
     89 045056  105061  040564                CLRB    DPRQS(R1)          ;CLEAR PORT REQUEST INDICATOR
     90 045062  004737  046224                JSR     PC,GETREQ          ;GET DPB ADDRESS
     91 045066  005702                      TST     R2                 ;QUEUE ENTRY FOR DRIVE ?
     92 045070  001407                      BEQ     13$                ;BR IF NONE
     93 045072  012762  100004  000016        MOV     #BIT15!BIT2,16(R2)      ;INFORM USER OF PORT REQUEST ERROR
     94 045100  004737  046130         12$: JSR     PC,EMPTYQ          ;CLEAR THE QUEUE FOR THE DRIVE
     95 045104  004737  045342                JSR     PC,SVRHXX          ;SAVE RHXX/RP07 REGISTERS
     96 045110  012603                 13$: MOV     (SP)+,R3           ;RESTORE R3
     97 045112  012601                      MOV     (SP)+,R1           ;RESTORE R1
     98 045114  000207                      RTS     PC                 ;RETURN
```

M1


```
                                        ;ROUTINE TO READ A RHXX/RP07 REGISTER
                                        ;
                                        ;CALL
                                        ;       JSR     R0,RD.RP        ;GO READ A REGISTER
                                        ;       INDEX                   ;REG. INDEX FROM BASE
                                        ;       ERRADR                  ;ERROR ADDRESS- PROCESS ERROR STARTING
                                        ;                               ;AT THIS ADDRESS
                                        ;       RETURN                  ;CONTENTS OF REG. IS ON THE STACK

045116  011646                  RD.RP:  MOV     (SP),-(SP)      ;SAVE R0
045120  013746  040640                  MOV     RPADR,-(SP)     ;ADDRESS OF THE
045124  062016                          ADD     (R0)+,(SP)      ;REG
045126  017666  000000  000004          MOV     @(SP),4(SP)     ;READ THE CONTENTS OF THE REG
045134  013716  040640                  MOV     RPADR,(SP)      ;CHECK IF NON EXIST DRIVE
045140  062716  000010                  ADD     #RPCS2,(SP)     ;
045144  032776  010000  000000          BIT     #BIT12,@(SP)    ;NED BIT SET ?
045152  001004                          BNE     1$              ;ERROR EXIT
045154  032777  020000  173456          BIT     #BIT13,@RPADR   ;NCPE SET ?
045162  001406                          BEQ     2$              ;EXIT
045164  016666  000002  000004  1$:     MOV     2(SP),4(SP)     ;MOVE THE R0 TO TOP OF STACK
045172  022626                          CMP     (SP)+,(SP)+     ;CLEAR OFF THE STACK
045174  011000                          MOV     (R0),R0         ;ERROR EXIT ADDRESS
045176  000403                          BR      3$              ;EXIT

045200  062700  000002          2$:     ADD     #2,R0           ;NORMAL EXIT
045204  005726                          TST     (SP)+           ;CLEAR OFF STACK
045206  000200                  3$:     RTS     R0              ;EXIT
```

```
                                        ;ROUTINE TO WRITE A REGISTER
                                        ;
                                        ;CALL
                                        ;       MOV     DATA, (SP)      ;DATA TO BE LOADED ON THE STACK
                                        ;       JSR     R0,WRT.RP       ;CALL THE ROUTINE TO LOAD(WRITE) THE REG.
                                        ;       INDEX                   ;INDEX OF THE REGISTER TO BE LOADED
                                        ;       ERRADR                  ;ADDRESS TO RETURN TO ON AN ERROR
                                        ;       RETURN                  ;ERROR FREE RETURN

045210                                  WRT.RP:
045210  012046                                  MOV     (R0)+,-(SP)     ;FORMING THE REG ADDRESS
045212  001017                                  BNE     2$              ;BRANCH IF NOT RPCS1
045214  122766  000150  000004                  CMPB    #150,4(SP)      ;DATA XTRNS COMMAND ?
045222  002413                                  BLT     2$              ;BRANCH IF NOT
045224  105777  173410          1$:             TSTB    @RPADR          ;SEE IF CONTROLLER READY
045230  100375                                  BPL     1$
045232  017746  173402                          MOV     @RPADR,-(SP)    ;READ RPCS1
045236  000316                                  SWAB    (SP)            ;MERG THE A17,A18,PSEL BITS
045240  042716  177770                          BIC     #^C7,(SP)       ;CHOP OFF THE REST BITS FROM RPCS1
045244  111666  000007                          MOVB    (SP),7(SP)      ;ATTACH A17,A18,PSEL TO COMMAND
045250  005726                                  TST     (SP)+           ;RESTORE STACK LEVEL
045252  063716  040640          2$:             ADD     RPADR,(SP)      ;THE DEST REG ADDRESS
045256  016676  000004  000000                  MOV     4(SP),@(SP)     ;WRITE THE REGISTER
045264  013716  040640                          MOV     RPADR,(SP)      ;CHECK NED,PAR BITS
045270  062716  000010                          ADD     #RPCS2,(SP)     ;
045274  032776  010000  000000                  BIT     #BIT12,@(SP)    ;NONE EXIST DRIVE ?
045302  001013                                  BNE     3$              ;BRANCH IF IT IS
045304  013716  040640                          MOV     RPADR,(SP)      ;ADDRESS RPER1
045310  062716  000014                          ADD     #RPER1,(SP)     ;
045314  032776  000010  000000                  BIT     #BIT3,@(SP)     ;PAR SET ?
045322  001003                                  BNE     3$              ;BRANCH IF SO
045324  062700  000002                          ADD     #2,R0           ;NORMAL RETURN
045330  000401                                  BR      4$              ;EXIT

045332  011000                  3$:             MOV     (R0),R0         ;ERROR EXIT
045334  005726                  4$:             TST     (SP)+           ;CLEAR OFF THE STACK
045336  012615                                  MOV     (SP)+,(SP)      ;MOVE R0 TO TOP OF STACK
045340  000200                                  RTS     R0              ;EXIT
```

```
                                        ;ROUTINE TO SAVE THE RHXX/RP07 REGISTERS AS PER DPB.14
                                        ;
                                        ;CALL
                                        ;       MOV     #DPB,R2         ;ADDRESS OF DRIVE PARAMETER BLOCK
                                        ;       JSR     PC,SVRHXX       ;SAVE THE DRIVES REG'S
                                        ;
045342  104412                  SVRHXX: SAVREG                          ;SAVE R0   R5
045344  005702                          TST     R2                      ;QUEUE ENTRY FOR THE DRIVE ?
045346  001455                          BEQ     7$                      ;BR IF NONE
045350  013704  040640                  MOV     RPADR,R4
045354  111264  000010                  MOVB    (R2),RPCS2(R4)          ;SELECT DRIVE
045360  016203  000014                  MOV     14(R2),R3               ;GET THE ERROR TABLE POINTER
045364  001446                          BEQ     7$                      ;EXIT IF NO ADDRESS
045366  005037  045422                  CLR     3$                      ;COUNTER & POINTER
045372  023727  045422  000020  1$:     CMP     3$,#RPDB                ;REACHED THE BUFFER REGISTER ?
045400  001006                          BNE     2$                      ;BR IF NOT
045402  032764  000200  000010          BIT     #OPT07,RPCS2(R4)                ;'OR' SET ?
045410  001002                          BNE     2$                      ;BR IF SET
045412  005023                          CLR     (R3)+                   ;STORE RPDB AS ZEROES
045414  000405                          BR      4$                      ;CONTINUE
                                        
045416  004037  045116          2$:     JSR     R0,RD.RP                ;READ THE SELECTED REGISTER
045422  000000                  3$:     .WORD   0                       ;REGISTER INDEX
045424  045450                          5$                              ;ERROR RETURN ADDRESS
045426  012623                          MOV     (SP)+,(R3)+             ;STORE THE REGISTER CONTENTS
045430  023727  045422  000046  4$:     CMP     3$,#RPEC2               ;REACHED THE END ?
045436  001406                          BEQ     6$                      ;BR IF YES
045440  062737  000002  045422          ADD     #2,3$                   ;INCREMENT THE REGISTER INDEX
045446  000751                          BR      1$                      ;CONTINUE READING THE REGISTERS
                                        
045450  004737  042750          5$:     JSR     PC,CI7                  ;PROCESS THE UNCORRECTABLE PARITY ERROR
045454  013746  001234          6$:     MOV     $CPUOP,-(SP)            ;CHECK THE CPU (RH) TYPE
045460  042716  003777                  BIC     #^C174000,(SP)          ;LEAVE THE CPU TYPE BITS
045464  022726  030000                  CMP     #30000,(SP)+            ;SEE IF RH70
045470  001004                          BNE     7$                      ;IF NE, NO
045472  063704  040646                  ADD     RHEXT,R4                ;POINT TO RPBAE
045476  012423                          MOV     (R4)+,(R3)+             ;STORE THE CONTENTS
045500  011413                          MOV     (R4),(R3)               ;GET RPCS3
045502  104413                  7$:     RESREG                          ;RESTORE R0 - R5
045504  000207                          RTS     PC                      ;RETURN
```

```
                                ;ROUTINE TO SET THE INTERRUPT ENABLE (IE) BIT IN RPCS1 WITHOUT GETTING A
                                ;TRANSFER ERROR (TRE).
                                ;CALL
                                ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER
                                ;       JSR     PC,SET.IE       ;SET INTERRUPT ENABLE ROUTINE
                                ;       RETURN

045506  010446                  SET.IE: MOV     R4, (SP)        ;SAVE R4
045510  013704  040640                  MOV     RPADR,R4        ;PICKUP ADDRESS OF RPCS1
045514  010164  000010                  MOV     R1,RPCS2(R4)    ;SELECT DRIVE
045520  011446                          MOV     (R4), (SP)      ;READ RPCS1
045522  052716  040000                  BIS     #BIT14,(SP)     ;SET THE "TRE" BIT OF THE WORD READ
045526  000316                          SWAB    (SP)            ;ADJUST FOR DATO
045530  112714  000100                  MOVB    #BIT06,(R4)     ;SET "IE"
045534  032764  010000  000010          BIT     #BIT12,RPCS2(R4) ;IS "NED"=1?
045542  001002                          BNE     1$              ;YES--CLEAR  TRE
045544  005726                          TST     (SP)+           ;CLEAN OFF THE STACK
045546  000402                          BR      2$

045550  112664  000001          1$:     MOVB    (SP)+,1(R4)     ;CLEAR "TRE"
045554  012604                  2$:     MOV     (SP)+,R4        ;RESTORE R4
045556  000207                          RTS     PC              ;RETURN TO CALLER

                                ;QUEUE COUNT
045560      000                 QCNT:   .BYTE   0               ;DRIVE 0
045561      000                         .BYTE   0               ;DRIVE 1
045562      000                         .BYTE   0               ;DRIVE 2
045563      000                         .BYTE   0               ;DRIVE 3
045564      000                         .BYTE   0               ;DRIVE 4
045565      000                         .BYTE   0               ;DRIVE 5
045566      000                         .BYTE   0               ;DRIVE 6
045567      000                         .BYTE   0               ;DRIVE 7
```

```
                                ;QUEUE INPUT POINTERS
                                ;
045570  045652  QINPT:  .WORD   QDRV0           ;DRIVE 0
045572  045672          .WORD   QDRV1           ;DRIVE 1
045574  045712          .WORD   QDRV2           ;DRIVE 2
045576  045732          .WORD   QDRV3           ;DRIVE 3
045600  045752          .WORD   QDRV4           ;DRIVE 4
045602  045772          .WORD   QDRV5           ;DRIVE 5
045604  046012          .WORD   QDRV6           ;DRIVE 6
045606  046032          .WORD   QDRV7           ;DRIVE 7

                                ;QUEUE OUTPUT POINTERS

045610  045652  QOUTPT: .WORD   QDRV0           ;DRIVE 0
045612  045672          .WORD   QDRV1           ;DRIVE 1
045614  045712          .WORD   QDRV2           ;DRIVE 2
045616  045732          .WORD   QDRV3           ;DRIVE 3
045620  045752          .WORD   QDRV4           ;DRIVE 4
045622  045772          .WORD   QDRV5           ;DRIVE 5
045624  046012          .WORD   QDRV6           ;DRIVE 6
045626  046032          .WORD   QDRV7           ;DRIVE 7

045630  045652  QSTART: .WORD   QDRV0           ;DRIVE 0 START ADDRESS
045632  045672  QSTOP:  .WORD   QDRV1           ;DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
045634  045712          .WORD   QDRV2           ;STOP DRIVE 1--START DRIVE 2
045636  045732          .WORD   QDRV3           ;STOP DRIVE 2--START DRIVE 3
045640  045752          .WORD   QDRV4           ;STOP DRIVE 3 -START DRIVE 4
045642  045772          .WORD   QDRV5           ;STOP DRIVE 4- START DRIVE 5
045644  046012          .WORD   QDRV6           ;STOP DRIVE 5- START DRIVE 6
045646  046032          .WORD   QDRV7           ;STOP DRIVE 6  START DRIVE 7
045650  046052          .WORD   QTERP           ;STOP DRIVE 7
```

RPO" DRIVER

```
                                ;DRIVE REQUEST QUEUES
                                ;
045652                          QDRV0:  .BLKW   10
045672                          QDRV1:  .BLKW   10
045712                          QDRV2:  .BLKW   10
045732                          QDRV3:  .BLKW   10
045752                          QDRV4:  .BLKW   10
045772                          QDRV5:  .BLKW   10
046012                          QDRV6:  .BLKW   10
046032                          QDRV7:  .BLKW   10
        046052                  QTERP=.
                                ;
                                ;ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
                                ;
                                ;CALL
                                ;       JSR     PC,CLRQUE
                                ;
046052  104412                  CLRQUE: SAVREG                  ;SAVE R0 - R5
046054  012702  045560                  MOV     #QCNT,R2        ;ZERO THE QUEUE COUNTS
046060  005022                          CLR     (R2)+           ;DRIVES 0 & 1
046062  005022                          CLR     (R2)+           ;DRIVES 2 & 3
046064  005022                          CLR     (R2)+           ;DRIVES 4 & 5
046066  005022                          CLR     (R2)+           ;DRIVES 6 & 7
046070  012703  000010                  MOV     #8.,R3          ;MOVE THE STARTING
046074  012701  045630                  MOV     #QSTART,R1      ;ADDRESS OF THE QUEUE INTO
046100  012122                  1$:     MOV     (R1)+,(R2)+     ;THE QUEUE INPUT POINTER
046102  005303                          DEC     R3
046104  001375                          BNE     1$
046106  012703  000010                  MOV     #8.,R3          ;MOVE THE STARTING ADDRESS
046112  012701  045630                  MOV     #QSTART,R1      ;OF THE QUEUE INTO THE
046116  012122                  2$:     MOV     (R1)+,(R2)+     ;QUEUE OUTPUT POINTER
046120  005303                          DEC     R3
046122  001375                          BNE     2$
046124  104413                          RESREG                  ;RESTORE R0 - R5
046126  000207                          RTS     PC
```

```
                                        ;EMPTY THE QUEUE SPECIFIED BY R1
                                        ;
                                        ;CALL
                                        ;       MOV     @DRVNUM,R1      ;DRIVE NUMBER
                                        ;       JSR     PC,EMPTYQ
                                        ;
046130  105061  045560          EMPTYQ: CLRB    QCNT(R1)        ;CLEAR NUMBER OF ITEMS IN QUEUE
046134  006301                          ASL     R1
046136  016161  045570  045610          MOV     QINPT(R1),QOUTPT(R1) ;SET OUTPUT QUEUE POINTER=INPUT POINTER
046144  006201                          ASR     R1
046146  000207                          RTS     PC

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

046150  122761  000010  045560  DRVQUE: CMPB    #10,QCNT(R1)    ;IS QUEUE FULL?
046156  001421                          BEQ     2$              ;BR IF YES-TAKE RETURN1
046160  105261  045560                  INCB    QCNT(R1)        ;INCREMENT QUEUE COUNT
046164  006301                          ASL     R1
046166  010271  045570                  MOV     R2,@QINPT(R1)   ;PUT THIS REQUEST IN QUEUE
046172  062761  000002  045570          ADD     #2,QINPT(R1)    ;UPDATE THE QUEUE POINTER
046200  026161  045570  045632          CMP     QINPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER
046206  001003                          BNE     1$              ;BRANCH IF NO
046210  016161  045630  045570          MOV     QSTART(R1),QINPT(R1) ;YES--RESET POINTER
046216  006201                  1$:     ASR     R1
046220  005720                          TST     (R0)+           ;TAKE RETURN 2
046222  000200                  2$:     RTS     R0              ;RETURN TO USER
```

```
                                ;ROUTINE TO GET THE "DPB" ADDRESS OF NEXT REQUEST IN QUEUE
                                ;
                                ;ON RETURN, R2 WILL CONTAIN POINTER ADDRESS OF "DPB" REQUESTED OR
                                ;ZERO IF NO REQUEST IN QUEUE.
                                ;
                                ;CALL
                                ;       MOV     @DRVNUM,R1      ;DRIVE NUMBER
                                ;       JSR     PC,GETREQ       ;GO GET THE REQUEST
                                
046224  005002                  GETREQ: CLR     R2
046226  105761  045560                  TSTB    QCNT(R1)        ;IS THERE ANY REQUEST IN QUEUE?
046232  001404                          BEQ     2$              ;NO---BRANCH
046234  006301                  1$:     ASL     R1
046236  017102  045610                  MOV     @QOUTPT(R1),R2  ;PICKUP "DPB" POINTER FOR THIS DRIVE
046242  006201                          ASR     R1
046244  000207                  2$:     RTS     PC              ;RETURN TO USER

                                ;ROUTINE TO "POP" THE REQUEST FROM QUEUE
                                ;
                                ;ON RETURN, R2 WILL CONTAIN POINTER ADDRESS OF "DPB" REMOVED
                                ;
                                ;CALL
                                ;       MOV     @DRVNUM,R1      ;DRIVE NUMBER
                                ;       JSR     PC,POPQUE       ;CALL TO REMOVE REQUEST

046246  105361  045560          POPQUE: DECB    QCNT(R1)        ;DECREMENT QUEUE COUNT
046252  006301                          ASL     R1
046254  017102  045610                  MOV     @QOUTPT(R1),R2  ;GET THE "DPB" POINTER
046260  005071  045610                  CLR     @QOUTPT(R1)     ;REMOVE DPB ADDRESS FROM THE QUEUE
046264  062761  000002  045610          ADD     #2,QOUTPT(R1)   ;UPDATE THE QUEUE POINTER
046272  026161  045610  045632          CMP     QOUTPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER?
046300  001003                          BNE     1$              ;NO--BRANCH TO EXIT
046302  016161  045630  045610          MOV     QSTART(R1),QOUTPT(R1) ;YES--RESET THE POINTER
046310  006201                  1$:     ASR     R1
046312  000207                          RTS     PC              ;RETURN TO USER
```

```
                                .SBTTL  DEVICE PARAMETER BLOCKS
                                ;
                                ;BLOCK LOCATION EQUATE STATEMENTS
                                ;
000000                          ;       = 0             ;DRIVE NUMBER (BYTE)
000001                          $FMT    = 1             ;FMT,MCI,ECI OR OFFSET CODE (BYTE)
000002                          $COMND  = $FMT+1        ;OPERATION CODE (BYTE)
000003                          $PSEL   = $FMT+2        ;PORT SELECT & BITS A16, A17 (BYTE)
000004                          $WCNT   = $FMT+3        ;WORD COUNT (2'S COMP)
000006                          $BUF    = $FMT+5        ;BUFFER ADDR OR REGISTER TABLE POINTER
000010                          $SEC    = $FMT+7        ;SECTOR ADDRESS OR 1ST REG ADDR
000011                          $TRK    = $FMT+10       ;TRACK ADDRESS OF LAST REG ADDR
000012                          $CYL    = $FMT+11       ;CYLINDER ADDR
000014                          $REG    = $FMT+13       ;REGISTER STORAGE (IF ERROR)
000016                          $STATUS = $FMT+15       ;STATUS WORD (SET BY DRIVER)
                                ;
                                ;DRIVE'S HISTORY AND CURRENT INDICATOR STORAGE EQUATES
                                ;
000020                          $WRDL   = $FMT+17       ;WORD COUNT (NOT 2'S COMP)
000022                          $SSEC   = $WRDL+2       ;SECTOR SIZE FOR CURRENT OPERATION (256. OR 258 )
000024                          $CODE   = $WRDL+4       ;PRESENT COMMAND SELECTION CODE
000026                          $PACK   = $WRDL+6       ;READ/WRITE COMMAND INDICATOR (BYTE)
000027                          $PREVO  = $WRDL+7       ;PREVIOUS COMMAND SELECTION CODE (BYTE)
000030                          $PATTC  = $WRDL+10      ;PATTERN CODE
000032                          $PREVA  = $WRDL+12      ;PREVIOUS ADDRESS- TRK, SEC, CYL (DOUBLE WORD)
000036                          $WDPAS  = $WRDL+16      ;WDPOS READ PER PASS (DOUBLE WORD)
000042                          $WTPAS  = $WRDL+22      ;WDPOS WRITTEN PER PASS (DOUBLE WORD)
000046                          $SEEKS  = $WRDL+26      ;NUMBER OF SEEKS PER PASS (DOUBLE WORD)
000052                          $OPERC  = $WRDL+32      ;OPERATION COUNT (DOUBLE WORD)
000056                          $WTOTL  = $WRDL+36      ;TOTAL WORDS WRITTEN X10**6 (DOUBLE WORD) &
                                                        ;1 X10**6 REPETITION COUNTER (DOUBLE WORD)
000066                          $RTOTL  = $WRDL+46      ;TOTAL WORDS READ X10**6 (DOUBLE WORD) &
                                                        ;1 X10**6 REPETITION COUNTER (DOUBLE WORD)
000076                          $STOTL  = $WRDL+56      ;TOTAL SEEK COUNT (DOUBLE WORD)
000102                          $TOTAL  = $WRDL+62      ;TOTAL ERRORS COUNT (ALL TYPES)
000104                          $SOFT   = $WRDL+64      ;'SOFT' ERROR COUNT
000106                          $HARD   = $WRDL+66      ;'HARD' ERROR COUNT
000110                          $SKI    = $WRDL+70      ;'SKI' ERROR COUNT
000112                          $MISPO  = $WRDL+72      ;PROG DETECTED MIS-POSITIONING ERROR COUNT
000114                          $PASSC  = $WRDL+74      ;PASS COUNTER
000116                          $FAIR   = $WRDL+76      ;OPERATION QUEUE 'FAIRNESS' COUNT
000120                          $HLDWC  = $WRDL+100     ;HOLD WORD FOR 'RELBUF' ROUTINE
                                ;
                                ;INDEX EQUATES TO THE NEXT OPERATION PARAMETERS
                                ;
000122                          $NCODE  = $WRDL+102     ;NEXT OPERATION CODE
000123                          $NPATC  = $NCODE+1      ;NEXT PATTERN
000124                          $NSEC   = $NCODE+2      ;NEXT SECTOR
000125                          $NTRK   = $NCODE+3      ;NEXT TRACK
000126                          $NCYL   = $NCODE+4      ;NEXT CYLINDER
000130                          $NEXT   = $NCODE+6      ;PARAMETER SELECTION INDICATOR
000132                          $FIRST  = $NCODE+10     ;FIRST OPERATION INDICATOR
```

```
                        ;INDEX EQUATES FOR MAXIMUM/MINIMUM ADDRESSES
                        ;
000134          MAXCYL  = $NCODE+12             ;MAXIMUM CYLINDER ADDRESS
000136          MINCYL  = MAXCYL+2              ;MINIMUM CYLINDER ADDRESS
000140          MAXTRK  = MAXCYL+4              ;MAXIMUM TRACK ADDRESS
000142          MINTRK  = MAXCYL+6              ;MINIMUM TRACK ADDRESS
000144          MAXSEC  = MAXCYL+10             ;MAXIMUM SECTOR ADDRESS
000146          MINSEC  = MAXCYL+12             ;MINIMUM SECTOR ADDRESS
                        ;
                        ;INDEX EQUATES FOR CYLLMT, TRKLMT, SECLMT ADDRESSES LIMITS AND 1ST FE CYLINDER
                        ;
000150          CYLLMT  = MAXCYL+14             ;CYLINDER ADDRESS LIMIT
000152          SECLMT  = CYLLMT+2              ;SECTOR ADDRESS LIMIT
000154          TRKLMT  = CYLLMT+4              ;TRACK ADDRESS LIMIT
000156          FE1     = CYLLMT+6              ;1ST FE CYLINDER
                        ;
                        ;DRIVE SERIAL NUMBER AREA INDEX EQUATE
                        ;
000160          $DRVSN  = CYLLMT+10             ;DRIVE SERIAL NUMBER (6 BYTES)
                        ;
                        ;RP/RM REGISTER EQUATES
                        ;
000166          $RPCS1  = $DRVSN+6              ;RP REGISTER STORAGE
000170          $RPWC   = $RPCS1+2
000172          $RPBA   = $RPCS1+4
000174          $RPDA   = $RPCS1+6
000176          $RPCS2  = $RPCS1+10
000200          $RPDS   = $RPCS1+12
000202          $RPER1  = $RPCS1+14
000204          $RPAS   = $RPCS1+16
000206          $RPLA   = $RPCS1+20
000210          $RPDB   = $RPCS1+22
000212          $RPMR1  = $RPCS1+24
000214          $RPDT   = $RPCS1+26
000216          $RPSN   = $RPCS1+30
000220          $RPOF   = $RPCS1+32
000222          $RPDC   = $RPCS1+34
000224          $RPCC   = $RPCS1+36
000226          $RPER2  = $RPCS1+40
000230          $RPER3  = $RPCS1+42
000232          $RPEC1  = $RPCS1+44
000234          $RPEC2  = $RPCS1+46
000236          $RPBAE  = $RPCS1+50
000240          $RPCS3  = $RPCS1+52
```

DEVICE PARAMETER BLOCKS

```
                        ;DPB FOR DRIVE 0
046514      000   000   DRIVE0: .BYTE   0,0              ;DRIVE NUMBER 0
046516                        .BLKW   5
046530  046502                .WORD   .+10$PCS1_10$REG
046532                        .BLKB   10$PCS3_10$REG

                        ;DPB FOR DRIVE 1
046556      001   000   DRIVE1: .BYTE   1,0              ;DRIVE NUMBER 1
046560                        .BLKW   5
046572  046744                .WORD   .+10$PCS1_10$REG
046574                        .BLKB   10$PCS3_10$REG

                        ;DPB FOR DRIVE 2
047020      002   000   DRIVE2: .BYTE   2,0              ;DRIVE NUMBER 2
047022                        .BLKW   5
047034  047206                .WORD   .+10$PCS1_10$REG
047036                        .BLKB   10$PCS3_10$REG

                        ;DPB FOR DRIVE 3
047262      003   000   DRIVE3: .BYTE   3,0              ;DRIVE NUMBER 3
047264                        .BLKW   5
047276  047450                .WORD   .+10$PCS1_10$REG
047300                        .BLKB   10$PCS3_10$REG

                        ;DPB FOR DRIVE 4
047524      004   000   DRIVE4: .BYTE   4,0              ;DRIVE NUMBER 4
047526                        .BLKW   5
047540  047712                .WORD   .+10$PCS1_10$REG
047542                        .BLKB   10$PCS3_10$REG

                        ;DPB FOR DRIVE 5
047766      005   000   DRIVE5: .BYTE   5,0              ;DRIVE NUMBER 5
047770                        .BLKW   5
050002  050154                .WORD   .+10$PCS1_10$REG
050004                        .BLKB   10$PCS3_10$REG

                        ;DPB FOR DRIVE 6
050230      006   000   DRIVE6: .BYTE   6,0              ;DRIVE NUMBER 6
050232                        .BLKW   5
050244  050416                .WORD   .+10$PCS1_10$REG
050246                        .BLKB   10$PCS3_10$REG

                        ;DPB FOR DRIVE 7
050472      007   000   DRIVE7: .BYTE   7,0              ;DRIVE NUMBER 7
050474                        .BLKW   5
050506  050660                .WORD   .+10$PCS1_10$REG
050510                        .BLKB   10$PCS3_10$REG
```

K2

DEVICE PARAMETER BLOCKS

```
                                ;GENERAL PURPOSE DEVICE PARAMETER BLOCK
                                ;
050734  000          GENDPB: .BYTE   0               ;DRIVER PARAMETER BLOCK. DRIVE 0
050735  000                  .BYTE   0               ;OFFSET VALUE OR FMT16. HCI OR ECI
050736  000                  .BYTE   C               ;COMMAND CODE
050737  000                  .BYTE   0               ;PSEL. A16 AND A17
050740  177776               .WORD   -2              ;WORD COUNT (NEG)
050742  063364               .WORD   CYLNDR          ;BUFFER ADDRESS
050744  000                  .BYTE   0               ;SECTOR ADDRESS
050745  000                  .BYTE   0               ;TRACK ADDRESS
050746  000000               .WORD   0               ;CYLINDER ADDRESS
050750  050754               .WORD   GENREG          ;ADDRESS TO SAVE ALL RH11/RP07 REG S
050752  000000               .WORD   0               ;STATUS WORD
                                ;
050754               GENREG: .BLKW   24              ;REGISTER STORAGE
```

ERROR MESSAGES

```
                                .SBTTL  ERROR MESSAGES
                        ;
051024  122     110     040     EM1:    .ASCIZ  /RH CONTROLLER INTERRUPT OCCURRED (RPAS= 0)/
051077  125     116     105     EM2:    .ASCIZ  /UNEXPECTED ATTENTION OCCURRED/
051135  115     101     123     EM3:    .ASCIZ  /MASSBUS PARITY ERROR (MCPE=1)/
051173  115     101     123     EM4:    .ASCIZ  /MASSBUS PARITY ERROR (PAR=1)/
051230  101     104     104     EM5:    .ASCIZ  /ADDRESS PLUG CHANGE BIT SET/
051264  122     110     040     EM6:    .ASCIZ  /RH CONTROLLER DIDN'T RESPOND TO ADDRESSING/
051337  125     116     103     EM10:   .ASCIZ  /UNCORRECTABLE MASSBUS PARITY ERROR/
051402  106     101     124     EM11:   .ASCIZ  /FATAL MASSBUS PARITY ERROR/
051435  120     105     122     EM12:   .ASCIZ  /PERSISTENT DEVICE UNSAFE/
051466  117     120     105     EM13:   .ASCIZ  /OPERATION NOT COMPLETED WITHIN TIME LIMIT/
051540  104     122     111     EM14:   .ASCIZ  /DRIVE WENT OFFLINE/
051563  116     117     040     EM15:   .ASCIZ  /NO RESPONSE TO PORT REQUEST/
051617  110     105     101     EM20:   .ASCIZ  /HEADER CRC ERROR/
051640  104     101     124     EM21:   .ASCIZ  /DATA CHECK ('DCK') ERROR/
051671  127     122     111     EM22:   .ASCIZ  /WRITE CHECK ERROR - DATA CHECK ('DCK') SET/
051744  127     122     111     EM23:   .ASCIZ  /WRITE CHECK ERROR - DATA CHECK ('DCK') NOT SET/
052023  110     105     101     EM24:   .ASCIZ  /HEADER READ ERROR - 'FMT' BIT DROPPED/
052071  110     105     101     EM25:   .ASCIZ  /HEADER READ ERROR - HEADER COMPARE ('HCE') ERROR/
052152  106     117     122     EM26:   .ASCIZ  /FORMAT ERROR ('FER')/
052177  110     105     101     EM27:   .ASCIZ  /HEADER COMPARE ('HCE') ERROR/
052234  115     111     123     EM30:   .ASCIZ  /MISCELLANEOUS DRIVE ERROR/
052266  117     120     105     EM31:   .ASCIZ  /OPERATION INCOMPLETE ('OPI') ERROR/
052331  104     122     111     EM32:   .ASCIZ  /DRIVE TIMING ('DTE') ERROR/
052364  120     101     122     EM33:   .ASCIZ  /PARITY ('PAR') ERROR AFTER OPERATION STARTED/
052441  127     122     111     EM34:   .ASCIZ  /WRITE CLOCK FAILURE ('WCF') ERROR/
052503  111     116     126     EM35:   .ASCIZ  /INVALID ADDRESS ('IAE') ERROR/
052541  127     122     111     EM36:   .ASCIZ  /WRITE LOCK ('WLE') ERROR/
052572  104     101     124     EM37:   .ASCIZ  /DATA CHECK ('DCK') SET DURING WRITE CHECK/
052644  122     110     130     EM40:   .ASCIZ  /RHXX OR UNIBUS TRANSFER ERROR/
052702  102     125     123     EM41:   .ASCIZ  /BUS ADDRESS OR WORD COUNT INCORRECT/
052746  104     101     124     EM42:   .ASCIZ  /DATA COMPARE ERRORS - NO OTHER ERROR(S) DETECTED/
053027  103     101     116     EM43:   .ASCIZ  /CAN'T MATCH DATA READ WITH A PATTERN - UNKNOWN DATA PATTERN/
053123  105     122     122     EM44:   .ASCIZ  /ERROR BIT(S) SET, BUT NO ERROR SIGNALED BY THE RH CONTROLLER/
053220  105     103     103     EM45:   .ASCIZ  /ECC LOGIC FAILURE - POSITION REGISTER VALUE NOT VALID/
053306  102     125     123     EM46:   .ASCIZ  /BUS ADDRESS AND WORD COUNT NOT CONSISTENT/
053360  105     103     103     EM47:   .ASCIZ  /ECC LOGIC FAILURE - PATTERN REGISTER IS ZERO/
053435  123     105     105     EM50:   .ASCIZ  /SEEK INCOMPLETE ('SKI') ERROR/
053473  120     122     117     EM51:   .ASCIZ  /PROGRAM DETECTED POSITIONING ERROR/
053536  105     103     110     EM52:   .ASCIZ  /ECH ERROR - UNCORRECTABLE ECC ERROR/
053602  104     122     111     EM60:   .ASCIZ  /DRIVE UNSAFE ERROR/
053625  105     101     122     EM70:   .ASCIZ  /EARLY WARNING, TEMPERATURE WARNING (TPE) ERROR/
053704  105     101     122     EM71:   .ASCIZ  /EARLY WARNING, AIR SYSTEM WARNING (AIR) ERROR/
053762  105     101     122     EM72:   .ASCIZ  /EARLY WARNING ERROR, AIR, TPE, NOT SET/
```

M2

ERROR MESSAGES

```
      1 054031     122     120     101  DM1:    .ASCIZ  /RPAS/
      2 054036     104     122     111  DM2:    .ASCIZ  /DRIVE   RPDS    RPER1   RPER3   RPER2   RPAS/
      3 054113     104     122     111  DM3:    .ASCIZ  /DRIVE   REG ADR  DATA/
      4 054141     104     122     111  DM4:    .ASCIZ  /DRIVE   REG ADR    GOOD    BAD/
      5 054200     122     120     101  DM6:    .ASCIZ  /RPADR/
      6 054206     104     122     111  DM14:   .ASCIZ  /DRIVE   RPCS1   RPCS2   RPDS    RPER1   RPER2   RPER3   RPEC1/<CRLF>
      7 054305     122     120     105  DM15:   .ASCIZ  /RPEC2   RPWC    RPBA    RPDA    RPAS    RPLA    RPDB    RPMR1/<CRLF>
      8 054404     122     120     104  DM16:   .ASCIZ  /RPDT    RPSN    RPOF    RPDC    RPCC    STATUS/<CRLF>
      9 054464     122     120     102  DM17:   .ASCIZ  /RPBAE   RPCS3/<CRLF>
     10                                .EVEN
```

ERROR MESSAGES

```
      1 054504  001316  000000          DT1:    .WORD   ATTN,0
      2 054510  001220  000000          DT2:    .WORD   DRIVE,0
      3 054514  001220  000000          DT3:    .WORD   DRIVE,0
      4 054520  001220  000000          DT4:    .WORD   DRIVE,0
      5 054524  001272  000000          DT6:    .WORD   $RPADR,0
      6
      7 054530  000166  000176  000200  DT14:   .WORD   $RPCS1,$RPCS2,$RPDS,$RPER1,$RPER2,$RPER3,$RPEC1,0
      8 054550  000234  000170  000172  DT15:   .WORD   $RPEC2,$RPWC,$RPBA,$RPDA,$RPAS,$RPLA,$RPDB,$RPMR1,0
      9 054572  000214  000216  000220  DT16:   .WORD   $RPDT,$RPSN,$RPOF,$RPDC,$RPCC,$TATUS,0
     10 054610  000236  000240  000000  DT17:   .WORD   $RPBAE,$RPCS3,0
```

```
                                        SBTTL   ASCII MESSAGES

 054616  120  122  123  LIN2C:  .ASCIZ  /PRSNT COMMAND= /
 054636  040  040  120  LIN2P:  .ASCIZ  /  PREVS COMAND= /
 054657  052  040  105  LIN2S:  .ASCIZ  @= ERROR AT BAD TRACK/SECTOR@
 054713  105  122  122  LINM3:  .ASCIZ  /ERROR AT C/
 054726  040  124  000  T:      .ASCIZ  / T/
 054731  120  122  123  LINN3:  .ASCIZ  /PRSNT ADDR= C/
 054747  040  123  000  S:      .ASCIZ  / S/
 054752  040  040  120  LINP3:  .ASCIZ  /  PREVS ADDR= C/
 054772  123  124  101  LINS3:  .ASCIZ  /START CYL= /
 055006  040  040  105  LINEN3: .ASCIZ  /  END CYL= /
 055022  040  040  111  LINA3:  .ASCIZ  /  ACTUAL CYL= /
 055041  040  040  124  LINT3:  .ASCIZ  /  TRK= /
 055051  040  122  120  LINCA3: .ASCIZ  / RPDC= /
 055061  122  120  104  LINDA3: .ASCIZ  /RPDA= /
 055070  122  120  102  LINB3:  .ASCIZ  /RPBA= /
 055077  040  040  122  LINW3:  .ASCIZ  /  RPWC= /
 055110  123  124  101  LINST3: .ASCIZ  /START TRK= /
 055124  123  124  101  LINSS3: .ASCIZ  /START SEC= /
 055140  102  125  106  LINM4:  .ASCIZ  /BUFFER ADDR= /
 055156  040  040  127  LINS4:  .ASCIZ  /  WRD CNT= /
 055172  040  040  116  LINX4:  .ASCIZ  /  NMBR WRDS XFRD= /
 055215  105  130  120  LIND5:  .ASCIZ  /EXPCTD DATA= /
 055233  040  040  122  LINB5:  .ASCIZ  /  RECEVD DATA= /
 055253  040  040  127  LINP5:  .ASCIZ  /  WORD POS= /
 055270  110  105  101  LINS5:  .ASCIZ  /HEADER FROM ERROR SECTOR= /
 055323  122  120  105  LINEP5: .ASCIZ  /RPEC1= /
 055333  040  040  122  LINE05: .ASCIZ  /  RPEC2= /
 055345  123  105  103  LINB6:  .ASCIZ  /SECTOR IS ECC CORRECTABLE /
 055400  123  105  103  LINC6:  .ASCIZ  /SECTOR READ CORRECTLY AFTER /
 055435  103  117  122  LING6:  .ASCIZ  /CORRECTED ON /
 055453  040  122  105  LINR6:  .ASCIZ  / RETRY(S)/
 055465  125  116  103  LINUO6: .ASCIZ  /UNCORRECTABLE AFTER /
 055512  040  040  115  LIN7M:  .ASCIZ  /  MIS POS ERRORS= /
 055535  124  117  124  LIN7P:  .ASCIZ  /TOTALS; SEEKS= /
 055555  040  040  123  LIN7S:  .ASCIZ  /  SKI ERRORS= /
 055574  124  117  124  LIN7T:  .ASCIZ  /TOTALS; ERRORS= /
 055615  040  127  122  LIN7X:  .ASCIZ  / WRDS WRITN= /
 055633  040  127  122  LIN7R:  .ASCIZ  / WRDS READ= /

 055650  105  122  122  LIN8M:  .ASCIZ  /ERROR DURING RETRY/
 055673  104  101  124  LIN9B:  .ASCIZ  /DATA COMPARISON ERRORS/
 055722  040  040  040  LIN9M:  .ASCII  /          WORD  EXPCTD  RECEVD/<CRLF>
 055760  101  104  104          .ASCIZ  /ADDR     POS   DATA    DATA/<CRLF>
 056015  040  040  040  LIN9I:  .ASCII  /          WORD  RECEVD/<CRLF>
 056043  101  104  104          .ASCIZ  /ADDR     POS   DATA/<CRLF>
 056070  124  117  124  LIN9E:  .ASCIZ  /TOTAL COMPARE ERRORS= /
 056117  124  110  105  LIN9G:  .ASCIZ  /THE DATA COMPARED OK/<CRLF>
 056145  105  122  122  LIN10A: .ASCIZ  /ERROR BURST BEGINS AT WORD /
 056201  040  111  116  LIN10B: .ASCIZ  / IN DATA FIELD OF ERROR SECTOR/<CRLF>
 056241  105  122  122  LIN10C: .ASCII  /ERROR WAS NOT IN THE DATA READ - /<CRLF>
 056303  105  103  103          .ASCIZ  /ECC CORRECTION CAN'T BE PERFORMED/
 056345  105  103  103  LIN10H: .ASCII  /ECC CORRECTION RESULTS/<CRLF>
 056374  040  040  040          .ASCII  /          WORD  BAD     CORRECTED /<CRLF>
 056436  101  104  104          .ASCIZ  /ADDR     POS   DATA    DATA/<CRLF>
 056473  103  117  116  LIN11H: .ASCIZ  /CONTENTS OF ERROR SECTOR (REPORTED ABOVE)/<CRLF>
```

ASCII MESSAGES

```
     62 056546      101      104      104  LIN11:   .ASCIZ  /ADDR      HEADER/<CRLF>
     63 056567      101      104      104  LIN11A:  .ASCIZ  /ADDR      DATA/<CRLF>
     64 056606      040                   BLNKS4:  .ASCII  / /
     65 056607      040                   BLNKS3:  .ASCII  / /
     66 056610      040                   BLNKS2:  .ASCII  / /
     67 056611      040      000          BLNKS1:  .ASCIZ  / /
     68 056613      122      105      124  LINX5:.ASCIZ     /RETRIEVED BY A RDMD COMMAND/
```

```
                                .SBTTL  TELETYPE MESSAGES

056647  077     000             QUES:   .ASCIZ  /?/
056651  075     000             EQUAL:  .ASCIZ  /=/
056653  054     000             COMMA:  .ASCIZ  /,/
056655  055     000             DASH:   .ASCIZ  / /
056657  011     000             TAB:    .ASCIZ  <11>
056661  104     122     111     DRVMSG: .ASCIZ  /DRIVE/
056667  040     117     106     UNTOFF: .ASCIZ  / OFFLINE/
056700  040     117     116     UNTON:  .ASCIZ  / ONLINE/
056710  040     116     117     UNTNOT: .ASCIZ  ' NOT BEING TESTED/
056732  040     101     114     UNTASN: .ASCIZ  / ALREADY BEING TESTED/
056760  040     116     117     NOTRP:  .ASCIZ  / NOT AN RP07/
056775  040     116     117     NOTPRS: .ASCIZ  / NOT PRESENT/
057012  040     116     117     NOTAVL: .ASCIZ  / NOT AVAILABLE/
057031  040     125     116     NOTSAF: .ASCIZ  / UNSAFE/
057041  040     114     117     LODEV:  .ASCIZ  / LOAD DEVICE/
057056  040     116     117     NINLEV: .ASCIZ  / NON-INTERLEAVED/
057077  200     116     117     NODFLT: .ASCIZ  <CRLF>/NO DEFAULT/
057113  200     104     122     DRSTAT: .ASCIZ  <CRLF>/DRIVE STATUS:/
057132  120     107     000     MSGPG:  .ASCIZ  /PG/
057135  040     040     040     MSHW1:  .ASCIZ  /   HW1 /
057145  040     040     040     MSHW2:  .ASCIZ  /   HW2 /
057155  122     120     060     $RP07:  .ASCIZ  /RP07/
057162  122     120     060     $RP07P: .ASCIZ  /RP07/
057167  200     012             MSGREP: .ASCII  <CRLF><LF>
057171  057     134     057             .ASCIZ  @/\/\/\/\/\/\/\/\/\/\/-PERFORMANCE REPORT-\/\/\/\/\/\/\/\/\/\/\/\@
057274  200             TRMREP: .ASCII  <CRLF>
057275  134     057     134             .ASCII  @\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\@
057377  200     000             .ASCIZ  <CRLF>
057401  055     055     055     DASH5:  .ASCIZ  /-----/
057407  055     055     055     DASH13: .ASCIZ  /------------ /
057425  123     125     115     MSGSUM: .ASCIZ  /SUMMARY. /
057437  104     122     117     MSGDRP: .ASCIZ  /DROPPED. /
057451  105     040     116     MSGEOP: .ASCIZ  /E N D   O F   P A S S/
057477  105     040     116     MSGEOT: .ASCIZ  /E N D   O F   T E S T   -          - -      - ---    ------EOT/
057573  200     103     117     MSGCON: .ASCIZ  <CRLF>/CONTINUING.../
057612  057     040     120     MSPASS: .ASCIZ  @/ PASS @
057622  040     124     117     MSTOTL: .ASCIZ  / TOTAL /
057632  040     130     061     MSGX10: .ASCIZ  / X10^6/
057641  007     200     077     DROPMG: .ASCIZ  <BELL><CRLF>/?FATAL OR EXCESSIVE ERRORS/
057676  040     117     116     MSGON:  .ASCIZ  / ON /
057703  040     123     124     ASGND:  .ASCIZ  / STARTED/
057714  200     007     077     NEDCLK: .ASCIZ  <CRLF><BELL>/? 'L' OR 'P' CLOCK REQUIRED ON SYSTEM/<CRLF>
057765  116     000             N:      .ASCIZ  /N/
057767  131     000             Y:      .ASCIZ  /Y/
057771  056     000             PERIOD: .ASCIZ  /./
057773  040     077     103     MSWRO:  .ASCIZ  / ?CAN'T WRITE IN 'READ ONLY' MODE/<CRLF>
060036  040     077     111     INVLD:  .ASCIZ  / ?INVALID COMMAND/<CRLF>
060061  200     105     116     ENTCOM: .ASCIZ  <CRLF>/ENTER COMMAND: /
060102  040     101     104     ENTLMT: .ASCIZ  / ADDRESS LIMITS:/<CRLF>
060124  200     105     116     ENTADR: .ASCIZ  <CRLF>/ENTER BAD SECTR ADRS:/<CRLF>
060154  072     000             COLON:  .ASCIZ  /:/
060156  116     117     116     NONE:   .ASCIZ  /NONE/
060163  040     077     111     BADENT: .ASCIZ  / ?INVALID ENTRY/<CRLF>
060204  200     012     116     NODRVS: .ASCIZ  <CRLF><LF>/NO DRIVES ASSIGNED/<CRLF>
060232  104     122     126     DRVSN:  .ASCIZ  @DRV S/N: @
```

```
     62 060244      200      103      110  MSPRM:  .ASCIZ  <CRLF>/CHANGE DRIVE PARAMETERS (L) N ? /
     63 060306      200      104      117  MESFE:  .ASCIZ  <CRLF>/DO YOU WANT TO WRITE ANYWHERE ON MEDIA (L) N ? /
     64 060367      200               OVRWRT: .ASCII  <CRLF>
     65 060370      007      011      041          .ASCII  <BELL>/ ! CUSTOMER DATA WILL BE OVERWRITTEN !/<CRLF>
     66 060440      007      011      055          .ASCII  <BELL>/ --  --  ----  ---  --  /<CRLF>
     67 060510      103      117      116          .ASCIZ  /CONTINUE (L)  ? /
     68 060531      200      052      040  FEONLY: .ASCIZ  <CRLF>/* TESTING FE CYLINDER ONLY */<CRLF>
     69 060570      200      052      040  MREAD:  .ASCIZ  <CRLF>/* PROGRAM IS LOCKED IN 'READ ONLY' MODE */<CRLF>
     70
                                        .EVEN
```

```
                                .SBTTL  PARAMETER ENTRY TABLE
                        ;
060644  061002  031000  001466  PARLST: .WORD   PAR1,12800.,,WRDCNT
060652  061041  077777  001470          .WORD   PAR2,32767.,,INTRVL
060660  061331  077777  001462          .WORD   PAR12,32767.,,CMPTIM
060666  061555  077777  001474          .WORD   PAR19,32767.,,PASSES
060674  061121  000017  001476          .WORD   PAR3,15.,,PATTERN
060702  061264  000001  001500          .WORD   PAR11,1,,RANDWC
060710  061414  000007  001502          .WORD   PAR14,7,,RATIO
060716  061511  000001  001504          .WORD   PAR16,1,,ENDING
060724  061447  000001  001506          .WORD   PAR15,1,,WRTCHK
060732  061577  000001  001510          .WORD   PAR21,1,,RANDOM
060740  000000                          .WORD   0                       ;TABLE TERMINATOR
                        ;
060742      040     057     040 SLASH:  .ASCIZ  / / /
060746      200     103     110 ASKPAR: .ASCIZ  <CRLF>/CHANGE PARAMETERS (L) N ? /
061002      115     101     130 PAR1:   .ASCIZ  /MAXIMUM WORD COUNT (6-12800.) /
061041      124     111     115 PAR2:   .ASCIZ  /TIME BETWEEN REPORTS (IN MINUTES, 0=NO REPORT) /
061121      104     101     124 PAR3:   .ASCIZ  /DATA PATTERN NUMBER (0=RANDOM, 1-15.=FIXED) /
061176      115     101     130 PAR4:   .ASCIZ  /MAX CYL /
061207      115     111     116 PAR5:   .ASCIZ  /MIN CYL /
061220      115     101     130 PAR6:   .ASCIZ  /MAX TRK /
061231      115     111     116 PAR7:   .ASCIZ  /MIN TRK /
061242      115     101     130 PAR8:   .ASCIZ  /MAX SEC /
061253      115     111     116 PAR9:   .ASCIZ  /MIN SEC /
061264      127     117     122 PAR11:  .ASCIZ  /WORD COUNT MODE (0=RANDOM, 1=FIXED) /
061331      124     111     115 PAR12:  .ASCIZ  /TIME BETWEEN DATA COMPARES (IN MINUTES, 0=ALWAYS) /
061414      122     105     101 PAR14:  .ASCIZ  /READ TO WRITE RATIO (0-7) /
061447      105     116     101 PAR15:  .ASCIZ  /ENABLE WRITE CHECK (0=NO, 1=YES) /
061511      105     116     104 PAR16:  .ASCIZ  /END OF PASS MODE (0=SEEKS, 1=DATA) /
061555      116     125     115 PAR19:  .ASCIZ  /NUMBER OF PASSES /
061577      123     105     105 PAR21:  .ASCIZ  /SEEK MODE (0=RANDOM, 1=SEQUENTIAL) /
                        ;
                                .EVEN
```

DRIVE PARAMETER TABLES (DPB)

```
                                        .SBTTL  DRIVE PARAMETER TABLES (DPB)
                                        ;
                                        ;PARAMETER TABLE POINTERS FOR ADDRESS LIMITS
                                        ;
061644  061664                  TABLE:  .WORD   TABLE0          ;PARAMETER TABLE FOR DRIVE 0
061646  061732                          .WORD   TABLE1          ;PARAMETER TABLE FOR DRIVE 1
061650  062000                          .WORD   TABLE2          ;PARAMETER TABLE FOR DRIVE 2
061652  062046                          .WORD   TABLE3          ;PARAMETER TABLE FOR DRIVE 3
061654  062114                          .WORD   TABLE4          ;PARAMETER TABLE FOR DRIVE 4
061656  062162                          .WORD   TABLE5          ;PARAMETER TABLE FOR DRIVE 5
061660  062230                          .WORD   TABLE6          ;PARAMETER TABLE FOR DRIVE 6
061662  062276                          .WORD   TABLE7          ;PARAMETER TABLE FOR DRIVE 7
                                        ;
                                        ;PARAMETER TABLE FOR ADDRESS LIMITS
                                        ;
061664  061207  000000  046452  TABLE0: .WORD   PAR5.0,MINCYL,DRIVE0
061672  061176  001166  046450          .WORD   PAR4.630.,MAXCYL,DRIVE0
061700  061231  000035  046456          .WORD   PAR7.29.,MINTRK,DRIVE0
061706  061220  000035  046454          .WORD   PAR6.29.,MAXTRK,DRIVE0
061714  061253  000041  046462          .WORD   PAR9.33.,MINSEC,DRIVE0
061722  061242  000041  046460          .WORD   PAR8.33.,MAXSEC,DRIVE0.0

061732  061207  000000  046714  TABLE1: .WORD   PAR5.0,MINCYL,DRIVE1
061740  061176  001166  046712          .WORD   PAR4.630.,MAXCYL,DRIVE1
061746  061231  000035  046720          .WORD   PAR7.29.,MINTRK,DRIVE1
061754  061220  000035  046716          .WORD   PAR6.29.,MAXTRK,DRIVE1
061762  061253  000041  046724          .WORD   PAR9.33.,MINSEC,DRIVE1
061770  061242  000041  046722          .WORD   PAR8.33.,MAXSEC,DRIVE1.0

062000  061207  000000  047156  TABLE2: .WORD   PAR5.0,MINCYL,DRIVE2
062006  061176  001166  047154          .WORD   PAR4.630.,MAXCYL,DRIVE2
062014  061231  000035  047162          .WORD   PAR7.29.,MINTRK,DRIVE2
062022  061220  000035  047160          .WORD   PAR6.29.,MAXTRK,DRIVE2
062030  061253  000041  047166          .WORD   PAR9.33.,MINSEC,DRIVE2
062036  061242  000041  047164          .WORD   PAR8.33.,MAXSEC,DRIVE2.0

062046  061207  000000  047420  TABLE3: .WORD   PAR5.0,MINCYL,DRIVE3
062054  061176  001166  047416          .WORD   PAR4.630.,MAXCYL,DRIVE3
062062  061231  000035  047424          .WORD   PAR7.29.,MINTRK,DRIVE3
062070  061220  000035  047422          .WORD   PAR6.29.,MAXTRK,DRIVE3
062076  061253  000041  047430          .WORD   PAR9.33.,MINSEC,DRIVE3
062104  061242  000041  047426          .WORD   PAR8.33.,MAXSEC,DRIVE3.0

062114  061207  000000  047662  TABLE4: .WORD   PAR5.0,MINCYL,DRIVE4
062122  061176  001166  047660          .WORD   PAR4.630.,MAXCYL,DRIVE4
062130  061231  000035  047666          .WORD   PAR7.29.,MINTRK,DRIVE4
062136  061220  000035  047664          .WORD   PAR6.29.,MAXTRK,DRIVE4
062144  061253  000041  047672          .WORD   PAR9.33.,MINSEC,DRIVE4
062152  061242  000041  047670          .WORD   PAR8.33.,MAXSEC,DRIVE4.0

062162  061207  000000  050124  TABLE5: .WORD   PAR5.0,MINCYL,DRIVE5
062170  061176  001166  050122          .WORD   PAR4.630.,MAXCYL,DRIVE5
062176  061231  000035  050130          .WORD   PAR7.29.,MINTRK,DRIVE5
062204  061220  000035  050126          .WORD   PAR6.29.,MAXTRK,DRIVE5
062212  061253  000041  050134          .WORD   PAR9.33.,MINSEC,DRIVE5
062220  061242  000041  050132          .WORD   PAR8.33.,MAXSEC,DRIVE5.0
```

H3

DRIVE PARAMETER TABLES (DPB)

```
[illegible]  061207  000000  050566  TABLE6:  .WORD   PARM5.0,MINCYL-DRIVE6
[illegible]  061176  001166  050564          .WORD   PARM4.650.,MAXCYL-DRIVE6
[illegible]  061251  000035  050572          .WORD   PARM7.29.,MINTRK-DRIVE6
[illegible]  061240  000035  050570          .WORD   PARM6.29.,MAXTRK-DRIVE6
[illegible]  061253  000001  050576          .WORD   PARM9.33.,MINSEC-DRIVE6
[illegible]  061242  000001  050574          .WORD   PARM8.33.,MAXSEC-DRIVE6.0

050776  061207  000000  050650  TABLE7:  .WORD   PARM5.0,MINCYL-DRIVE7
[illegible]  061176  001166  050646          .WORD   PARM4.650.,MAXCYL-DRIVE7
[illegible]  061251  000035  050654          .WORD   PARM7.29.,MINTRK-DRIVE7
[illegible]  061240  000035  050652          .WORD   PARM6.29.,MAXTRK-DRIVE7
[illegible]  061253  000001  050660          .WORD   PARM9.33.,MINSEC-DRIVE7
[illegible]  061242  000001  050656          .WORD   PARM8.33.,MAXSEC-DRIVE7.0
```

```
                        .SBTTL  ROUTINE TO SIZE MEMORY

                        ;;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                        ;CALL:
                        ;       JSR     PC,@SIZE
                        ;       RETURN
                        ;@LSTAD WILL CONTAIN THE LAST AVAILABLE MEMORY LOCATION

062344  010046                SIZE:   MOV     R0,-(SP)        ;;SAVE R0 ON THE STACK
062346  010146                        MOV     R1,-(SP)        ;;SAVE R1 ON THE STACK
062350  013746  000114                MOV     @#114,-(SP)     ;;SAVE MEMORY ERROR VECTOR PS & PC
062354  013746  000116                MOV     @#116,-(SP)
062360  012737  000116  000114        MOV     #116,@#114      ;;IGNORE PARITY ERRORS WHILE SIZING
062366  012737  000002  000116        MOV     #2,@#116
062374  013746  000004                MOV     @#MEMVEC,-(SP)  ;;SAVE PRESENT ERROR VECTOR PS & PC
062400  013746  000006                MOV     @#MEMVEC+2,-(SP)
062404  010600                        MOV     SP,R0           ;;SAVE THE STACK POINTER
                        ;;SET THE [MEMVEC PS TO THE PRESENT PS
062406  104400                        TRAP                    ;;PUSH OLD PSW AND PC ON STACK
062410  012637  000006                MOV     (SP)+,@#MEMVEC+2        ;;SAVE THE PSW IN @#MEMVEC+2
062414  012737  062456  000004        MOV     #2$,@#MEMVEC    ;;SET FOR TIMEOUT
062422  012701  020000                MOV     #20000,R1       ;;FIRST ADDRESS
062426  005711                1$:     TST     (R1)            ;;TEST THIS ADDRESS
062430  005721                        TST     (R1)+           ;;STEP TO NEXT ADDRESS
062432  000775                        BR      1$              ;;TRY ANOTHER
062434  162701  000002        2$:     SUB     #2,R1           ;;DROP BACK
062440  010006                        MOV     R0,SP           ;;RESTORE THE STACK
062442  012637  000006                MOV     (SP)+,@#MEMVEC+2 ;;RESTORE ERROR VECTOR
062446  012637  000004                MOV     (SP)+,@#MEMVEC
062452  012637  000116                MOV     (SP)+,@#116     ;;RESTORE MEMORY ERROR VECTOR
062456  012637  000114                MOV     (SP)+,@#114
062462  010137  062474                MOV     R1,@LSTAD       ;;LAST ADDRESS
062466  012601                        MOV     (SP)+,R1        ;;RESTORE R1
062470  012600                        MOV     (SP)+,R0        ;;RESTORE R0
062472  000207                        RTS     PC
062474  000000        LSTAD:  .WORD   0               ;;CONTAINS THE LAST ADDRESS
```

```
                                        .SBTTL  DUSADR - GET BUS ADDRESS AND VECTOR ADDRESS
                                ;
                                ;THIS ROUTINE IS USED TO INQUIRE THE BUS ADDRESS OF THE BOOT DEVICE
                                ;IS SETUP FOR THE VECTOR ADDRESS. IT WILL ALSO READ THE ADDRESS
                                ;FROM THE TTY IF REQUIRED.
                                ;
                                ;NOTE: THIS ROUTINE DESTROYS R0-R4
                                ;CALL:
                                ;
                                ;       JSR     PC,DUSADR
                                ;       RETURN
                                ;
000001                          DUSADR: TST     DSCADR          ;INPUT FROM TTY REQUESTED?
000002                                  BGE     1$              ;NO  BRANCH
000003                                  CLR     DSCADR          ;YES--CLEAR THE REQUEST FLAG
000004                                  TYPE    ,$CRLF          ;CR LF
000005                          1$:     CLR     CFLAG           ;CLEAR CONTROL C FLAG
000006                                  MOV     #SAVADR,R0      ;FIRST ADDRESS
000007                                  TYPE    ,MSPCS1         ;"SPCS1="
000008                                  MOV     (R0),-(SP)      ;PRESENT SPCS1 ADDRESS
000009                                  TYPOC                   ;TYPE IT
000010                                  TYPE    ,.BLNKS2        ;TYPE 2 BLANKS
000011                                  RDLIN                   ;GET THE ENTRY
000012                                  MOV     (SP)+,R1        ;ADDRESS OF ASCII TEXT
000013                                  TST     CFLAG           ;WAS (^C) TYPED?
000014                                  BNE     10              ;BR IF YES
000015                                  JSR     R5,CK.NUM       ;ENTER AND STORE THE NEW ADDRESS
000016                                  BR      10              ;ERROR EXIT
                                ;
000020                          2$:     MOV     #SAVVEC,R0      ;VECTOR ADDRESS
000022                                  TYPE    ,MSPVEC         ;"SPVEC="
000024                                  MOV     (R0),-(SP)      ;PRESENT BOOT/DOT VECTOR ADDRESS ON THE STACK
000026                                  TYPOC                   ;TYPE IT
000027                                  TYPE    ,.BLNKS2        ;TYPE 2 BLANKS
000030                                  RDLIN                   ;READ THE ENTRY
000031                                  MOV     (SP)+,R1        ;ASCII TEXT ADDRESS
000032                                  TST     CFLAG           ;WAS (^C) TYPED?
000033                                  BNE     10              ;BR IF YES
000034                                  JSR     R5,CK.NUM       ;ENTER AND STORE NEW ADDRESS
000035                                  BR      2$              ;ERROR EXIT
                                ;
000037                          3$:     MOV     #SAVADR,R0      ;FIRST ADDRESS OF NEW PARAMETERS
000041                                  MOV     #SAVADR,R1      ;FIRST ADDRESS OF WHERE TO PUT THEM
000043                                  MOV     (R0)+,(R1)+     ;BUS ADDRESS
000045                                  MOV     (R0)+,(R1)+     ;VECTOR ADDRESS
000047                                  RTS     PC              ;RETURN
                                ;
000050  122  120  103           MSPCS1: .ASCIZ  <SPCS1=>
000055  122  120  126           MSPVEC: .ASCIZ  <SPVEC=>
```

```
                        ;+
                        ; CK.OCM  CHECK NUMBER (OCTAL)
                        ;THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL CHARACTERS
                        ;AND FORMS AN OCTAL NUMBER IN R2
                        ;CALL:
                        ;       MOV     #MAX,R1         ;#CHRS OF ASCII STRING
                        ;       JSR     R5,CK.OCM       ;R5 CHANGED
                        ;       RET                     ;ERROR EXIT
                        ;       RET                     ;NORMAL EXIT
                        ;-

062734  010206          CK.OCM: MOV     R2,-(SP)        ;SAVE R2
062736  010346                  MOV     R3,-(SP)        ;SAVE R3
062740  010446                  MOV     R4,-(SP)        ;SAVE R4
062742  012703  000006          MOV     #6,R3           ;MAX OCTAL DIGITS IN THE NUMBER
062746  005002                  CLR     R2              ;FINAL OCTAL VALUE
062750  112104          10$:    MOVB    (R1)+,R4        ;GET CURRENT POINTED BYTE
062752  001424                  BEQ     30              ;BRANCH IF TERMINATOR DETECTED
062754  120427  000060          CMPB    R4,#'0          ;SMALLER THAN ASCII-0
062760  103425                  BLO     50              ;YES,ERROR EXIT
062762  120427  000067          CMPB    R4,#'7          ;LARGER THAN ASCII-7 ?
062766  101022                  BHI     50              ;YES,ERROR EXIT
062770  006302                  ASL     R2              ;SHIFT LEFT
062772  103420                  BCS     50              ;
062774  006302                  ASL     R2              ;ONE
062776  103416                  BCS     50              ;
063000  006302                  ASL     R2              ;OCTAL DIGIT
063002  103414                  BCS     50              ;ERROR IF CARRY BIT SET
063004  042704  177770          BIC     #177770,R4      ;CHOP OFF HIGHER BITS
063010  060402                  ADD     R4,R2           ;APPENDING CURRENT DIGIT TO NUMBER
063012  005303                  DEC     R3              ;DECREMENT BYTE COUNT
063014  001401                  BEQ     20              ;BRANCH IF LAST BYTE
063016  000754                  BR      10              ;LOOPING BACK

063020  112104          20$:    MOVB    (R1)+,R4        ;CHECK TERMINATOR
063022  001004                  BNE     50              ;ERROR EXIT
063024  005702          30$:    TST     R2              ;FINAL VALUE= 0
063026  001401                  BEQ     40              ;YES,THEN NOT REPLACE THE ORIGINAL VALUE
063030  010210                  MOV     R2,(R0)         ;REPLACE THE ORIGINAL VALUE
063032  005725          40$:    TST     (R5)+           ;ADJUST FOR NORMAL RETURN
063034  012604          50$:    MOV     (SP)+,R4        ;RESTORE R4
063036  012603                  MOV     (SP)+,R3        ;RESTORE R3
063040  012602                  MOV     (SP)+,R2        ;RESTORE R2
063042  000205                  RTS     R5              ;EXIT

                        .EVEN
063044  000000          PATCH:  .WORD   0               ;ALLOCATE 200 OCTAL LOCATIONS FOR PATCHING

063364                  CYLNDR: .BLKW   256.            ;ONE SECTOR OF WORDS
064370                  ENDPGM=.
```

```
                                        .SBTTL  HELP TEXT MESSAGE

  064370    200                         MSHELP: .ASCII  <CRLF>
  064371    007     124     110                 .ASCII  <BELL>/THE HELP MESSAGE CAN ONLY BE TYPED ONCE./
  064442    007                                 .ASCII  <BELL>
  064443    200     104     117                 .ASCIZ  <CRLF>/DO YOU WANT IT TYPED (L) ? /
  064500    200                         HELPTX: .ASCII  <CRLF>
  064501    200     104     122                 .ASCII  <CRLF>/DRIVE PARAMETERS:/
  064523    200     115     101                 .ASCII  <CRLF>/MAXCYL    :max cylinder adrs/
  064557    200     115     111                 .ASCII  <CRLF>/MINCYL    :min cylinder adrs/
  064613    200     115     101                 .ASCII  <CRLF>/MAXTRK    :max track adrs/
  064644    200     115     111                 .ASCII  <CRLF>/MINTRK    :min track adrs/
  064675    200     115     101                 .ASCII  <CRLF>/MAXSEC    :max sector adrs/
  064727    200     115     111                 .ASCII  <CRLF>/MINSEC    :min sector adrs/
  064761    200                                 .ASCII  <CRLF>
  064762    200     124     110                 .ASCII  <CRLF>/THE FOLLOWING ARE VALID COMMANDS /
  065024    200     040     127                 .ASCII  <CRLF>/ W#<CR> assign drive and do SEQUENTIAL WRITE data/
  065106    200     040     122                 .ASCII  <CRLF>/ R#<CR> assign drive and do SEQUENTIAL READ data/
  065167    200     040     124                 .ASCII  <CRLF>/ T#<CR> assign drive  to do (random or sequential) TEST/
  065257    200     127     124                 .ASCII  <CRLF>/WT#<CR> assign drive, to do SEQUENTIAL WRITE data and/
  065345    200     040     040                 .ASCII  <CRLF>/           to do (random or sequential) TEST/
  065417    200     040     104                 .ASCII  <CRLF>/ D#<CR> DROP a drive from test/
  065456    200     040     123                 .ASCII  <CRLF>/ S#<CR> type the performance SUMMARY report/
  065532    200                                 .ASCII  <CRLF>
  065533    200     040     040                 .ASCII  <CRLF>/  # is the drive number (0-7 or  A  for all drives)/
  065617    200                                 .ASCII  <CRLF>
  065620    200     123     127                 .ASCII  <CRLF>/SWITCH REGISTER SETTINGS:/
  065652    200     123     127                 .ASCII  <CRLF>/SW15= (100000)  halt on error/
  065710    200     123     127                 .ASCII  <CRLF>/SW14= (040000)  /
  065731    200     123     127                 .ASCII  <CRLF>/SW13= (020000)  inhibit error typeout/
  065777    200     123     127                 .ASCII  <CRLF>/SW12= (010000)  /
  066020    200     123     127                 .ASCII  <CRLF>/SW11= (004000)  /
  066041    200     123     127                 .ASCII  <CRLF>/SW10= (002000)  ring tty bell on error/
  066110    200     123     127                 .ASCII  <CRLF>@SW09= (001000)  change end of pass to 1/4 of normal@
  066174    200     123     127                 .ASCII  <CRLF>/SW08= (000400)  inhibit end of pass messages/
  066251    200     123     127                 .ASCII  <CRLF>/SW07= (000200)  display all data compare errors/
  066331    200     123     127                 .ASCII  <CRLF>/SW06= (000100)  don't change parameters (loop on present values)/
  066432    200     123     127                 .ASCII  <CRLF>/SW05= (000040)  A. partial register display if error/
  066517    200     040     040                 .ASCII  <CRLF>/                B. no ECC correction results displayed if error/
  066617    200     123     127                 .ASCII  <CRLF>/SW04= (000020)  A. do not check for maximum error count/
  066707    200     040     040                 .ASCII  <CRLF>/                B. do not drop drive at end of test/
  066773    200     123     127                 .ASCII  <CRLF>/SW03= (000010)  A. display error sector if DCK, DTE or WCF error/
  067074    200     040     040                 .ASCII  <CRLF>/                B. display sector if DCK uncorrectable after 28th ret
ry/
  067204    200     040     040                 .ASCII  <CRLF>/                C. if data compare error & SW07 set, display remainde
r of bu
  067325    200     123     127                 .ASCII  <CRLF>/SW02= (000004)  A. do not type drive status at program start/
  067422    200     040     040                 .ASCII  <CRLF>/                B. do not type performance report after specified tim
e/
  067531    200     040     040                 .ASCII  <CRLF>/                C. prompt 'write anywhere' question during auto test
mode/
  067643    200     123     127                 .ASCII  <CRLF>@SW01= (000002)  inhibit data compare after read without DCK errors@
  067745    200     123     127                 .ASCII  <CRLF>/SW00= (000001)  read only mode/
```

M3

HELP TEXT MESSAGE

```
      1 070004      200                              .ASCII  <CRLF>
      2 070005      200      116      117            .ASCII  <CRLF>/NOTE: If a DCK error occurs,  the program  will execute/
      3 070075      200      040      040            .ASCII  <CRLF>/       a data compare on the data in memory, regardless/
      4 070165      200      040      040            .ASCII  <CRLF>/       of the setting in SW01./
      5 070223      200                              .ASCII  <CRLF>
      6 070224      200      124      171            .ASCII  <CRLF>/Type control-c (^C) to HALT the program/
      7 070274      200                              .ASCII  <CRLF>
      8 070275      200      124      157            .ASCII  <CRLF>/To change RHXX adresses,  start program at address 204/
      9 070364      200                              .ASCII  <CRLF>
     10 070365      200      056      105            .ASCII  <CRLF>/.END OF HELP/
     11 070402      200                              .ASCII  <CRLF>
     12 070403      200      000                     .ASCIZ  <CRLF>
     13
     15           000200                    .END     200
```

N3

SYMBOL TABLE

```
ABASE = 176700          ASGN7   027624          BIT5  = 000040          CMPTIM  001462          DPR   = 000400
ABNRML  031452          ASGN8   027666          BIT6  = 000100          CMSEC   001372          DPRQS   040564
ACDW1 = 000000          ASKPAR  060746          BIT7  = 000200          CMSTR   014276          DRIVE = 001220
ACDW2 = 000000          ASNERR  031332          BIT8  = 000400          CMSTR2  014436          DRIVE0  046314
ACPUOP= 000000          ASNLST  001542          BIT9  = 001000          CMTRK   001373          DRIVE1  046556
ACTDRV  040600          ASNMSG  031356          BLKADR  002056          COLON   060154          DRIVE2  047020
ACTSTR  040601          ASSIGN  027202          BLNKS1  056611          COMMA   056653          DRIVE3  047262
ADDW0 = 000000          ASWREG= 000000          BLNKS2  056610          COMTBL  002076          DRIVE4  047524
ADDW1 = 000000          ATA   = 100000          BLNKS3  056607          CPSAVE  035406          DRIVE5  047766
ADDW10= 000000          ATABIT  040630          BLNKS4  056606          CR    = 000015          DRIVE6  050230
ADDW11= 000000          ATESTN= 000000          BPE   = 000020          CRLF  = 000200          DRIVE7  050472
ADDW12= 000000          ATTN    001316          BPTVEC= 000014          CTRAP   034756          DROP    031362
ADDW13= 000000          AT0   = 000001          BSE   = 100000          CYLLMT= 000150          DROPD   027700
ADDW14= 000000          AT1   = 000002          BUFTBL  001654          CYLNDR  063364          DROPNG  057641
ADDW15= 000000          AT2   = 000004          BUSADR  062476          DASH    056655          DRQ   = 004000
ADDW2 = 000000          AT3   = 000010          CFLAG   001334          DASH13  057407          DRSTAT  057113
ADDW3 = 000000          AT4   = 000020          CHGADR  001332          DASH5   057401          DRVACT  040524
ADDW4 = 000000          AT5   = 000040          CHKWC   021572          DATAPK  030132          DRVCLR= 000111
ADDW5 = 000000          AT6   = 000100          CI1     042066          DATA0   002360          DRVER   011770
ADDW6 = 000000          AT7   = 000200          CI2     042232          DATA1   002420          DRVINT  041030
ADDW7 = 000000          AUNIT = 000000          CI3     042260          DATA10  003060          DRVMSG  056661
ADDW8 = 000000          AUSWR = 000000          CI4     042372          DATA11  003120          DRVNO   001320
ADDW9 = 000000          AUTLST  032100          CI5     042712          DATA12  003160          DRVPAR  001426
ADEVCT= 000000          AVAIL   001610          CI6     042734          DATA13  003220          DRVPRM  030402
ADEVM = 0 0000          AVECT1= 000254          CI7     042750          DATA14  003260          DRVQUE  046150
AENV  = 000000          AVECT2= 000000          CI7B    042764          DATA15  003320          DRVSN   060232
AENVM = 000000          A16   = 000400          CI8     043036          DATA2   002460          DRVSTA  040534
AFATAL= 000000          A17   = 001000          CKBUS   014110          DATA3   002520          DRVTYP  040544
AIR   = 000004          BADENT  060163          CKCLK   024574          DATA4   002560          DRY   = 000200
AMADR1= 000000          BADSEC  001336          CKCLK1  024656          DATA5   002620          DSWR  = 177570
AMADR2= 000000          BADTMO  003440          CKCLK2  024730          DATA6   002660          DTE   = 010000
AMADR3= 00000C          BAI   = 000010          CKCLK3  024762          DATA7   002720          DTEER   012616
AMADR4= 000000          BEGCOD  001514          CKERR   014010          DATA8   002760          DTUW    040626
AMAMS1= 000000          BEGPAT  001512          CKFMT   012022          DATA9   003020          DT00  = 000001
AMAMS2= 000000          BEGWC   001516          CKHCE   012212          DCK   = 100000          DT01  = 000002
AMAMS3= 000000          BELL  = 000007          CKLMTS  021412          DCKER   010546          DT02  = 000004
AMAMS4= 000000          BIT0  = 000001          CKSWR = 104407          DCKER1  010730          DT03  = 000010
AMSGAD= 000000          BIT00 = 000001          CK.CHR  033202          DCU   = 000040          DT04  = 000020
AMSGLG= 000000          BIT01 = 000002          CK.DEC  033154          DDISP = 177570          DT05  = 000040
AMSGTY= 000000          BIT02 = 000004          CK.DIG  033254          DDRVS   001544          DT06  = 000100
AMTYP1= 000000          BIT03 = 000010          CK.NUM  062654          DGE   = 000001          DT07  = 000200
AMTYP2= 000000          BIT04 = 000020          CK.OCT  033126          DH1     054031          DT08  = 000400
AMTYP3= 000000          BIT05 = 000040          CLKFLG  001310          DH14    054206          DT1     054504
AMTYP4= 000000          BIT06 = 000100          CLKOFF  024770          DH15    054305          DT14    054530
AOE   = 001000          BIT07 = 000200          CLR   = 000040          DH16    054404          DT15    054550
APASS = 000000          BIT08 = 000400          CLRDPB  030164          DH17    054464          DT16    054572
APRIOR= 000000          BIT09 = 001000          CLRQUE  046052          DH2     054036          DT17    054610
APTCSU= 000040          BIT1  = 000002          CMCNT   001366          DH3     054113          DT2     054510
APTENV= 000001          BIT10 = 002000          CMCYL   001370          DH4     054141          DT3     054514
APTSIZ= 000200          BIT11 = 004000          CMDAT   014450          DH6     054200          DT4     054520
APTSPO= 000100          BIT12 = 010000          CMHED   014360          DISPLA  001156          DT6     054524
ASGND   057703          BIT13 = 020000          CMPAR   014174          DISPLY= 104414          DVA   = 004000
ASGN1   027300          BIT14 = 040000          CMPARD  014200          DISPRE  000174          DVC   = 000200
ASGN2   027354          BIT15 = 100000          CMPLMT  001460          DLT   = 100000          ECBAD0  001412
ASGN3   027444          BIT2  = 000004          CMPRES  022260          DONE    010206          ECBAD1  001420
ASGN4   027610          BIT3  = 000010          CMPRT   015060          DPE   = 000010          ECBIT   001376
ASGN6   027612          BIT4  = 000020          CMPRX   015052          DPINT   040554          ECC     015576
```

SYMBOL TABLE

```
ECCX     016412   ENDCMP  015440   HOUR    001340   LINE6   024056   MAIN1   006506
ECC1     016210   ENDCON  001446   HT    * 000011   LINE6A  024070   MAIN2   006644
ECC2     016406   ENDING  001504   IAE   * 002000   LINE6C  024076   MASK    001322
ECGL     001410   ENDPGM* 061370   IAEER   012732   LINE6D  024104   MATCH   015506
ECGD1    001416   ENDSEK  001452   IBSAVE  035410   LINE7   024136   MAXCYL* 000134
ECH    * 000100   ENTADR  060124   IDLE    007234   LINE7A  024256   MAXER   001456
ECI    * 004000   ENTCOM  060061   IE    * 000100   LINE8   024354   MAXSEC* 000144
ECMSK0   001402   ENTLMT  060102   ILF   * 000001   LING6   055435   MAXTRK* 000140
ECMSK1   001404   ENTPR   005556   ILR   * 000002   LINM3   054713   MCPE  * 020000
ECSEC    001400   EQUAL   056651   ILV   * 000004   LINM4   055140   MDPE  * 000400
ECWRD    001406   ERCTR   001362   INCWRD  026106   LINN3   054731   MESFE   060306
ECWRD1   001414   ERPRC1  007636   INCMIS  026156   LINOCT  024366   MINCYL* 000136
EMPTYQ   046130   ERPROC  007622   INCSKI  026132   LINP3   054752   MINSEC* 000146
EMTVEC* 000030   ERR   * 040000   INCSOF  026062   LINP5   055253   MINTRK* 000142
EM1      051024   ERROR * 104000   INCTOT  026202   LINR6   055453   MINUTE  001342
EM10     051337   ERRVEC* 000004   INTRVL  001470   LINSS3  055124   MNTBL   002124
EM11     051402   EWN   * 000002   INVLD   060036   LINST3  055110   MOH   * 020000
EM12     051435   EWNERR  013474   IOTVEC* 000020   LINS3   054772   MOL   * 010000
EM13     051466   FACTOR  017570   IR    * 000100   LINS4   055156   MREAD   060570
EM14     051540   FAIRNS  001326   ISR     043254   LINS5   055270   MRPCS1  062636
EM15     051563   FALPAR  007776   ITCNT   022254   LINT3   055041   MRPVEC  062645
EM2      051077   FALPR1  010012   IXU   * 000100   LINU06  055465   MSGCON  057573
EM20     051617   FEONLY  060531   KSR     026534   LINW3   055077   MSGDRP  057437
EM21     051640   FER   * 000020   KSR1    026542   LINX4   055172   MSGEOP  057451
EM22     051671   FE1   * 000156   KWSVR   026352   LINX5   056613   MSGEOT  057477
EM23     051744   FILBUF  020162   LBC   * 002000   LIN10A  056145   MSGON   057676
EM24     052023   FILLZ   032626   LBT   * 002000   LIN10B  056201   MSGPG   057132
EM25     052071   FILLO   032734   LCE   * 001000   LIN10C  056241   MSGPWR  040102
EM26     052152   FMTER   013012   LF    * 000012   LIN10H  056345   MSGREP  057167
EM27     052177   FMTRK * 000163   LIMIT   001364   LIN11   056546   MSGSUM  057425
EM3      051135   FMT16 * 010000   LINA3   055022   LIN11A  056567   MSGX10  057632
EM30     052234   FRSTER  001352   LINB3   055070   LIN11H  056473   MSHELP  064370
EM31     052266   F0    * 000002   LINB5   055233   LIN2C   054616   MSHW1   057135
EM32     052331   F1    * 000004   LINB6   055345   LIN2P   054636   MSHW2   057145
EM33     052364   F2    * 000010   LINCA3  055051   LIN2S   054657   MSPASS  057612
EM34     052441   F3    * 000020   LINC6   055400   LIN3.1  023242   MSPRM   060244
EM35     052503   F4    * 000040   LINDA3  055061   LIN3.3  023350   MSTOTL  057622
EM36     052541   GENDPB  050734   LINDEC  024420   LIN3.4  023402   MSWAIT  040256
EM37     052572   GENPAR  020672   LIND5   055215   LIN6.2  024112   MSWRO   057773
EM4      051173   GENREG  050754   LINEN3  055006   LIN7M   055512   MXF   * 001000
EM40     052644   GETBUF  017572   LINE05  055333   LIN7P   055535   MXWNDW  040650
EM41     052702   GETID   031070   LINEP5  055323   LIN7R   055633   M.DPID  032250
EM42     052746   GETLMT  030766   LINE1   022274   LIN7S   055555   M.DP40  032306
EM43     053027   GETPAT  021352   LINE2   022342   LIN7T   055574   M.DP41  032342
EM44     053123   GETREG* 000141   LINE2A  022512   LIN7X   055615   M.DP42  032352
EM45     053220   GETREM  032102   LINE2B  022530   LIN8M   055650   M.DP44  032404
EM46     053306   GETREQ  046224   LINE3   022776   LIN9B   055673   M.DP50  032416
EM47     053360   GO    * 000001   LINE3A  023004   LIN9E   056070   N       057765
EM5      051230   GODRIV  020240   LINE3B  023012   LIN9G   056117   NED   * 010000
EM50     053435   GTSWR * 104406   LINE3C  023024   LIN9H   055722   NEDCLK  057714
EM51     053473   HCE   * 000200   LINE3D  023034   LIN9I   056015   NEM   * 004000
EM52     053536   HCEER   013070   LINE3E  023102   LKPAR   005246   NEWASN  030110
EM6      051264   HCI   * 002000   LINE3F  023170   LODEV   057041   NEWUNT  001566
EM60     053602   HCRC  * 000400   LINE4   023446   LODPAR  021742   NINLEV  057056
EM70     053625   HCRCER  011630   LINE5   023536   LSTAD   001330   NODFLT  057077
EM71     053704   HELPTX  064500   LINE5A  023746   MAIN    006340   NODRVS  060204
EM72     053762   HERTZ   001312   LINE5B  024014   MAINDA  006364   NOMTCH  013616
```

C2RJOBO RP07 PERF EXER MACRO V04.00 1 DEC 83 10:32:28 PAGE 135 3 SEQ 0247

SYMBOL TABLE

```
NONE    060156
NOOP  = 000101
NOTAVL  057012
NOTPRS  056775
NOTRP   056760
NOTSAF  057031
NSA   = 100000
OFFDIR= 000200
OFFON = 000001
OFLIN   010110
ONES    003262
ONESEC  001346
ONESUM  025122
OPI   = 020000
OPIER   012506
OPIER1  012556
OPT     041622
OPTBL   002104
OR    = 000200
ORDERQ  001520
OVRWRT  060367
PACK    030162
PAR   = 000010
PARENT  031170
PARER   012640
PARLST  060644
PAR1    061002
PAR11   061264
PAR12   061331
PAR14   061414
PAR15   061447
PAR16   061511
PAR19   061555
PAR2    061041
PAR21   061577
PAR3    061121
PAR4    061176
PAR5    061207
PAR6    061220
PAR7    061231
PAR8    061242
PAR9    061253
PASSES  001474
PAT   = 000020
PATCH   062764
PATTER  001476
PERIOD  057771
PERM    043626
PFECH   035570
PFECH1  035600
PFECH2  035662
PFECH3  035714
PFECH4  035724
PFTSTN  035730
PGE   = 100000
PGM   = 001000
PIP   = 020000
PIRQ  = 177772
PIRQVE= 000240
POPQUE  046246
POSER   012434
PROCES  007472
PRTBAD  016436
PRTIM   010150
PR0   = 000000
PR1   = 000040
PR2   = 000100
PR3   = 000140
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSEL  = 002000
PSW   = 177776
PUNSAF  007720
PWRFLG  040100
PWRUP   040124
PWRVEC= 000024
QCNT    045560
QDRV0   045652
QDRV1   045672
QDRV2   045712
QDRV3   045732
QDRV4   045752
QDRV5   045772
QDRV6   046012
QDRV7   046032
QINPT   045570
QOUTPT  045610
QSTART  045630
QSTOP   045632
QTERP = 046052
QUES    056647
RANCYL  021062
RANDOM  001510
RANDWC  001500
RANPAT  021322
RANSEC  021152
RANSIZ  021230
RANTRK  021116
RANXIT  021342
RATIO   001502
RDCHR = 104410
RDDAT = 000171
RDHD  = 000173
RDLIN = 104411
RDONLY  001424
RDY   = 000200
RD.RP   045116
READDR  024444
READHD  017026
READIN= 000121
RECAL = 000107
RECALT  016710
RECALO  017000
REDAPK  030120
RELBUF  017726
RELSE = 000113
REPLZ   032632
RESREG= 104413
RESVEC= 000010
RETRY   001324
REV     001432
RHEXT   040646
RMR   = 000004
RPADR   040640
RPAS  = 000016
RPBA  = 000004
RPBAE = 000050
RPCC  = 000036
RPCS1 = 000000
RPCS2 = 000010
RPCS3 = 000052
RPDA  = 000006
RPDB  = 000022
RPDC  = 000034
RPDS  = 000012
RPDT  = 000026
RPEC1 = 000044
RPEC2 = 000046
RPER1 = 000014
RPER2 = 000040
RPER3 = 000042
RPINIT  040652
RPLA  = 000020
RPMR1 = 000024
RPOF  = 000032
RPSN  = 000030
RPSTU0  040424
RPSTU1  040434
RPSTU2  040444
RPSTU3  040454
RPSTU4  040464
RPSTU5  040474
RPSTU6  040504
RPSTU7  040514
RPTMR   044402
RPVEC   040642
RPWC  = 000002
RP07    041364
RTC   = 000117
RTURN   032050
RWU1  = 002000
RWU2  = 004000
RWU3  = 010000
R6    =%000006
R7    =%000007
S       054747
SAVEFG  040602
SAVER1  001356
SAVER5  001360
SAVPOS  001354
SAVREG= 104412
SC      043630
SCMND   030006
SCOPE = 000004
SC04  = 000400
SC1   = 000100
SC10  = 001000
SC100 = 010000
SC11    044102
SC12    044200
SC13    044264
SC2   = 000200
SC20  = 002000
SC3     043676
SC4     043702
SC40  = 004000
SC5     043714
SC6     044024
SC8     044052
SEARCH= 000131
SECLMT= 000152
SECOND  001344
SEEK  = 000105
SEEKFG  040604
SELDRV= 000145
SEQPAR  020330
SETFMT= 000143
SETVEC  005604
SET.IE  045506
SIZE70  004742
SIZMEM  005112
SKI   = 040000
SKIER   013246
SLASH   060742
SPOTCK  016420
SRCHWT  040576
STA     006074
STACK = 001100
START   003522
START1  003532
START2  003550
STATIN  001314
STATIS  017270
STATPR  025020
STKLMT= 177774
STNDAT  002314
STO     044472
SUPRS   032442
SUPRSL  032426
SUPR1   032454
SUPR2   032516
SVRHXX  045342
SWR     001154
SWREG   000176
SWTIM   010052
SW0   = 000001
SW00  = 000001
SW01  = 000002
SW02  = 000004
SW03  = 000010
SW04  = 000020
SW05  = 000040
SW06  = 000100
SW07  = 000200
SW08  = 000400
SW09  = 001000
SW1   = 000002
SW10  = 002000
SW11  = 004000
SW12  = 010000
SW13  = 020000
SW14  = 040000
SW15  = 100000
SW2   = 000004
SW3   = 000010
SW4   = 000020
SW5   = 000040
SW6   = 000100
SW7   = 000200
SW8   = 000400
SW9   = 001000
T       054726
TAB     056657
TABLE   061644
TABLE0  061664
TABLE1  061732
TABLE2  062000
TABLE3  062046
TABLE4  062114
TABLE5  062162
TABLE6  062230
TABLE7  062276
TAB.XY= 001114
TAP   = 040000
TBITVE= 000014
TCF   = 000400
TD      043404
THEAD   020776
TIMER   040606
TKVEC = 000060
TPE   = 000002
TPVEC = 000064
TRAPVE= 000034
TRE   = 040000
TRFER   013146
TRKLMT= 000154
TRMREP  057274
TRNSWT  040574
TRTVEC= 000014
TSTANY  001422
TST1    003540
TYDRV   033044
```

SYMBOL TABLE

```
TYPDRV  033050   $CHARC  036262   $MLDWC= 000120   $PSEL = 000003   $SUPRL  032526
TYPDS = 104405   $CKSWR  034010   $ICNT   001120   $PWRAD  040066   $SUPRS  032542
TYPE  = 104401   $CMTAG  001114   $ILLUP  040072   $PWRDN  037720   $SUPR1  032554
TYPOC = 104402   $CM3  = 000000   $INTAG  001151   $PWRMG  040062   $SUPR2  032616
TYPON = 104404   $CM4  = 000001   $ITEMB  001130   $PWRUP  037772   $SVPC = 000210
TYPOS = 104403   $CNTLC  034715   $LF     001204   $QUES   001202   $SWR  = 122000
TYPSUM  025150   $CNTLG  034727   $LFLG   037203   $RAND   037206   $SWREG  001230
UCPAR   007760   $CNTLU  034722   $LKCSB  001300   $RDCHR  034352   $TATUS= 000016
UNS   = 040000   $CODE = 000024   $LKCSR  001276   $RDLIN  034442   $TERM = 000032
UNSAF   013402   $COMND= 000002   $LKS    001304   $RDPAS= 000036   $TESTN  001212
UNTASN  056732   $CPUOP  001234   $LLVEC  001306   $RDSZ = 000017   $TIME   026226
UNTNOT  056710   $CRLF   001203   $LONUM  037306   $REG  = 000014   $TKB    001162
UNTOFF  056667   $CYL  = 000012   $LPADR  001122   $REPLZ  032742   $TKCNT  033472
UNTON   056700   $DBDIV  032174   $LPERR  001124   $RESRE  037346   $TKINT  033510
UPE   = 020000   $DBLK   036730   $LPVEC  001302   $RETRY  017120   $TKQEN= 033507
US1   = 000001   $DB2D   037404   $LSTAD  062474   $RPADR  001272   $TKQIN  033474
US2   = 000002   $DB2O   037600   $MADR1  001240   $RPAS = 000204   $TKQOU  033476
US4   = 000004   $DECVL  037564   $MADR2  001244   $RPBA = 000172   $TKQSR  033500
VV    = 000100   $DEVCT  001216   $MADR3  001250   $RPBAE= 000236   $TKS    001160
WAIT    001632   $DEVM   001264   $MADR4  001254   $RPCC = 000224   $TKSRV  033560
WATPAK  030144   $DIV    032126   $MAIL   001206   $RPCS1= 000166   $TMPO   001174
WCE   = 040000   $DOAGN  032044   $MAMS1  001236   $RPCS2= 000176   $TN   = 000002
WCF   = 000040   $DRVSN= 000160   $MAMS2  001242   $RPCS3= 000240   $TNPWR  037514
WCFER   013304   $DSPLY  033100   $MAMS3  001246   $RPDA = 000174   $TOTAL= 000102
WCHKX = 040424   $DTBL   036720   $MAMS4  001252   $RPDB = 000210   $TPB    001166
WCKD  = 000151   $ENDAD  032034   $MBADR  001102   $RPDC = 000222   $TPFLG  001173
WCKER   011242   $ENDCT  032020   $MFLG   037202   $RPDS = 000200   $TPS    001164
WCKHD = 000153   $ENV    001226   $MISPO= 000112   $RPDT = 000214   $TRAP   040336
WC.HK   043316   $ENVM   001227   $MNEW   034745   $RPEC1= 000232   $TRAP2  040360
WLE   = 004000   $EOP    031500   $MSGAD  001222   $RPEC2= 000234   $TRK  = 000011
WLEER   012764   $EOPCT  032012   $MSGLG  001224   $RPER1= 000202   $TRP  = 000015
WOR   = 001000   $ERFLG  001117   $MSGTY  001206   $RPER2= 000226   $TRPAD  040372
WRDCNT  001466   $ERMAX  001131   $MSWR   034734   $RPER3= 000230   $TSTM   001104
WRDPOS  001374   $ERROR  035022   $MTYP1  001237   $RPLA = 000206   $TSTNM  001116
WRL   = 004000   $ERRPC  001132   $MTYP2  001243   $RPMR1= 000212   $TTYIN  034676
WRTCHK  001506   $ERRTB  003360   $MTYP3  001247   $RPOF = 000220   $TYPDS  036514
WRTDAT= 000161   $ERRTY  035412   $MTYP4  001253   $RPSN = 000216   $TYPE   035732
WRT.RP  045210   $ERTTL  001126   $NCODE= 000122   $RPVEC  001274   $TYPEC  036144
WRYUNS= 000400   $ETABL  001226   $NCYL = 000126   $RPWC = 000170   $TYPEX  036264
XXDP    001430   $ETEND  001272   $NEXT = 000130   $RPO7   057155   $TYPOC  036312
Y       057767   $FATR = 000116   $NPATC= 000123   $RPO7P  057162   $TYPON  036326
ZEROS   002360   $FATAL  001210   $NSEC = 000124   $RTNAD  032046   $TYPOS  036266
ZROIND  001350   $FFLG   037204   $NTRK = 000125   $RTOTL= 000066   $UNIT   001220
$APTHD  001100   $FILLC  001172   $NULL   001170   $SAVRE  037310   $UNITM  001110
$ATYC   036764   $FILLS  001171   $NWTST= 000000   $SAVR6  040076   $USWR   001232
$ATY1   036740   $FILLZ  032736   $OCNT   036510   $SB2D   033412   $VECT1  001256
$ATY3   036746   $FIRST= 000132   $OCTVL  037702   $SB2O   033442   $VECT2  001260
$ATY4   036756   $FMT  = 000001   $OMODE  036512   $SEC  = 000010   $WCNT = 000004
$AUTOB  001150   $GDADR  001134   $OPERC= 000052   $SEEKS= 000046   $WRDL = 000020
$BASE   001262   $GDDAT  001140   $PACK = 000026   $SETUP= 000156   $WTOTL= 000056
$BDADR  001136   $GET42  032024   $PASS   001214   $SIZE   062344   $WTPAS= 000042
$BDDAT  001142   $GTSWR  034100   $PASSC= 000114   $SKI  = 000110   $XOFF = 000023
$BELL   001176   $HARD = 000106   $PASTM  001106   $SOFT = 000104   $XON  = 000021
$BUF  = 000006   $HD   = 000000   $PATTC= 000030   $SSEC = 000022   $$GET4= 000000
$CDW1   001266   $HIBTS  001100   $PREVA= 000032   $STOTL= 000076   $OFILL  036511
$CDW2   001270   $HINUM  037304   $PREVO= 000027   $STUP = 177777   .$X   = 001100
```

```
CZRJOBC RPO7 PERF EXER MACRO V04.00 1-DEC 85 10:52:29 PAGE 155 5                SEQ 0249
SYMBOL TABLE


. ABS.  070405      000
        000000      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  62464 WORDS  ( 244 PAGES)
DYNAMIC MEMORY AVAILABLE FOR  71 PAGES
.CZRJOB'C-[20,12]CZRJOB.DOC,CZRJOB.MIS,CZRJOB.[20,0]SYSMAC/M
```

CROSS REFERENCE TABLE (CREF V04.00)

```
$$$EOP    73 1
$$$EOT    73 1
$$$IT    129 16    129-48
$$$MFG     1 5       1-11      4 3       5-53      5 82      9 0       9 0       12 99     15 5      15 87     16 6      21 45     22 4      22 37
          24-25     34-7      35 41     37 87     38 5      41 43     50 60     51 8      58 51     59 61     62 16     62-63     62 77     67 16
          67-65    128-80    133-43    134-50    135 14    135 15
$$$ODT     7-4
$$GET4    73-1      73 10
$OFILL    89-1      89-10     89-1•     89-1•
$40CAT    86-1
$APTHD     7-11      7-110
$ASTAT    91-1      91-1
$ATY1     91 10
$ATY3     88-1      91-10
$ATY4     86-1      91-10
$ATYC     91-1      91-10
$AUTOB     8-00     12-51•    12-77     12-112    14-16     15 10     15-88     17 16     20-7      58 31     67 30     73-1      73-1      85 1
          85-1      85-1
$BASE      8 00     12-72     12-74
$BDADR     8-00
$BDDAT     8-00
$BELL      8-00     57-8      85-1      85-1      85-1      86-1      86-1      86 1
$BUF      20-24•    20-62•    35-30     36-10     45-10     47 17     47 23     47 28     47 32     48-12     57-95     57-240    57 247    101-41
         120-100
$CDW1      8-00
$CDW2      8-00
$CHARC    88-1      88 10     88-1•     88-1•     88 1•
$CKSWR    85-10     97-1      97-1
$CM3       8-0       8-0       8 00      8-00
$CMTAG     8-00     12 25     12-25     12 25
$CNTLC    85-1      85-1      85-1      85-1      85 10
$CNTLG    85-1      85-10
$CNTLU    85-1      85-1      85-10
$CODE     36-5      36-31     37-45     40-16     42-33     42 39     45-14     45 16     45-37     48-10     50-35•    50-57•    50-63•    56 30•
          56 34•    56-42•    56-48     66-33•    66-43    107 27•   120-210
$COMND    43 9•     43 30•    44-14•    50-58•    50-67•    56-28     56-31•    56-36•    56-37     56-41•    66-14     66-35•    101-38    107 18
         107-28•   120 70
$CPUOP     8 00     13-5•     13-33•    57-83     57-258    57-280    114-32
$CRLF      8 00     12-51     16-18     16-69     18-39     19-15     23-14     23-15     23-21     23-22     34-22     34-37     38 40     38 65
          38 93     41-21     41-41     42-37     42-50     42-64     42-66     42-67     57-11     57-12     57-29     57-63     57-65     57 99
          57 113    57-132    57-152    57-171    57-186    57-208    57-231    57-250    57-290    57-304    57-317    57-324    57-362    57 394
          57-424    59-22     59-28     59-168    62-20     63-72     64-44     71-7      71-30     71-36     73-1      73-1      73-1      73 1
          73-1      85-1      85-1      85-1      85-1      86-1      86-1      86-1      86-1      86-1      87-1      87-1      87-1      87-1
          88-1      88-1      88-1      96-21    132-21
$CYL      36-16     43-20•    44-12•    49-15     50-24     50-26•    50 34•    56 15     56-46•    57-159    66-14     67-75•    101-44    107-25
         120 130
$DB2D     57-383    57-390    57-415    59-97     59-103    59-113    59-119    59-129    59-135    84-13     94-10
$DB2O     84-30     95-10
$DBDIV    45-26     45-47     75-260
$DBLK     90-1      90-1      90-10
$DECVL    94-1      94-10
$DEVCT     8-00     73-1•     73-1•
$DEVM      8-00
$DIV      40-11     42 26     57-275    74-10     74-380
$DOAGN    73-1      73-1      73-10
```

CROSS REFERENCE TABLE (CREF V04.00)

```
$DRVSN   66 25    66 27    69 25*   81 14   121 19@  121 23
$DSPLY   81 26@   97 2
$DTBL    90 1     90 1@
$ENDAD    8       73 1@
$ENDCT   73 1@
$ENV      8 00    12 51    12 67    17 11    86 1     88 1     91 1     91 1
$ENVM     8 00    12 25    88-1     88 1     91-1
$EOP     21 33    50 38    73 1@
$EOPCT   73 1     73 1@
$ERFLG    8 00    86 1     86-1     86 1*
$ERMAX    8-00
$ERROR   12-25    86-1@
$ERRPC    8 00    86-1     86 1     86-1     86 1*    86 1*    87 1     87 1
$ERRTB   11-00    87 1
$ERRTY   86-1     87-1@
$ERTTL    8 00    86-1     86-1     86 1*
$ETABL    8-00
$ETEND    7-11     8-00
$FAIR    20 27*   20-66*   20-75*   20 76   120-41@
$FATAL    8 00    91 1*
$FFLG    91 1     91-1@    91-1*    91 1*    91 1*
$FILLC    8-00    88-1     88-1     88-1
$FILLS    8 00    88 1     88-1
$FILLZ   80-13@
$FIRST   12 98*   12 98*   12-98*   12-98*   12 98*   12 98*   12 98*   12 98*   17 39*   66 25    67-11*   120-52@
$FMT    101 37   120-60   120-7    120-8    120-9    120-10   120-11   120-12   120 13   120 14   120-15   120-19
$GDADR    8-00
$GDDAT    8-00
$GET42   20-17    58 33    73-1@
$GTSWR   85 1@    97 1     97 1
$HARD    59 153   59 165   60 17    60 17*   120 37@
$HD       5 85     5-85     5 85
$HIBTS    7 11@
$HINUM   12 25*   74-8     92-1     92 1     92-1@    92 1*
$HLDWC   47-15    47-24    47-29    47-33    50-51*   56-52*   120-42@
$ICNT     8 00
$ILLUP   96 1     96-1     96-1@
$INTAG    8-00    85-1     85-1     85-1     85-1     85 1*
$ITEMB    8 00    86-1     86-1     86-1     86-1     86-1     86-1*    86 1*    87-1
$LF       8-00    85-1     85-1     85-1     86-1     86-1     88 1     88 1
$LFLG    91-1@    91-1*
$LKCSB    9 00    58-13*
$LKCSR    9-00    58-8     58 14*   58-47*
$LKS      9-00    58-20    58 25*   58-49*
$LLVEC    9-00    58 22
$LONUM   12-25*   51-21    74-7     92-1     92-1     92 1@    92-1*
$LPADR    8-00
$LPERR    8-00
$LPVEC    9-00    58-10
$LSTAD   14-6     131-1@   131-1*
$MADR1    8-00
$MADR2    8-00
$MADR3    8 00
$MADR4    8 00
$MAIL     7-11     7 11     8-00     12-19    12-25    12 31    86-1     88-1
$MAMS1    8-00
```

CROSS REFERENCE TABLE (CREF V04.00)

```
$MBMS2     8 00
$MBMS3     8 00
$MBMS4     8 00
$MBADR     7 110
$MFLG     91 1      91 10     91 1•     91 1•
$MISPO    57 419    59-159    59-165    60 19     60 190    120 590
$MREM     85 1      85 10
$MSGAD     8 00     91 1      91 1•
$MSGLG     8 00     91 1•
$MSGTY     8 00     91 1      91 1      91 1•     91 1•
$MSGR     62 24     85 1      85 10
$MTYP1     8 00
$MTYP2     8 00
$MTYP3     8 00
$MTYP4     8 00
$NCODE    51 25•    53-17     53 26     56-13     56 34     56 35     120 460   120 47    120 48    120 49    120 50    120 51    120 52    121 3
$NCVL     51 27•    51 41•    52 10•    56-46     120-500
$NEXT     20 26•    20 50     20-65•    50-70•    50-74•    53 30•    66 20     120 510
$NPATC    53 29•    56-44     120-470
$NSEC     51 26•    51-36     51-39•    52 32•    52-36     53-13     56 45     120 480
$NTRK     51-40•    52-21•    120-490
$NULL      8 00     88-1      88-1      88-1
$NWTST    12 190
$OCNT     89 10     89-1•     89 1•
$OCTVL    95 1      95 10
$OPMODE   89 1      89-10     89 1•     89-1•     89-1•     89 1•
$OPLRC    49 13•    49 14•    50-14     50 16     51 31     51 33     73-1      120 290
$PACK     17 40•    20 52     21-15     21-17     50 36     50 55     50-73•    56 9      63 81•    120 220
$PASS      8 00     12 25•    73-1      73-1      73-1•     73 1•     73-1•     73 1•
$PASSC    59 83     66-42•    73-1      73-1•     120-400
$PASTM     7 110
$PATTC    48 14     50-69•    56-44•    66-36•    66-37•    120-240
$PREVA    43 17•    43 18•    43 19•    49 15     50-23•    50-24•    56 15•    56 16•    56 19•    56-20•    56 21•    57 198    57 202    57 206
          57 214    107-25•   107-26•   120 250
$PREVO    43 8•     50 22•    56-11•    56 12•    56 39     57-39     107-24•   120 230
$PSEL    101 39     120 80
$PWRAD    96 10
$PWRDN    12 25     96-1      96-10
$PWRMG    96 10
$PWRUP    96 1      96 10
$QUES      8 00     85 1      85-1      85-1      85-1      86-1      86-1      98 1      88 1
$R2A      97 1
$RAND     51 7      52 39     53 10     53-40     92 10
$RDCHR    85 10     97-1      97 1
$RDDEC    97 1
$RDLIN    85 10     97-1      97 1
$RDOCT    97 1
$RDPAS    21 21     21 24     45-39•    45 40•    59 112    73 1      73 1      120-260
$RDSZ     85 1      85-10
$REG     101 45     120 140   122-8     122 8     122-8     122-8     122-8     122-8     122 8     122 8     122-8     122 8     122 8     122 8
          122 8     122 8     122-8     122 8
$REPLZ    34 31     38-78     41-9      41-29     80-150
$RESRE    93 10     97-1
$RETRY    25 27     27 13     28-20     29-27     30 33     30-70     31-11     32 28     33 12     33 29     33-57     44 340    44 60
$RPOT     16-50     59 73     128 240
$RPOTP    16 53     59-76     128-250
```

C2R XREF PROC 7 PRIM 6 VER MACRO V3A 00 1 DEC 83 10 % PAGE 5 4 SEQ 0253

CROSS REFERENCE TABLE (CREF V08 07)

```
$RPADR     9 00    12 74*   12 80    13 8     13 22    126 5    132 23   132 47
$RPAS    121 300   126 8
$RPBA     35 31    40 3     40 39    40 61    42 18    42 57    45 9     57 96    57-107   57 206   57 257*  57 262*  57 245*  57 264
         121 250   126 8
$RPBASE  121 450   126-10
$RPCC    121 300   126 9
$RPCS1    22 9     22 11    24-49    35 3     43 8     50 22    56-11    57 33    107 28   121 230  121-24   121 25   121 26   121 27
         121 28   121 29   121 30   121 31   121 32   121 33   121 34   121 35   121 36   121 37   121 38   121 39   121 40   121 41
         121 42   121 43   121 44   122 8     122 8     122-8    122 8     122 8     122 8     122 8     122 8     126 7
$RPCS2    24 45    27 25    28 13    29 19    30 25    31 45    32 8     32 24    35 5     57 270   121 270  126-7
$RPCS3    57-263   57 285   66 27    121 440  122 8     122-8    122 8     122 8     122-8    122 8     122-8    122-8    126-10
$RPDA     50 25    51 26    57 180   57 229   57 483   57 484   121 260  126 8
$RPDB     57 269   121 320  126 8
$RPDC     30 12    50 26    51 27    57 184   57 217   57 485   121 370  126 9
$RPDS     16-62    17 33    22 15    24 41    24 87    44 52    63 68    121 280  126 7
$RPDT    121 340   126-9
$RPMC1    25-7     25 15    25 15    27 32    27 36    27 38    40-22    57 311   121-410  126 7
$RPMC2    25-17    27-40    40 43    57 315   121 420  126 8
$RPLA1    24 37    24 53    24 57    24 61    24 65    24 69    24 73    24 77    24 81    24 85    24 91    24 95    25 5     26 7
          26 9     26-30    26 32    27-3     27 29    27 68    29-3     30 3     35 7     44 46    44 56    121 290  126 7
$RPMR2    35-12    121-390  126 7
$RPMR3    24-99    24 101   33-43    33-47    35-9     42-7     121 400  126-7
$RPLA    121-310   126 8
$RPPDR1  121 330   126-8
$RPOF    121 360   126-9
$RPSN     69 14    121 350  126-9
$RPVEC     9 00    12-71*   12 81    132-35
$RPWC     35 26    36 12    40-4     42 20    57-93    57 150   57-271   121 240  126-8
$RTNAD    73 10
$RTOTL    45-41*   45-42*   45-43    45 44    45 50*   45-51*   45 52*   45 53*   57 388   59 118   120 320
$SAVR6    96-1     96 10    96-1*    96 1*    96-1*
$SAVRE    93 10    97 1     97 1
$SHAD     34 30    38 77    40 34    41 8     41-28    57-460   59 84    61 14    61 19    61 24    70 17    84 110
$SHZO     57-444   84-280
$SEC      36 18    43 21*   44-11*   50 18    50-23    50 25*   50 28    50 32*   50-43    56 16    56 45*   57-169   67 80*   101 42
         107-26   120-110
$SEEKS    21 28    21 31    49-17*   49-18*   59-128   120-280
$SETUP     6-29     6-29     6 29     6-29     6 29     6 290    6 290    6 290    6-290    6 290    6 290    12-25    1_ 25    12 25
          12 25    12-25    12-25    12 25    12 25    12-25    12 25    12 25    12-25    12 31    12-31    12 31    73 1     73 1
          85-1     85-1     85 1     85-1     85-1     86 1     86 1     86 1     86 1
$SIZE     14-5     131-10
$SKI      57 422   59-156   59-165   60-18    60 18*   120-380
$SOFT     59-150   59-165   60 16    60-16*   120 360
$SSEC     36 24    36-29    36-30    40 10    42 25    50-53*   56-47*   56-50*   57 274   66-41*   66 45*   120 200
$STOTL    49 19*   49 20*   57-413   59 134   120 340
$STUP      6-29     6-29     6-29     6-29     6-29     6 290    6-290    6 290    6-290    6 290    6-290    6-290    6 290    6-290
          6 290
$SUPR1    78 19    78 240
$SUPR2    78-18*   78-23*   78 33    78-35*   78 370
$SUPRL    40 35    57 384   57-391   57-416   57 461   78-160
$SUPRS    78 210
$SVPC      7-8      7-80
$SWR       5 730    5 85     5-87     5 87     5 87     5-87     5 87     5 87     5 87     5 87     8 0      8 0      8 0      12 19
          12 25    73 1     73 1     73 1     73 1     73-1     86 1     86 1     86 1     86 1     86 1     86 1     86 1     86 1
          86 1     86 1     86 1     96 1
$SWREG     8 00    12 25
```

CROSS REFERENCE TABLE (CREF VOR 00)

$STATUS	21 6	22 7	22 36	22 51	22 53	22 55	22 57	22 59	22 61	26 53	26 21	26 24	27 58	43 14	
	43 34	44 19	44 35	44 42	45 7	57 48	57 237	66 20	101-48	107 234	120 150	126 9			
$TIMM	97 30														
$TI_TM	8 00	12 196	78 34	87 1	87 1										
$TIME	18 40	57 13	59 29	61 40	62 21	79 1	85 1	85 20							
$TMB	8 00	62 97	85 1	85 1	85 1	85 1	85 1	85 1	85 1	88 1	88 1				
$TMCNT	85 1	85 1	85-10	85 10	85 10	85 10	85 60								
$TMINT	12 66	17 3	62 26	73 4	85 1	85 1	85 10	85 10							
$TMMIN	85 1	85 1	85 10	85 9											
$TMOIN	85 1	85 1	85 10	85 10	85-10	85 10	85 10	85 70	85 80	85 9	85 110				
$TMOUT	85 1	85 1	85 10	85 10	85 10	85 10									
$TMUSR	85 1	85 1	85 1	85 10	85 11										
$TMS	8 00	62 90•	85-1	85 1	85 1	85 1	85 1	85 10	85-10	85 10	85-10	85 10	85 10	88 1	
	88 1														
$TMSRV	85 1	85-10													
$TMPO	8 00														
$TMN	5 740	5 85	12-10	12-10	12-10	12 190									
$TMPWR	98 1	98 1	98 10												
$TOTAL	57 370	59 162	60-20	60 20•	71-48	120-350									
$TPB	8-00	88 1	88 1	88-10											
$TPFLG	8-00	88 1	88 1	88 1											
$TPS	8 00	88 1	88-1	88-1											
$TRAP	12 25	97-10													
$TRAP2	97 1	97 10													
$TRK	44 10•	50 35•	57-164	67-70•	101-48	120-120									
$TRP	97 1	97 1	97 1	97 1	97-1	97-1	97-1	97 1	97 1	97-1	97-1	97-1	97 1	97 1	
	97 1	97 1	97-1	97-1	97-1	97-1	97-1	97-1	97-1	97-1	97-1	97-1	97 1	97-1	
	97 1	97 1	97-1	97 1	97-1	97-1	97-1	97-1	97-1	97-1	97-1	97-1	97 1	97 1	
	97 1	97 1	97-10	97 10	97-10	97-10	97-10	97-10	97-10	97 10	97-10	97-10	97 10	97-10	
	97-2	97 2	97-2	97-2	97-20										
$TRPRO	97-1	97-10	97-3												
$TSTM	7-110														
$TSTMM	8 00	86 1	86 1	86 1											
$TTYIN	85 1	85 1	85-1	85 1	85-1	85-10									
$TYPBN	97-1														
$TYPOS	90 10	97-1	97 1												
$TYPE	81 29	88 10	91-1	97 1	97-1										
$TYPEC	85-1	88 1	88-1	88-1	88-10										
$TYPEX	88 1	88 1	88 10												
$TYPOC	89-10	97 1	97 1												
$TYPON	89 1	89 10	97-1												
$TYPOS	89 10	97-1													
$UNIT	8-00	9-0													
$UNITM	7 110														
$USRR	8 00														
$VECT1	8 00	12-69	12-71												
$VECT2	8 00														
$WCNT	44 15•	50-50•	50-52•	56-51•	56-53•	66-39•	66-40•	101-40	120-90						
$WROL	35-28	36-11	40-5	42-19	46-15	46-22	46-24•	46-25	46 26•	48-13	50-46•	50-47•	50 49•	50 50	
	50-51	53 23•	56-51	56-52	57-92	57-243	57-272	66-38•	120 190	120 20	120-21	120 22	120 23	120 24	
	120-25	120-26	120-27	120-28	120-29	120-30	120-32	120-34	120-35	120-36	120-37	120-38	120 39	120 40	
	120-41	120 42	120-46												
$WTOTL	45 20•	45-21•	45 22	45 23	45-29•	45-30•	45-31•	45 32•	57-381	59-102	120-300				
$WTPAS	45 18•	45 19•	59-96	120-270											
$XOFF	88-1	88-1													
$XON	85-1	88-1	88-1												

K4

C70 [illegible] V08 00 1 DEC 05 10-32:20 PAGE 9-6 SEQ 0258

CROSS REFERENCE TABLE (CREF V08 00)

$AS[illegible]A	91 1	91 1				
$$	7 11	7 110				
A16	5 1120	13 26	13 31			
A17	5 1150	13 26	13 31			
[illegible]	6 260	8 0	8 0			
[illegible]	21 13	71 410				
A[illegible]W1	8 0	8 0				
A[illegible]W2	8 0	8-0				
AC[illegible]OP	8 0	8 0				
ACT[illegible]	99 450	104 130	104 670	107 30	107-110	110-6
AC[illegible]TR	99 530	110 80	110 220			
A[illegible]W0	8 0					
A[illegible]W1	8 0					
A[illegible]W10	8 0					
A[illegible]W11	8 0					
A[illegible]W12	8 0					
A[illegible]W13	8 0					
A[illegible]W14	8 0					
A[illegible]W15	8 0					
A[illegible]W2	8 0					
A[illegible]W3	8 0					
A[illegible]W4	8 0					
A[illegible]W5	8 0					
A[illegible]W6	8 0					
A[illegible]W7	8-0					
A[illegible]W8	8-0					
A[illegible]W9	8-0					
ADI VCT	8 0	8-0				
ADI VM	8 0	8 0				
AE NV	8-0	8 0				
AE NVM	8 0	8 0				
AFATAL	8-0	8-0				
AIR	5 2640					
AMADR1	8 0	8-0				
AMADR2	8 0	8-0				
AMADR3	8 0	8-0				
AMADR4	8 0	8 0				
AMAMS1	8 0	8-0				
AMAMS2	8 0	8 0				
AMAMS3	8 0	8-0				
AMAMS4	8 0	8-0				
AMSGAD	8 0	8-0				
AMSGLG	8-0	8-0				
AMSGTY	8 0	8-0				
AMTYP1	8 0	8 0				
AMTYP2	8 0	8 0				
AMTYP3	8 0	8-0				
AMTYP4	8-0	8 0				
AOE	5 1860					
APASS	8 0	8-0				
APRIOR	8 0					
APTCSU	88 1	91-10				
APTENV	86-1	88-1	91-1	91-10		
APTSIZ	12 25	91-10				
APTSPO	88 1	91-1	91-10			
ASGN1	63 250					

CROSS REFERENCE TABLE (CREF V04.00)

```
BIT12     5 1040#  13 33    22-51    24 91    26 6     26 7     26 32    57-225#  103 25   103 58   103 77   104 60   105 25   106 203
        112 16     113 26   115-15
BIT13     5 1040#  13 33    24 65    30 68    32 24    81 26    86 1     112-18#
BIT14     5 1040#  22-11    22 13    22 59    24 37    24 45    24 49    24-99#   27 25    28 13    29 19    30 25    31-45    32-8
         33 27     57-278#  104 51   105 22   108 44   109 124  111-85   115 12#
BIT15     5 1040#  22-9     24-101   26-6     27-68    32 24    104 51   104-57#  104 60   105-22   105 25   106 20   106-55   106-175
        106 205    106 208  108-39   109 124  111-28   111 43   111-85   111-93#
BIT2      5 1040#  22-61    33-47    111 93
BIT3      5-1040#  24 69    113-30
BIT4      5 1040#  29 25
BIT5      5 1040#  24 73
BIT6      5 1040#  26 32    57-90
BIT7      5 1040#  26-24    30-31    44-42
BIT8      5 1040#  28 18    29-3     29 8     30-3     30 8     32-24
BIT9      5 1040#  24 85    32-24
BLKADR    9-0#     12-85    16-59    17-30    59-24    63 60    63-80    64-15    96-33
BLNKS1   19-19     42-47    42-59    57-14    59-70    59-72    59-80    62-23    63-76    70-20    70 22    127-67#
BLNKS2   16-47     34-34    38-62    38-81    41-12    41-17    41-17    41 32    41-37    41-37    42-56    57-69    57-110   57-161
        57 166     57-182   57-302   57 302   57-313   127-66#  132 27   132 39#
BLNKS3   42-44     127-65#
BLNKS4   16-20     127-64#
BPE       5-266#
BPTVEC    5-104#
BSE       5-274#   42-7
BUFTBL    9-0#     14-7#    14-8#    14-9#    14-12#   14-13#   14-15#   14-18#   14-19    14-21    46-12    46-14    46-28#   47-12
         47-13     47-25#   47-30#   47-35    47-36    47-49#
BUSADR   12-79     132-13#
CFLAG     9-0#     12-95#   12-120   15-18#   15-22    15-37#   15-43    15-86#   15-94    17-50#   18-3     20-9#    34-5     34 24
         38-46     62-19#   62-32    63-4#    63-9     70-9#    70-25    70-35#   85-5#    96-22#   96-26    132-22#  132-30   132-42
CMGADR    9-0#     12-15#   12-18#   17-18    17-47#   67-32    96-30#   132-13   132-15#
CMKWC    50-45     53-14    55-16#
CI1      105-39    106-12#
CI2      106-44    106-53#  106-67   106-103
CI3      105-42    106-61#
CI4      105-32    106-86#
CI5      106 51    106-84   106-160# 106-170
CI6      106-104   106-115  106-118  106-120  106-166#
CI7      104-37    105-52   106-23   106-33   106-37   106-49   106-58   106-66   106-77   106-82   106-94   106-102  106-110  106 114
        106-125    106-129  106-140  106-152  106-169  106-172# 108 16   109-67   114-31
CI7B     106-177#  109-70
CI8      106-180   106-190# 109-7    111-76
CK.CHR   82-32#    83-20    83-28
CK.DEC   82-10#    82-38
CK.DIG   70-27     83-14#
CK.NUM  132-32     132-44   133-10#
CK.OCT   81-42#    82-40
CKBUS    22-19     35-26#
CKCLK    16-4      58-5#    62-38    73-7     96-19
CKCLK1   58-7      58-17#
CKCLK2   58-19     58-28#
CKCLK3   58-15     58-26    58-37#
CKERR    22-18     35-3#
CKFMT    24-59     29-3#
CKHCE    24-63     30-3#
CKLMTS   50-19     50 29    51-37    52-37    54-15#
CKSWR    86-1      86-1     97-1#
```

CROSS REFERENCE TABLE (CREF V04.00)

```
CLKFLG    9 0#    58-5*    58-9*    58-21*   58 44    61-4
CLKOFF   58-44#   62-27
CLR       5-132#  62-15
CLRDPB   17-31    63-66    66-9#
CLRQUE  102-17   106-226  111-52   117-18#
CMCNT     9-0#    36-11*   36-12*   36-24    36-26    36-27*   36-30*   37-47    37-50*   37-55    37-83*
CMCYL     9-0#    36-16*   36-17*   36-36    37-65*   37-66    37-72
CMDAT    36-32    36-47    37-3#    37-51    37-53
CMMED    36-36#
CMPAR    22-31    36-3#
CMPARD   26-66    36-5#
CMPLMT    9-0#    34-43    36-19
CMPRES   18-45    20-37    20-88    21-40    56-67#   56-70
CMPRT    36-51    37-18    37-30    37-42    38-8#
CMPRX    36-48    37-49    37-56    37-70    37-84    37-93#
CMPTIM    9-0#    12-94    12-94*   20-15    20-15*   22-22    22-24    22-24    22-26    22-28    22-30*   61-39*   129-5
CMSEC     9-0#    36-18*   37-57*   37-58    37-60*
CMSTR    36-21#
CMSTR2   36-41    36-45    36-50#   37-77    37-81
CMTRK     9-0#    37-61*   37-62    37-64*
COLON    61-17    61-22   128-57#
COMMA    19-18    59-79    62-22    63-75   128-5#
COMTBL    9-0#    56-36    66-35
CPSAVE   86-1     86-1     86-1     86-1     86-1     86-1#    86-1*    86-1*    87-1
CR        5-104#  88-1     88-1
CRLF      5-104#   9-0     12-6     12-31    12-31    12-45    12-56    12-62    59-93    59-109   59-125   59-148   59-149   88-1
          88-1    96-5     96-37    96-38    96-38   125-6    125-7    125-8    125-9    127-48   127-49   127-50   127-51   127-53
         127-55  127-56   127-58   127-59   127-60   127-61   127-62   127-63   128-19   128-20   128-26   128-28   128-30   128-37
         128-41  128-48   128-48   128-52   128-53   128-54   128-55   128-56   128-56   128-59   128-60   128-60   128-62   128-83
         128-84  128-85   128-86   128-88   128-88   128-89   128-89   129-25   134-6    134-9    134-10   134-11   134-12   134-13
         134-14  134-15   134-16   134-17   134-18   134-19   134-20   134-21   134-22   134-23   134-24   134-25   134-26   134-27
         134-28  134-29   134-30   134-31   134-32   134-33   134-34   134-35   134-36   134 37   134-38   134-39   134-40   134-41
         134-42  134-43   134-44   134-45   134-46   134-47   134-48   134-49   134-51   134-53   134-54   135-1    135-2    135-3
         135-4   135-5    135-6    135-7    135-8    135-9    135-10   135-11   135-12
CTRAP     85-1    85-1     85-5#
CYLLMT   37-68    67-13    67-37    67-38    67-73    68-10*   68-17*   121-12#  121-13   121-14   121-15   121-19
CYLNDR   30-10    30-11*   30-12    30-15*   30-48*   57-224   57-302    57-302   123-8    133-79#
DASH     59-71   128-6#
DASH13   59-50    64-43    73-1     73-1    128-32#
DASH5    18-41    59-30    73-1    128-31#
DATA0    10-0     10-0#
DATA1    10-0     10-0#
DATA10   10-0     10-0#
DATA11   10-0     10-0#
DATA12   10-0     10-0#
DATA13   10-0     10-0#
DATA14   10-0     10-0#
DATA15   10-0     10-0#
DATA2    10-0     10-0#
DATA3    10-0     10-0#
DATA4    10-0     10-0#
DATA5    10-0     10-0#
DATA6    10-0     10-0#
DATA7    10-0     10-0#
DATA8    10-0     10-0#
DATA9    10-0     10-0#
```

CROSS REFERENCE TABLE (CREF V04.00)

CROSS REFERENCE TABLE (CREF V04.00)

```
DRVMSG  19 16    23-26    59-68    63-73    71 9     71-38    128 8#
DRVNO    9 0#    22-3*    23-27    57-67    86 1*
DRVPAR   9 0#    63-5*    63-21*   63-70    67 34
DRVPRM  17 36    63-79    67-8#
DRVQUE 104-29   104-40   107-20   118-22#
DRVSN   81-11   128-61#
DRVSTA  16-21    17-28    63-63    98-37#  102-29*  102-30*  102-31*  102-32*  102 43*  103 21*  103-30*  103 73*  103 79*  104-19
       104-24   104-49   105-19   105 23   106 231* 109-32   109-38   109-44   109-119  111 80*
DRVTYP  16-25    16-49    59 74    63-88    68-14    98-52#  103-22*  103-41*  103 46*  103 51*  104 55   106-232*
DR1      5 165#
DSWR     5-104#   8-0      12-25
DT00     5-211#
DT01     5-212#
DT02     5-213#
DT03     5-214#
DT04     5-215#
DT05     5-216#
DT06     5-217#
DT07     5-218#
DT08     5-219#
DT1     11-6     126-1#
DT14    57-70    126-7#
DT15    57-76    126-8#
DT16    57-80    126-9#
DT17    57-88    126-10#
DT2     11-13     11-34   126-2#
DT3     11-20    126-3#
DT4     11-27    126-4#
DT6     11-41    126-5#
DTE      5-189#
DTEER   24-93     30-81#
DTUW    62-10    100-33#  102-24   105-35   106-50*  106-187* 106-198  106 216  106-218* 106-224* 107-6    108-6*   111-25   111-40
       111-50*  111-62
DVA      5-157#
DVC      5-269#
ECBAD0   9-0#    40-51*   41-17
ECBAD1   9-0#    40-68*   41-37
ECBIT    9-0#    40-22*   40-23*   40-24    40-25*   40-45*
ECC     26-57    40-3#
ECC1    40-66    41-2#
ECC2    40-41    40-64    41-40#
ECCX    41-20    41-38    41-41#
ECGD     9-0#
ECGD1    9-0#
ECH      5-183#  25-5     27-29
ECI      5-242#
ECMSK0   9 0#    40-43*   40-47*   40-52    40-54    40-63
ECMSK1   9-0#    40-44*   40-48*   40-57    40-69    40-71
ECSEC    9-0#    40-3*    40-15*   40-18*   40-20*   40-38
ECWRD    9-0#    40-24*   40-26*   40-27*   40-28*   40-29*   40-31    40-38*   40-39    40-51    40-53    40-54*   40-55*   40-59
        41-6     41-17
ECWRD1   9-0#    40-59*   40-60*   40-61    40-65*   40-68    40-70    40-71*   40-72*   41-19    41-25    41-37
EM1     11-4     124-3#
EM10    23-16    124-9#
EM11    23-23    124-10#
EM12    23-4     124-11#
```

CROSS REFERENCE TABLE (CREF V04.00)

```
EM13      23 37    124-120
EM14      24 4     124-130
EM15      24 15    124-140
EM2       11 11    124-40
EM20      28-8     124-150
EM21      25-39    124-160
EM22      27-24    124-170
EM23      27-6     124-180
EM24      29-15    124-190
EM25      30-21    124-200
EM26      31-41    124-210
EM27      32-4     124-220
EM3       11-18    124-50
EM30      28-32    124-230
EM31      30-61    124-240
EM32      30-84    124-250
EM33      31-4     124-260
EM34      33-4     124-270
EM35      31-22    124-280
EM36      31-32    124-290
EM37      27-27    124-300
EM4       11-25    124-60
EM40      32-19    124-310
EM41      35-34    124-320
EM42      38-33    124-330
EM43      34-16     34-21   124-340
EM44      35-16    124-350
EM45      25-23     27-44   124-360
EM46      57-98    124-370
EM47      25-20     27-42   124-380
EM5       11-32    124-70
EM50      32-39    124-390
EM51      30-45    124-400
EM52      25-10     27-31   124-410
EM6       11-39    124-80
EM60      33-23    124-420
EM70      33-45    124-430
EM71      33-49    124-440
EM72      33-51    124-450
EMPTYQ   106-181  106-230  108-40   111-94   118-70
EMTVEC     5-104#   12-25*   12-25*
ENDCMP    37-93    38-86#
ENDCON     9-0#    17-4*    17-5*    17-8*    17-9*    17-13*   17-14*   21-21    21-24
ENDING     9-0#    21-19   129-10
ENDPGM    14-8     14-13   133-80#
ENDSEK     9-0#    21-28    21-31
ENTADR   128-56#
ENTCOM    62-28   128-54#
ENTLMT    67-43   128-55#
ENTPR     15-101   15-108#
EQUAL    128-4#
ERCTR      9-0#    36-9*    36-50*   37-14    37-29*   38-88    38-91
ERPRC1    22-44    22-50#   26-28    44-69
ERPROC    22-8     22-12    22-14    22-36#
ERR        5-172#
ERROR      5-104#
```

CROSS REFERENCE TABLE (CREF V04.00)

```
ERRVEC    5 104#  12-25    12-25*  12-25*  12 27*  12-28*  13 6    13-7*   13 34*  58 6    58 7*   58 19*  58 37*  86 1
          86 1*   86-1*    131-1   131-1   131 1*  131 1*  131 1*  131 1*
EWN        5 162#
EWNERR    24-43    33-41#
FO         5-152#
F1         5 153#
F2         5-154#
F3         5-155#
F4         5-156#
FACTOR    45-9*    45 10*  45-12*  45 18   45 20   45 39   45 41   45 59#
FAIRNS     9-0#
FALPAR    22 56    23-21#
FALPR1    23-17    23-25#
FE1       54-43    55-46   67 19   67 68   67-70   67 71   68-9*   68-16*  106-53  121-15#
FEONLY    15-59   128-88#
FER        5-181#
FILBUF    20-29    20-64   48-9#
FILLO     79-11*   79-15   79-36*  79-39#  80-13*  80-17   80-38*
FILLZ     61-15    61-20   61-25   79 11#
FMT16      5-243#
FMTER     29-10    31-40#
FMTRK      6-22#
FRSTER     9-0#    26-65*  26-67   34-13   34-42*  36-3*   37-33   38-28   38-43*  38-86
GENDPB    43-30*   43-32   43-34   44-10*  44-11*  44-12*  44-13*  44-14*  44-15*  44-17   44-19   63-58*  123-3#
GENPAR    20-57    51-7#
GENREG    29-8     30-8    123-12  123-15#
GETBUF    20-23    20-61   46-9#
GETID     17-32    63-67   69-10#
GETLMT    67-10    68-7#
GETPAT    50-68    53-28   53-35#  53-43
GETREG     6-15#
GETREM    51-15    52-8    52-19   52-30   53-7    53-41   74-7#
GETREQ   105-12   106-200  108-28  109-34  109-60  109-66  109-123 111-27  111-42  111-82  111-90  119-10#
GNS        7-1      7-1     12-6    12-31   12-45   12-56   12-62   59-82   59-93   59-109  59-125  59-148  59-149  59-152
          59-155   59-158   59-161  97-1    97-1    97-1    97-1    97-1    97-1    97-1    97-1    97-1    97-1    97-1
          97-1     97-1     97-1    97-1    97-1    97-1    97-1    97-1    97-1    97-1    97-1    97-2    97-2
GO         5-151#
GODRIV    20-30    20-68    26-20   27-57   44-34   49-7#
GTSWR     12-31    97-1#
HCE        5-184#
HCEER     30-16    32-3#
HCI        5-241#
HCRC       5-185#
HCRCER    24-55    28-3#    29-5    30-5
HELPTX    12-134   134-10#
HERTZ      9-0#    12-89    20-10   61-33   61-49
HOUR       9-0#    12-90*   20-11*  61-7    61-44*  61-45   61-47*
HT         5-104#  88-1     88-1
IAE        5-187#
IAEER     24-79    31-21#
IBSAVE    86-1     86-1     86-1    86-1    86-1    86-1#   86-1*   86-1*   86-1*
IDLE      18-7     20-43    20-48   21-3#
IE         5-110#
ILF        5-177#
ILR        5-178#
ILV        5-163#  16-62    17-33   63-68
```

CROSS REFERENCE TABLE (CREF V04.00)

```
INCMRD  26-48    27-83    60-170
INCMIS  30 50    60 190
INCSKI  32-43    60-180
INCSOF  26 56    26-64    27-80    28 16    29 23    30 29    30 65    60-160
INCTOT  23-8     23-28    23-41    24 8     24 19    25 25    26 26    26 49    26 69    27 11    27 84    28 17    28 35    29-24
        30-30    30-51    30-67    31 8     31 25    31 34    31 49    32-12    32-23    32 42    33 8     33 26    33-54    35-20
        35-38    38-94    60-200
INTRVL   9 0#    12-93#   20-14#   61 38#   61 53    61-55    61-55    61-58#   96 17#   96 28    129-4
INVLD   62-94   128-530
IOTVEC   5-104#
IR       5-133#
ISR    102-34   107-3#
ITCNT   56 23#   56-25    56-25#   56-57#
IXU      5-268#
KIPARO 131-1
KSR     18-8     62-10#   62-11
KSR1    62-12#
KWSVR   58-11    58-23    61-31#
LBC      5-272#
LBT      5-168#
LCE      5-271#
LF       5-104#  12-6     88-1     88-1     96-5    128-26   128-60
LIMIT    9-0#    34-43#   36-19#   36-20#   38-48    38-50#
LIN10A  40-30   127-54#
LIN10B  40-37   127-55#
LIN10C  41-40   127-56#
LIN10M  41-2    127-58#
LIN11   42-41   127-62#
LIN11A  42-52   127-63#
LIN11M  42-38   127-61#
LIN2C   57-32   127-4#
LIN2P   57-38   127-5#
LIN2S   57-64   127-6#
LIN3.1  57-120   57-126   57-191#
LIN3.3  57-131   57-138   57-213#
LIN3.4  57-139   57-223#
LIN6.2  57-331   57-337   57-343   57-358#
LIN7M   57-418  127-36#
LIN7P   57-412  127-40#
LIN7R   57-387  127-44#
LIN7S   57-421  127-41#
LIN7T   57-377  127-42#
LIN7X   57-380  127-43#
LIN8M   26-27    44-68    57-433  127-46#
LIN9B   38-39   127-47#
LIN9E   38-90   127-52#
LIN9G   26-70   127-53#
LIN9M   38-41   127-48#
LIN9I   34-23   127-50#
LINA3   57-223  127-14#
LINB3   57-146  127-18#
LINB5   57-268  127-26#
LINB6   57-323  127-31#
LINC6   57-330  127-32#
LINCA3  57-183  127-16#
LIND5   57-256  127-25#
```

CROSS REFERENCE TABLE (CREF V04.00)

```
LINDAS   57 179   127 170
LINDEC   38 92     57-160   57-165   57-170   57-192   57-194   57-196   57 199   57-203   57 207   57 215   57 218   57 226   57 230
         57 244    57-249   57 289   57 360   57 379   57-420   57-423   57 4590
LINE1    23 5      23-36    24-3     24 14    25 9     25 19    25 22    25-38    27-5     27 23    28 7     28 31    29 14    30 20
         30 44     30-60    30 83    31-3     31 21    31 31    31-40    32 3     32 18    32 38    33 3     33-22    33 41    34 15
         35-15     35-33    38-32    57-60
LINE2    23-5      23-38    24-5     24 18    25-24    26-2     27-10    27 51    28-9     28 33    29-16    30 22    30-46    30-62
         31-5      31-23    31-33    31 42    32 5     32-20    32-40    33-5     33-24    33 52    34 17    35-17    35-35    38-34
         57-280    57-434
LINE2A   57-37     57-42    57-610
LINE2B   57-62     57-650
LINE3    23 6      23-39    24 6     24 18    26 3     27 10    27 51    28 10    28 34    29-17    30-23    30-63    31-43    32-6
         32-21     33-25    33 53    35-18    57 1190
LINE3A   33-6      34-18    38-35    57-1250
LINE3B   32-41     57-1310
LINE3C   30-47     57-1380
LINE3D   35-36     57-100   57-1440
LINE3E   31-6      57-1580
LINE3F   31-24     57-1770
LINE4    23-7      23-40    24-7     24-18    26-4     27-10    27-51    28-11    29-18    30-24    30 64    31-7     31-44    32 7
         32-22     33-7     34-19    35 19    35-37    38-36    57-101   57-2370
LINE5    27-10     27-51    28-15    29-21    30-27    31-47    32-10    57-2560
LINE5A   28-12     29-22    30-28    30-49    31-48    32-11    57-2960
LINE5B   26-55     57-3100
LINE6    26-13     26-54    57-3230
LINE6A   26-63     57-3300
LINE6C   25-29     27-16    27-62    28 22    29 29    30-35    30-72    31 13    32-30    33-14    33-31    33-59    57 3360
LINE6D   25-32     26-60    27-19    28 25    29-32    30-38    30-75    31-16    32-32    33-17    33-33    33-61    57-3420
LINE7    23-9      23-42    24-9     24-20    25-33    26 50    26-71    27-85    28-23    28-26    28-36    29-30    29 33    30-36
         30-39     30-73    30-76    31-14    31-26    31-35    31-50    32-13    32-33    33-15    33-36    35-21    35-39    38-95
         44-64     57-3770
LINE7A   30 52     32-44    57 4120
LINE8    26-47     27-78    44-63    57-4330
LINEN3   57-216   127-130
LINEO5   57-314   127-300
LINEP5   57-310   127-290
LING6    57-336   127-330
LINM3    57-119   127-70
LINM4    57-239   127-220
LINN3    57-125   127-90
LINOCT   34-27     34-36    38-58    38-61    38-64    41-6     41 17    41-17    41-25    41-37    41-37    42-43    42-46    42-49
         42-55     42-61    57-68    57-108   57-148   57-151   57-181   57 185   57-241   57-267   57-270   57-302   57-302   57-312
         57-316    57-4430
LINP3    57-197   127-110
LINP5    57-288   127-270
LINR6    57-361   127-340
LINS3    57-158    57-213   127-120
LINS4    57-242   127-230
LINS5    57-297   127-280
LINSS3   57-167   127-210
LINST3   57-162   127-200
LINT3    57-227   127-150
LINU06   57-342   127-350
LINW3    57-149   127-190
LINX4    57-245   127-240
LINX5    57-303   127-680
```

CROSS REFERENCE TABLE (CREF V04 00)

```
LKPAR    15 30     15 23     15 33     15 44
LODEV    16 44     63 30     63 44    128 :70
LODPAR   20 58     56-80
LSTAD     9 00     14 6•     14 9      14 10
M.DP40   76 250    76-46
M.DP41   76 30     76-370
M.DP42   76-36     76 410
M.DP44   76 48     76 530
M.DP50   76 23     76 570
M.DPID   74 45     75 33     76 140
MAIN     18 30     21 58     50 76     73 8
MAIN1    18 20     19 30
MAIN2    19 13     20 40
MAINDA   18 130
MASK      9 00     26-6•     26-30     27-7•     28 18•    29-25•    30-31•    30 68•    31-9•     32-27•    33 11•    33-27•    33 55•    44 44
         44 50     44 56
MATCH    37 4      39 160
MAXCYL   52-3      54-38     55 28     67-13•    67-47     67-50•    67 71     67-73•    121-30    121 4     121 5     121-6     121-7     121 8
        121 12    130-20    130 20    130 20    130 20    130 20    130 20    130 20    130-20
MAXER     9-00     71-43
MAXSEC   52-25     54-29     55-19     55 34     67-15•    67 59     67 62•    121 70    130 20    130-20    130 20    130 20    130-20    130-20
        130 20    130 20
MAXTRK   52 14     54-33     55-23     55-30     67-14•    67-55     67 56•    121-50    130 20    130-20    130-20    130-20    130-20    130 20
        130-20    130-20
MCPE      5-1150
MDPE      5-1350
MESFE    15-19     128-830
MINCYL   50 34     51-41     52-4      52-6      52-9      54-16     54-18     67-19•    67-21•    67-46     67-51•    67-68     67 70•    67-75
        121-40    130-20    130-20    130-20    130-20    130-20    130-20    130 20    130-20
MINSEC   50-32     51-39     52-26     52-28     52-31     54-22     54 24     54 31     54-35     55-17     67-23•    67-58     67-63•    67-80
        121 80    130-20    130-20    130-20    130-20    130-20    130-20    130-20    130-20
MINTRK   50 33     51-40     52-15     52-17     52-20     54-19     54-21     54 36     55-21     67 22•    67 52     67-57•    67-79     121-60
        130-20    130-20    130-20    130-20    130-20    130-20    130 20    130 20
MINUTE    9-00     12-91•    20-12•    61-18     61-40•    61-41     61-43•
MNTBL     9-00     57-53
MOM       5-2210
MOL       5-1700
MREAD    15-63     128-890
MRPCS1  132 24     132-530
MRPVEC  132-36     132-540
MSGCON   64-45     128-370
MSGDRP   18-42     128-340
MSGEOP   73-1      128-350
MSGEOT   73-1      128-360
MSGOM    23-25     71-32     128-420
MSGPG    81-12     128-210
MSGPWR   96 1      96-50
MSGREP   21-54     128-260
MSGSUM   59-31     73-1      128-330
MSGX10   57-386    57-393    59-106    59-122    128-400
MSHELP   12-117    134-60
MSHW1    38-70     128-220
MSHW2    38-74     128-230
MSPASS   59 94     59-110    59-126    128-380
MSPRM    63-6      128 620
MSTOTL   59-100    59-116    59-132    128-390
```

CROSS REFERENCE TABLE (CREF V04 00)

```
MXWAIT   96 25    96 370
MYMRO    62 92   128 520
MYY       5 1360
MXRNDM  101 100  106 70
N       128 490
NED       5 1390
NEDCLK   58 30   128 480
NEM       5 1380
NEWASN   62 51    65 30
NEWINT    9 00    17 35•   19-7     19 20    19-27    19 28•   65 80•
NXNLEV   16 64    63 98   128 180
NXXLLT   12 132   15 55   128 190
NXXRVS   20 16   128 600
NXXTCM   34 50    37 6
NXNE    128 580
NXXP      6 30
NOTAVL  128 150
NOTPRS   16-31    63 93   128 140
NOTRP    16 28    63 91   128 130
NOTSXF   16 37    63-85   128 160
NSA       5 2250
OFFDIR    5 2400
OFFON     5 1610
OFFSET    9-0      9-0      9-0     26 16    26 37    43 37    57 344
OFLIN    22 60    24 30
ONES     10 0     10 00
ONESEC    9-00    12 89•   20 10•   61-31•   61 33•
ONESUM   18 43    59-450   73-1
OPI       5 1900
OPIER    24-67    30-580
OPILR1   30 690
OPT     104-34   105 80   108 34   109 87   109 139
OPTBL     9-00    57 44    57 46
OR        5-1340
ORDERR    9 00    12-83    18-5     20-31    20 42    20-69    21-3     96-31
OVRWRT   15 38   128 840
PACK     63 81    65-30    65-8•    65-13•   65-19•   65 220
PAR       5-1800
PAR1     129-3   129-260
PAR11    129-8   129-380
PAR12    129-5   129-390
PAR14    129-9   129-400
PAR15    129-11  129-410
PAR16    129-10  129-420
PAR19    129-6   129-430
PAR2     129-4   129-270
PAR21    129-15  129-470
PAR3     129-7   129-280
PAR4     129-290  130-20   130-20   130-20   130-20   130-20   130-20   130-20   130-20   130-20
PAR5     129-300  130-20   130-20   130-20   130-20   130-20   130-20   130-20   130-20   130-20
PAR6     129-310  130-20   130-20   130-20   130-20   130-20   130-20   130-20   130-20   130-20
PAR7     129-320  130-20   130-20   130-20   130-20   130-20   130-20   130-20   130-20   130 20
PAR8     129-330  130-20   130-20   130-20   130-20   130-20   130-20   130-20   130-20   130-20
PAR9     129 340  130-20   130-20   130-20   130-20   130-20   130-20   130-20   130-20   130-20
PARENT   15-109   67-44    70-80
PARER    24-71    31 30
```

CROSS REFERENCE TABLE (CREF VOM 00)

```
PARMS1   14 220    15 108   129 30
PA.*.**   9 0      71-45
PA'SE'    9 00     73-1     129-6
PAT       5 1510
PATCH   133 720
PATTLB    9 00     39 21    53 36    53 38    129 7
PERIOD   36 33     38 00    40 36    41 11    41 31    57 305   57 392   57-417   57 462   59 87    59 99    59 105   59 115   59 121
         59 131    59 137   59 151   59 156   59 157   59 160   59 167   81 17    120 510
PFRM     71 200   108 36   108 460  108 530
PFICH    87 1     87 10
PFICH1   87 1     87 10
PFICH2   87 1     87 10
PFICH3   87 1     87-10
PFICH4   87-1     87 10
PFISTN   87-1     87 10    87-16
PGE       5 1570   5-2500
PGM       5 1670
PIP       5-1710
PINB      5 1040
PINOVE    5 1040
POPULE  105 21   106 12   106-153  109 77   109 126  119 260
POSLR    30 14    30 440
PR0       5 1040
PR1       5 1040
PR2       5 1040
PR3       5 1040
PR4       5 1040   62 13
PR5       5 1040
PR6       5 1040   12-28    58 12    58 24
PR7       5 1040
PROCES   21 11    22 30
PR'HAD   26 5     26 51    33 9     42 160
PRTIM    22 62    24 140
PS        5 104     5 1040   12-88•   62 13•   62 99•   73 5•    81 28•   96 14•   102 15   102 16•   102 47•   104 11   104 12•   104 68•
        105-9    105-56•
PSEL      5 1140
PSW       5 1040
PUNSAV   22 52    23-30
PUSH'LG  15 3     16-12    17 37    17-51•   66-10    96 1•    96 30
PUSHUP   96-1     96-110
PUSVEC    5 1040   12-25•   12-25•   96 1•    96-1•    96 1•    96 1•    96 1•    96 1•
QCNT    115 250  117-19   118-7•   118 22   118 24•  119 11   119 26•
QDRV0   116 3    116-14   116 23   117 30
QDRV1   116 4    116-15   116 24   117-40
QDRV2   116 5    116-16   116-25   117 50
QDRV3   116-6    116-17   116 26   117-60
QDRV4   116 7    116-18   116-27   117-70
QDRV5   116-8    116 19   116-28   117 80
QDRV6   116-9    116-20   116-29   117-90
QDRV7   116 10   116 21   116-30   117-100
QINPT   116-30   118-9    118-26•  118 27•  118-28   118 30•
QOUTPT  116 140  118 9•   119-14   119 28   119 29•  119-30•  119-31   119-33•
QSTART  116 230  117-25   117-30   118 30   119 33
QSTOP   116 240  118 28   119 31
QTERP   116-31   117 110
QUES     70-21    71 8     128-30
```

CROSS REFERENCE TABLE (CONT'D)

[illegible]	5 1060	12 25	12 25•	12 25•										
[illegible]	5 1060													
[illegible]	51 30	52 30	52 60											
[illegible]	9 00	51 29	129 15											
[illegible]	9 00	51 6	129 8											
[illegible]	51 250													
[illegible]	52 250													
[illegible]	51 45	51 30	51 11											
[illegible]	52 140													
[illegible]	51 27	51 300												
[illegible]	9 00	51 16	129 9											
[illegible]	103 37	103 62	103 68	106 108	106 125	106 130	108 10	109 5	111 15	111 60	112 100	110 22		
[illegible]	85 1	97 10												
[illegible]	6 250	9 0	50 58	56 61										
[illegible]	6 240	9 0	44 14											
[illegible]	12 110	15 20	15 41	15 92	62 50	63 7	70 29	91 10	132 29	132 60				
[illegible]	9 00	12 1050	12 1060	15 58	15 61	50 61	51 9	62 64	62 70					
[illegible]	5 1110													
[illegible]	28 5	29 6	30 6	43 16	56 18	57 191	57 4700							
[illegible]	28 6	29 7	30 7	44 100										
[illegible]	6 120													
[illegible]	6 50	43 9	43 30											
[illegible]	43 300	63 61												
[illegible]	50 53	43 70												
[illegible]	62 73	65 80												
[illegible]	21-41	47-70												
[illegible]	6-70													
[illegible]	50 65	70 150												
[illegible]	48 23	62 96	76-48	75-37	94 1	95 1	97-10	102 48	104-61	108-65	108-97	108-250	107 10	110 21
	114-39	117-34												
[illegible]	5-1060													
[illegible]	9-00	25 26•	26 15•	26 34•	26 35	26 35	27 12•	27 22•	27-45•	27-61•	27-65•	27-66	27 66	28 19•
	29 26•	30 32•	30 69•	31 10•	32 26•	33 10•	33 28•	33 56•	44-38•	44-58•	44-59	44-59	57 359	
[illegible]	9 00	12 30												
[illegible]	13 4•	13-13•	13-17•	13-23	101-60	114-36								
[illegible]	5-1790													
RPOT	43 10	43-31	44-16	49-9	104-110									
RPADR	12-80•	62 14	101-40	102 36	104 18	106 15	106 61	106 86	107-5	111-13	112-11	112-14	112-18	113 15
	113-17	113 22	113 24	113 28	114 10	115-9	132 48							
RPAS	101-210	103-67•	108-24•	108-47•	109 3	109-50•	109 82•	109 113•	111 55					
RPBA	101-160													
RPBAE	101-340													
RPCC	101-290	104-43•	105-49•											
RPCS1	101-140	103-24•	103 31	103-56	105-16•	105-17	106-48	106-81	106-139	106 151	106 168	106 179	108 15	109 6
	109-48	109-129	111-65											
RPCS2	62-15•	101-180	102-37•	103 23•	103 25	104 28•	105-15•	106-16•	106-62•	106 87•	106-203	106 227•	108-13•	109-61•
	109-90•	109-112•	111-14•	111-30•	112-15	113-25	114-11•	114-17	115 10•	115 15				
RPCS3	101-350													
RPDA	101-170	106-32	106-76	106-93										
RPDB	101-230	114 15												
RPDC	101-280	106 36	106-65	106-101										
RPDS	101-190	103 63	108-44	109-94	111-16									
RPDT	101-250	103-38												
RPEC1	101 320													
RPEC2	101-330	114 26												
RPER1	29-8	30 8	101-200	103-69	109-95	113-29								

```
ARMRB2   101 300  109 96
ARMRB3   101 310  109 97
ARPINIT   14 5    102 140
ARPLA    101 220
ARPMB1   101 240  106 22   106 57
ARPOF    101 270  103 60   106 109  106 113  106 120  106 120
ARPSAV   101 260
ARPSTUB   98 160  102 18   109 90-  109 95-  109 96-  109 97-
ARPSTUI   98 160
ARPSTUP   98 160
ARPSTUS   98 160
ARPSTUW   98 160
ARPSTUX   98 160
ARPSTUZ   98 160
ARPSTUA   98 160
ARPSTUY   98 160
ARPTMP    61 52   110-60
ARPVEC    12 81-  101-50   102-33   102 35   104 12
ARPWC    101 150
ATC        6 110
ATCRTN    73 1     73-30
AWAIT      5 2550
AWAIT      5 2560
AWAIT      5 2570
$         57 195   57-204   127-100
SAVFPG    16 90-   100 70   106 155  108 19   109 79
SAVIR1     9 00     36 22-   37 36-   38 11    38 17    38 22-
SAVIR5     9 00     36 23-   37-37-   38 18    38 23-
SAVPUS     9 00     37 38-   38 19
SAVREG    48 9      62-12    74-38    75-26    90-1     95 1     97-10    102 14   104-14   105-0    106-190  107-4    110 9    114 7
          117-18
SC        107 9    109-30
SC04        5-2290
SC1         5 2270
SC10        5-2300
SC100       5-2350
SC11      109-62   109-730
SC12      109-36   109-42   109-900
SC13      109-30   109-1120
SC2         5-2280
SC20        5-2310
SC3       109-200  109-24
SC4       109-220  109-55   109-71   109 80   109 140
SC40        5-2320
SC5       109 21   109-280
SC6       109-41   109-570
SC8       109-49   109-640  109-105
SCMND      62-59    64-280
SCOPE       5-1040
SEARCH      6-130
SECLMT    37-58    55-19    57-492   57-497   67-15    67-41    67-42    68 12-   68 19-   121-130
SECOND     9-00    12-92-   20-13-   61-23    61-34-   61-35    61 37-   96-16-
SEEK       6-40
SEEKFG    100 150  102-19   105-37
SELDRV     6-170
SEQPAR    17-41    20-54    50-140
SET.IE    103-27   104-46   105-55   106-233  109-13   109-136  115-80
```

CROSS REFERENCE TABLE (CREF V04 00)

SETFMT	6 160												
SETVEC	15 4	15 89	15 97	15-103	15 111	16 40							
SIZE70	13 40												
SIZMEM	14 30	96 35											
SKI	5 2730												
SKIER	24 105	32 380											
SLASH	129 240												
SPOTCK	25 3	28 3	29-12	30 18	30-58	30-81	42-70						
SRCMNT	99-390	105-10•	106-83•	109-74	109-83•								
STA	17 30												
STACK	5-1040	12-25	12-87	73 4	96-15								
START	7 3	12-150	58-35	62-36									
START1	7-1	12-11	12-180										
START2	12-16	12-210											
STATIN	9-00	12-82•	21-49	21-51•	61-57•								
STATIS	21-10	45-70											
STATPR	21-55	59-70	64-42										
STKLMT	5-1040												
STNDAT	10-00	39-29	39-35	48-15									
STO	110-16	111-110											
SUPR1	77-15	77-200											
SUPR2	77-14•	77-19•	77-29	77-31•	77-33•								
SUPRS	59-98	59-104	59-114	59-120	59-130	59-136	77-170						
SUPRSL	70-18	77-120											
SVRHXX	16-61	69-12	106-157	108-21	108-41	109-35	109-81	109-125	111-29	111-95	114-70		
SW0	5-1040	12-104	51-12	62-67	62-81								
SW00	5-104	5-1040											
SW01	5-104	5-1040	22-20										
SW02	5-104	5-1040	15-15	16-14	21-43								
SW03	5-104	5-1040											
SW04	5-104	5-1040	50-39	71-41	73-1								
SW05	5-104	5-1040	57-73	57-144									
SW06	5-104	5-1040	56-26										
SW07	5-104	5-1040	38-52										
SW08	5-104	5-1040	73-1										
SW09	5-104	5-1040											
SW1	5-1040												
SW10	5-1040	57-6											
SW11	5-1040												
SW12	5-1040												
SW13	5-1040	57-9	86-1										
SW14	5-1040												
SW15	5-1040	23-29	57-395	57-425									
SW2	5-1040												
SW3	5-1040	37-16	42-16										
SW4	5-1040												
SW5	5-1040	57-177											
SW6	5-1040												
SW7	5-1040												
SW8	5-1040												
SW9	5-1040	17-6											
SWR	8-0#	12-25	12-25	12-25•	12-25•	12-25•	12-31	12-104	15-15	16-14	17-6	21-43	22-20	23-29
	37-16	38-52	42-16	50-39	51-12	56-26	57-6	57-9	57-73	57-144	57-177	57-395	57-425	62-25
	62-67	62-81	71-41	73-1	73-1	81-26	85-1	85-1	85-1•	86-1	86-1	86-1	86-1	86-1
	96-1	96-1•												
SWREG	7-1#	12-25	12-31	85-1	85-1	85-1								

CROSS REFERENCE TABLE (CREF V04.00)

```
SWTIM    22 58    23-36#
T        57-193   57-200  127-8#
TAB     128-7#
TAB.XY    7-12#    8-0
TABLE    67-36   130-5#
TABLE0  130-5    130-20#
TABLE1  130-8    130-20#
TABLE2  130-8    130-20#
TABLE3  130-8    130-20#
TABLE4  130-8    130-20#
TABLE5  130-8    130-20#
TABLE6  130-8    130-20#
TABLE7  130-8    130-20#
TAP       5-222#
TBITVE    5-104#
TCF       5-270#
TD      107-8    108-3#
THEAD    51-29#
TIMER   100-20#  103-18*  103-83*  105-48*  106-161* 106-209* 108-8*   109-58*  109-85*  110-12   110-14*  111-45*  111-73*  111-81*
        111-88*
TKVEC     5-104#  85-1*    85-1*
TPE       5-263#
TPVEC     5-104#
TRAPVE    5-104#  12-25*   12-25*
TRE       5-116#
TRFER    24-51    32-18#
TRKLMT   37-62    54-47    55-23    55-50    57-498   67-14    67-39    67-40    68-11*   68-18*   121-14#
TRMREP   21-56   128-28#
TRNSWT   99-31#  106-13*  106-184  106-186* 106-197  106-219* 106-225* 108-10   108-11*  111-24   111-39   111-51*
TRTVEC    5-104#
TST1     12-19#
TSTANY    9-0#    15-7*    15-57*   67-17    67-66
TYDRV    59-81    81-11#
TYPDRV   19-21    63-77    81-12#
TYPDS    59-150   59-153   59-156   59-159   59-166   97-1#
TYPE     12-6     12-30    12-31    12-45    12-51    12-56    12-62    12-117   12-130   12-132   12-134   15-19    15-32    15-38
         15-53    15-55    15-59    15-63    15-91    15-104   16-17    16-18    16-20    16-28    16-31    16-34    16-37    16-44
         16-46    16-47    16-55    16-64    16-69    18-39    18-41    18-42    19-15    19-16    19-18    19-19    19-22    20-16
         21-54    21-56    23-14    23-15    23-16    23-21    23-22    23-23    23-25    23-26    57-8     57-11    57-12    57-14
         58-30    59-22    59-28    59-30    59-31    59-50    59-68    59-70    59-71    59-72    59-77    59-79    59-80    59-82
         59-87    59-93    59-94    59-99    59-100   59-105   59-106   59-109   59-110   59-115   59-116   59-121   59-122   59-125
         59-126   59-131   59-132   59-137   59-148   59-149   59-151   59-152   59-154   59-155   59-157   59-158   59-160   59-161
         59-167   59-168   61-17    61-22    62-20    62-22    62-23    62-24    62-28    62-92    62-94    63-6     63-19    63-72
         63-73    63-75    63-76    64-43    64-44    64-45    67-43    70-12    70-20    70-21    70-22    70-30    71-7     71-8
         71-9     71-11    71-30    71-31    71-32    71-33    71-36    73-1     73-1     73-1     73-1     73-1     73-1     73-1
         73-1     73-1     73-1     73-1     77-32    79-32    81-11    81-12    81-15    81-17    85-1     85-1     85-1     85-1
         85-1     85-1     85-1     85-1     85-1     85-1     85-1     85-1     85-1     85-1     85-1     85-1     85-1     85-1
         86-1     86-1     86-1     86-1     87-1     87-1     87-1     87-1     87-1     87-1     87-1     88-1     89-1     90-1
         96-1     96-21    96-25    97-1#    132-21   132-24   132-27   132-36   132-39
TYPOC    12-8     62-25    85-1     87-1     87-1     97-1#    132-26   132-38
TYPON    97-1#
TYPOS    12-48    12-59    16-19    19-17    23-27    59-69    63-74    71-10    71-35    97-1#
TYPRI4   80-41
TYPSUM   59-32    59-49    59-64#
UCPAR    22-54    23-14#
UNS       5-191#
```

CROSS REFERENCE TABLE (CREF V04.00)

```
UNSPC    24 35    24-39    33-22#
UNTASN   63-25    63-39   128-12#
UNTNOT   64 21    64-37   128-11#
UNTOFF   16 34    63-95   128-9#
UNTON    16 46   128-10#
UPE       5-140#
US1       5-127#
US2       5 128#
US4       5-129#
VV        5-164#
WAIT      9-0#    18-30    20-19    20-21    20-35    20-78
WATPAK   62-90    65-28#
WC.HK   107-16#  108-27
WCE       5-141#
WCF       5-182#
WCFER    24-75    33-3#
WCHKX    98-2#   108-26
WCKD      6-19#    9-0     56-28    56-37   107-28
WCKER    24-47    27-3#
WCKHD     6-20#
WLE       5-188#
WLEER    24-83    31-31#
WOR       5-254#
WRDCNT    9-0#    14-19    14-21*   14-22    15-110   15-112*  50-44    53-3     129-3
WRDPOS    9-0#    34-28*   34-29    34-38    36-40*   36-44*   37-3*    37-11*   37-38    37-76*   37-80*   38-16    38-19*   38-24*
         38-68    38-72    38-76    40-33*   41-7     41-26
WRL       5-169#
WRT.RP  103-55   103-59   106-21   106-31   106-35   106-47   106-56   106-64   106-75   106-80   106-92   106-100  106-112  106-127
        106-150  106-167  106-178  109-47   113-10#
WRTCHK    9-0#   107-16   129-11
WRTDAT    6-21#    9-0     50-67    56-31   107-18
WRYUNS    5-253#
XXDP      9-0#    12-36*   12-39*   12-40    12-42*   12-47    12-58    16 40    16-42    17-24    17-26    63-26    63-28    63-40
         63-42
Y       128-50#
ZEROS     10-0    10-0#
ZROIND    9-0#    36-21*   37-28*   37-35*   37-40
```

CROSS REFERENCE TABLE (CREF V04.00)

```
$$CMRE    7 600#
$$CMTM    7 600#    8 0
$$ESCA    5 104#
$$NEWT    5 104#   12 19
$$SET    97-1      97-1      97 1     97-1     97 1     97 1     97 1     97-1     97 1     97 1     97 1     97-1#    97 2
$$SETM   12-25     12-25#
$$SKIP    5-104#
.$ACT1    5-76#     7-8
.$APTB    5-79#     8-0       8 0#
.$APTH    5-79#     7 11
.$APTY    5-79#    91-1
.$CATC    5-77#     7-1
.$CMTA    5-77#     7-600
.$D82D    5-78#    94-1
.$D82O    5-78#    95-1
.$EOP     5-79#    73-1
.$ERRO    5-77#    86-1
.$ERRT    5-77#    87-1
.$POWE    5-79#    96-1
.$RAND    5-78#    92-1
.$READ    5-76#    85-1
.$SAVE    5-78#    93-1
.$SIZE    5-78#   131-1
.$TRAP    5-78#    97-1
.$TYPD    5-77#    90-1
.$TYPE    5-76#    88-1
.$TYPO    5-77#    89-1
.EQUAT    5-76#     5-104
.HEADE    5-76#     5-85
.SETUP    5-76#     6-29
.SWRHI    5-76#     5-87
.SWRLO    5-76#     5-87#     5-88     5-89     5-90     5-92     5-94     5-99     5-101    5-102
A        60-2#     60-16     60-17    60-18    60-19    60-20
CKCHR     5-17#    83-20     83-28
CKDIG     5-27#    70-27
CKNUM     5-40#
COMMEN    5-104#
ENDCOM    5-104#
ERENTR   85-13#    86-1
ERROR     5-104#  109-12
ESCAPE    5-104#
GETPRI    5-104#  131-1
GETSWR    5-104#   12-31     12-31#
MORETA    7-14#     8-0
MULT      5-104#
NEWTST    5-104#   12-19
POP       5-104#   30-15     30-48    38-24    57-102   59-36    64-47    66-46    67-81    68-20    69-29    79-34    80-36    90-1
         91-1      91-1      92-1     93-1     96-1     96-1
PUSH      5-104#   30-10     38-16    57-28    58-6     59-7     64-28    66-9     68-7     69-10    90-1     91-1     91-1     91-1
         92-1      93-1      96-1     96-1
REPORT    5-104#
SETPRI    5-104#   85-1
SETTRA   97-1      97-1      97-1     97-1     97-1     97-1     97-1     97-1     97-1     97-1     97-1     97-1#    97-2
SETUP     5-104#   12-25
SKIP      5-104#
SLASH     5 104#
```

CROSS REFERENCE TABLE (CREF V04.00)

STARS	5 1040	7 8	7-11	7 11	7-11	8 0	8 0	8-0	12-19	73-1	77-4	78 4	85-1	85 1
	85 1	85 1	85-1	86 1	87-1	88 1	89-1	90 1	91-1	92 1	93-1	94-1	95-1	96 1
	96 1	97-1	131-1											
SWRSU	5 1040	12-25	12-250											
TRMTRP	97 10	97-3												
TYPBIN	5-1040													
TYPDEC	5-1040	59-150	59-153	59-156	59-159									
TYPNAM	5-1040	12-31												
TYPNUM	5-1040													
TYPOCS	5 1040	16-19	19-17	23-27	59-69	63-74	71 10	71-35						
TYPOCT	5-1040	62-25	85-1	87-1	87-1									
TYPTXT	5-1040	12-6	12-45	12-56	12-62	59 82	59-93	59 109	59-125	59-148	59-149	59-152	59-155	59-158
	59 161													
UPCODE	95-20	96-1												
XXEP	72-110	73-1												