RP04/05/06

MULTI-DRIVE EXERCISER

MD-11-DZRJD-B

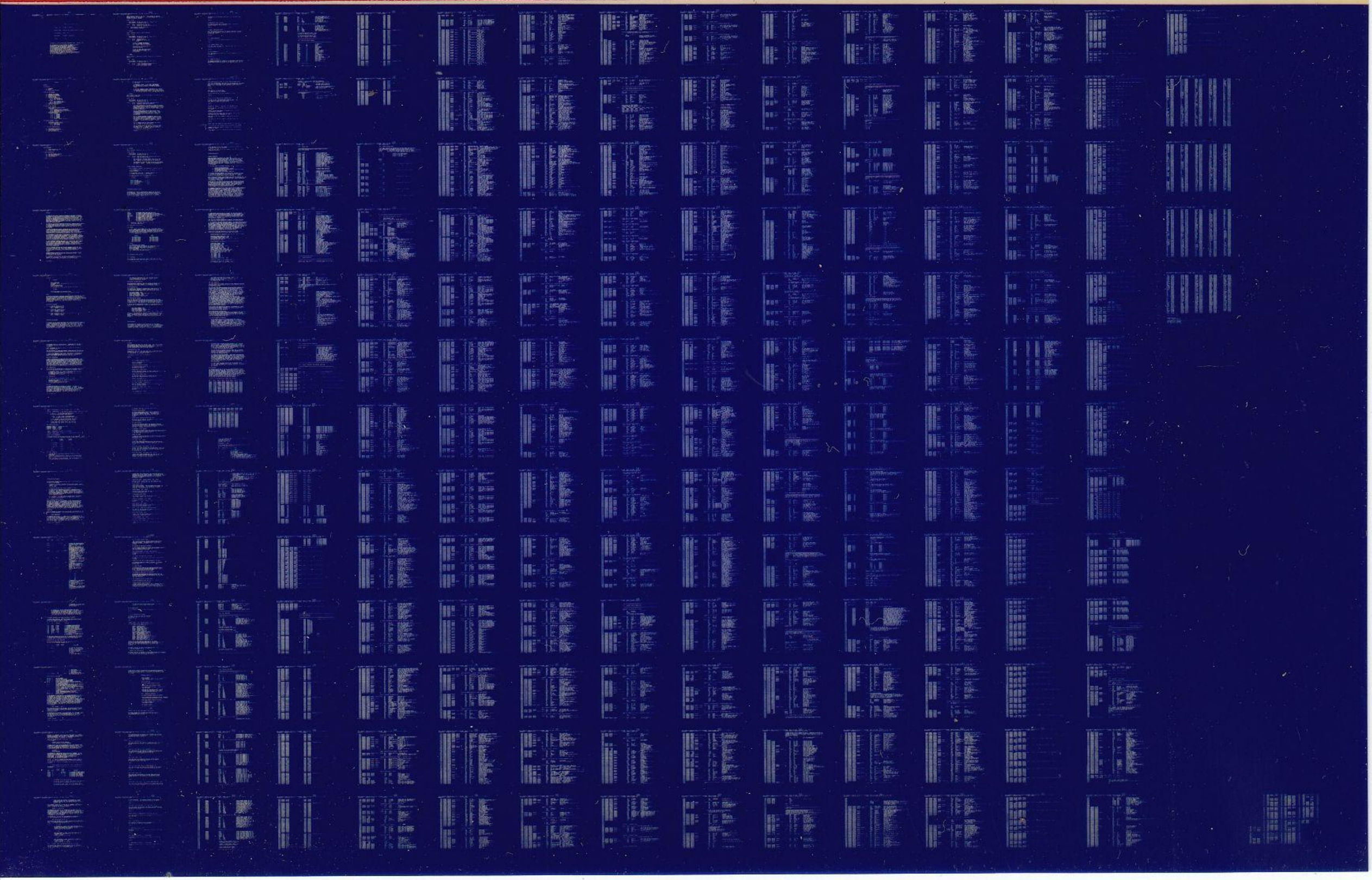
EP-DZRJD-B-DL-A

FICHE 1 OF 1

AUG 1977

digital

MADE IN USA



EOF1DZKB0RSB04111 [illegible]RJD-B,[illegible]RIVE EXERCISE[illegible]M411 MACY11 2[illegible]RJD[illegible]AY-77 11:01 [illegible] 770720
DZRJD8.P11 24-MAY-77 11:04

.REM @

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

PRODUCT CODE: MAINDEC-11-DZRJD-B-D

PRODUCT NAME: RP04/5/6 MULTI-DRIVE EXERCISER PROGRAM

DATE CREATED: APRIL, 1977

MAINTAINER: DIAGNOSTIC ENGINEERING

AUTHOR: C. HESS

CONTENTS

1. ABSTRACT

THE RP04/5/6 MULTIDRIVE EXERCISER PROGRAM IS DESIGNED TO PERFORM AN INTERACTIVE TEST ON RP04/5/6 DISK DRIVES CONNECTED TO A MASSBUS SUBSYSTEM. THE SUBSYSTEM MAY BE COMPOSED OF INTERMIXED RP04, RP05 OR RP06 DISK DRIVES CONTROLLED BY EITHER AN RH11 OR AN RH70. 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 SPECIFICATIONS.

THE RP04/5/6 MULTIDRIVE 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 NON DUAL PORT DRIVES.

TO OBTAIN INTERACTIVE TESTING, OPERATIONS ON THE DRIVES ARE OVERLAPPED (OTHER DRIVES ARE PERFORMING SEEK/SEARCH OPERATIONS WHILE ONE DRIVE IS PERFORMING A DATA TRANSFER OR WRITE CHECK OPERATION). 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 ARE USED (I.E., WRITE DATA, WRITE HEADER & DATA, READ DATA, AND READ HEADER & DATA) AS WELL AS WRITE CHECK DATA AND WRITE CHECK HEADER & DATA COMMANDS. RECALIBRATE AND READ-IN PRESET COMMANDS ARE USED AT STARTUP AND DRIVE INITIALIZATION. RECALIBRATE, OFFSET, AND RETURN-TO-CENTERLINE COMMANDS ARE USED DURING ERROR PROCESSING.

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.

PROGRAM/OPERATOR COMMUNICATIONS ARE THROUGH THE KEYBOARD; DYNAMIC PROGRAM OPTIONS ARE SELECTED BY SWITCH REGISTER SETTINGS. ERRORS ARE REPORTED ON THE TELETYPE.

ALL COMMANDS, DATA PATTERNS, AND DATA BUFFER SIZES ARE SELECTED RANDOMLY BY THE PROGRAM. ADDITIONALLY THE ADDRESSES (EG, CYLINDER, TRACK, AND SECTOR) FOR EACH OPERATION ARE SELECTED RANDOMLY.

2. REQUIREMENTS

2.1 EQUIPMENT

REQUIRED

PDP-11 PROCESSOR
16K MEMORY (20K IF THE PROGRAM IS INCLUDED IN AN 'XXDP' CHAIN
TELETYPE
PROGRAM LOADING DEVICE
KW11-L OR KW11-P CLOCK
RH11 OR RH70 WITH 1 RP04, RP05, OR RP06 DISK DRIVE

OPTIONAL

ADDITIONAL MEMORY TO A MAXIMUM OF 28K
1 TO 7 ADDITIONAL RP04/5/6'S ON THE SAME RH11 OR RH70

2.2 MEDIA

THE RP04/5/6 MULTIDRIVE EXERCISER PROGRAM REQUIRES FORMATTED DISK PACKS WHICH CONTAIN RANDOM OR PATTERNED DATA RECOGNIZED BY THE EXERCISER. DISK PACKS USED BY THE PROGRAM MAY BE GENERATED BY THE RP04/5/6 FORMATTER PROGRAM (MAINDEC-11-DZRJB) OR BY THE 'W' COMMAND OF THE RP04/5/6 MULTIDRIVE EXERCISER (SEE SECTION 4.4). THE PACKS MUST BE FORMATTED IN 22 SECTOR (16 BIT) MODE; THE ALTERNATE (20 SECTOR - 18 BIT) MODE IS NOT SUPPORTED.

2.3 PRELIMINARY PROGRAMS

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

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

RP04/5/6 DUAL CONTROLLER LOGIC TEST (FOR DUAL PORT DRIVE TESTING)
PART 1 (MAINDEC-11-DZRJE)
PART 2 (MAINDEC-11-DZRJF)

3. OPERATING THE PROGRAM

3.1 THE PROGRAM MAY BE LOADED FROM PAPER TAPE USING THE ABSOLUTE LOADER OR IT MAY BE LOADED FROM THE APPROPRIATE 'XXDP' MEDIA USING THE ASSOCIATED LOADER. THE PROGRAM MAY BE INCLUDED IN AN 'XXDP' CHAIN. IF THE PROGRAM IS BEING RUN ON A PROCESSOR WITH 16K, THE 'XXDP' LOADER WILL NOT BE PRESERVED. THE PROGRAM MUST BE RUN ON A SYSTEM WITH 20K OR MORE TO PRESERVE THE 'XXDP' LOADER. THE 'ABSOLUTE' LOADER WILL BE PRESERVED IN A 16K SYSTEM, HOWEVER.

3.2 THE PROGRAM STARTS AT LOCATION 200(8). PARAMETERS NOT INCLUDED IN THE TELETYPE DIAGLOGUE GROUP MUST BE CHANGED BEFORE THE PROGRAM IS STARTED.

3.3 START THE PROGRAM AT LOCATION 204(8) IF THE RH11 OR THE RH70 IS NOT AT ADDRESS 176700.

3.4 ONCE THE PROGRAM IS LOADED AND STARTED, OPERATIONS ARE INITIATED AND CONTROLLED BY KEYBOARD COMMANDS AND SWITCH REGISTER SWITCH SETTINGS.

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

3.5 PASS/TEST TERMINATION

A PASS 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 SPECIFICATIONS FOR RP04'S, RP05'S, AND RP06'S SPECIFY NO MORE THAN 1 SOFT ERROR (NON-PACK RELATED) IN 1 X 10↑9 BITS READ OR NO MORE THAN 1 SEEK ERROR IN 1 X 10↑6 SEEKS. THE NUMBER OF BITS OR SEEKS DETERMINING A PASS WERE SELECTED TO PROVIDE A 90% CONFIDENCE LEVEL THAT THE DRIVE IS PERFORMING TO THE APPLICABLE SPECIFICATION.

3.5.1 PASS TERMINATION

END OF PASS MAY BE DETERMINED BY EITHER OF THE FOLLOWING CONDITIONS. THE END OF PASS CONDITION USED IS DETERMINED BY PARAMETER 'ENDET'

A. IF PARAMETER 'ENDET' IS 1, END OF PASS OCCURS WHEN THE DRIVE HAS READ 1.875 X 10↑8 WORDS (3 X 10↑9 BITS).
B. IF PARAMETER 'ENDET' IS 0, END OF PASS OCCURS WHEN THE DRIVE HAS PERFORMED 3 X 10↑6 SEEKS.

3.5.2 TEST TERMINATION

THE TEST FOR A DRIVE IS TERMINATED (SW<04> = 0) WHEN:

A. THE DRIVE HAS COMPLETED THE NUMBER OF PASSES SPECIFIED IN PARAMETER 'PASCNT'.
B. THE TOTAL ERRORS ACCUMULATED EXCEED 100.
C. A FATAL ERROR OCCURS: EM12 OR EM14.

3.6 RUN TIME

THE EXERCISER PROGRAM MAY BE RUN IN TWO MODES. THE MODE IS DETERMINED BY THE VALUE IN PARAMETER 'MAXDL'. IF 'MAXDL' IS ONE SECTOR, THE PROGRAM RUNS IN A SEEK HEAVY MODE; IF 'MAXDL' APPROACHES ONE TRACK IN SIZE (5720 DECIMAL) THE PROGRAM RUNS IN A DATA TRANSFER HEAVY MODE. THE PROGRAM RUN TIME VARIES GREATLY DEPENDING ON THE OPERATION MODE SELECTED, THE MEMORY AVAILABLE OVER 16K, THE READ/WRITE RATIO PARAMETER - 'RATIO', AND BY SWITCHES 0, 1, & 2.

3.6.1 DATA TRANSFER MODE

1 DRIVE - APPROXIMATELY 2.5 HRS (TO REACH 1.875 X 10↑8 WORDS)
TO
8 DRIVES - APPROXIMATELY 11 HRS (FOR ALL DRIVES TO REACH 1.875 X 10↑8 WORDS)

NOTE: IF SW<01> = 1 (NO SOFTWARE DATA COMPARSIONS), THE RUN TIMES ARE THE FOLLOWING VALUES, APPROXIMATELY:

1 DRIVE - 1.7 HRS (1.875 X 10↑8 WORDS READ)
ADD 1/2 HOUR FOR EACH ADDITIONAL DRIVE TESTED.

IF THE PROGRAM IS RUN WITH BOTH SW<00> AND SW<01> SET, THE RUN TIMES SHOULD BE ABOUT 20% FASTER.

NOTE: ON THE FIRST PASS (QUICK VERIFY) THE TIMES ARE APPROXIMATELY ONE EIGHTH OF THE ABOVE VALUES.

3.6.2 SEEK VERIFICATION MODE

PARAMETER 'MAXDL' = 1 SECTOR (256 WORDS)
PARAMETER 'MAXTRK' = 'MINTRK'
PARAMETER 'MAXSEC' = 'MINSEC'
SW<00> = 1 (READ ONLY MODE)

1 DRIVE - APPROXIMATELY 25 HRS (3 X 10↑6 SEEKS)
TO
8 DRIVES - APPROXIMATELY 40 HRS (3 X 10↑6 SEEKS FOR ALL DRIVES)

3.7 UNIBUS & VECTOR ADDRESSES

THE PROGRAM ASSUMES THE FOLLOWING UNIBUS AND VECTOR ADDRESSES. (REFER TO SECTION 4.2.2 FOR THE LOCATIONS AT WHICH TO CHANGE THESE ADDRESSES.)

UNIT	UNIBUS ADDRESS	VECTOR ADDRESS
RH11 OR RH70	176700	254
TTY PRINTER	177564	NOT USED
TTY KEYBOARD	177560	60
KW11-L	177546	100
KW11-P	172542	104

3.8 DUAL PORT OPERATION

A. LOAD THE RP04/5/6 MULTIDRIVE 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; CYCLE THE DRIVES UP.
C. START THE PROGRAM IN EACH PROCESSOR. RUN THE PROGRAM AS THROUGH EACH PROCESSOR WERE RUNNING INDEPENDENTLY OF THE OTHER.

4. CONTROLLING THE PROGRAM

THE FOLLOWING KEYBOARD CONVENTIONS ARE USED BY THE KEYBOARD ENTRY ROUTINES IN THE PROGRAM:

A. TO DELETE AN INCORRECT CHARACTER FROM AN ENTRY STRING, TYPE A 'RUBOUT' ('RO'). TYPING A 'RO' WILL DELETE SUCESSIVE CHARACTERS FROM THE INPUT.

B. TO DELETE AN ENTIRE LINE, TYPE A 'CONTROL U' ('↑U').

C. AN ENTRY MUST BE TERMINATED BY EITHER A 'CARRIAGE RETURN' OR A 'PERIOD'. THE 'PERIOD' TERMINATION IS RECOGNIZED BY THE PROGRAM AS A DEFAULT ENTRY REQUEST. WHEN A LINE IS TERMINATED BY A 'PERIOD' INSTEAD OF A 'CARRIAGE RETURN', THE PROGRAM WILL ACCEPT THE ENTERED VALUE AND WILL DEFAULT TO THE PRELOADED VALUES FOR ANY REMAINING ENTRIES.

D. IF A 'CONTROL C' IS TYPED DURING KEYBOARD ENTRY, THE PROGRAM WILL RETURN TO THE BEGINNING OF THE GROUP BEING ENTERED.

4.1 DATE & OPERATOR IDENTIFICATION

WHEN THE PROGRAM IS INITIALLY STARTED, IT WILL ASK FOR DATE AND OPERATOR I.D. ENTRIES. (THE REQUEST FOR THESE ENTRIES OCCURS ONLY WHEN THE PROGRAM IS FIRST STARTED AND WILL NOT APPEAR WHEN THE PROGRAM IS RESTARTED.) THESE ENTRIES ARE OPTIONAL AND MAY BE BYPASSED BY ENTERING A 'CARRIAGE RETURN' IN RESPONSE TO THE REQUEST. THE PROGRAM DOES NOT EDIT OR CHECK EITHER ENTRY. UP TO 8 CHARACTERS OF DATE INFORMATION AND UP TO 6 CHARACTERS OF OPERATOR IDENTIFICATION MAY BE ENTERED. BOTH THE DATE AND THE OPERATOR I.D. WILL BE TYPED WHEN THE 'SA' COMMAND IS PERFORMED (SEE SECTION 4.4.3).

4.2 PARAMETERS

WHEN THE PROGRAM IS STARTED FROM LOCATION 200, THE OPERATOR WILL BE ASKED TO ENTER PARAMETERS. THE FOLLOWING MESSAGE WILL BE DISPLAYED:

```
        ENTER PARAMETERS:
```

THE OPERATOR MUST ENTER A 'Y' TERMINATED BY A CARRIAGE RETURN <CR> IF PARAMETER ENTRIES ARE TO BE MADE. ANY OTHER CHARACTER IS ACCEPTED AS A 'NO' ENTRY. THE PROGRAM WILL IDENTIFY THE PARAMETER BY THE NAME GIVEN BELOW, DISPLAY THE CURRENT VALUE OF THE PARAMETER 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.

4.2.1 KEYBOARD ENTRY PARAMETERS

	DEFAULT	VALUE

NAME	BASE	VALUE	RANGE	FUNCTION
MAXDL	10	(SEE NOTE)		CONTROLS THE MAXIMUM BUFFER SIZE USED FOR DATA TRANSFERS
PASCNT	10	1	1 - 999	NUMBER OF PASSES TO END OF TEST.
INTRVL	10	5	0 - 256	DETERMINES THE INTERVAL (IN MINUTES) BETWEEN AUTOMATIC PERFORMANCE SUMMARY TYPEOUTS ERRORS PRINTED OUT IF SW(07)=0
CMPLMT	10	3	0 - 'MAXDL'	THE NUMBER OF DATA COMPARSION
WCSEL	8	000000	0 OR 1	IF PARAMETER = 0, THE DATA TRANSFER WORD COUNT IS RANDOMLY SELECTED BETWEEN 4 (10) AND THE VALUE IN 'MAXDL'. IF PARAMETER = 1, THE DATA TRANSFER WORD COUNT WILL BE THE VALUE IN 'MAXDL'
ENDET	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.
FORMAT	8	000001	0 OR 1	IF PARAMETER = 0; DO NOT PERFORM WRITE HEADER & DATA ORDERS; IF PARAMETER > 0, PERFORM WRITE HEADER & DATA ORDERS
RATIO	8	000003	0 - 7	CONTROLS THE APPROXIMATE RATIO OF READ TO WRITE ORDERS.

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

AUTOCK	8	000001	0 OR 1	IF PARAMETER = 1, THE PROGRAM PERFORM WRITE CHECKS AFTER EACH WRITE COMMAND. IF PARAMETER = 0, THE PROGRAM WILL PERFORM WRITE CHECKS RANDOMLY.
NOTPRT	8	000001	0 OR 1	IF PARAMETER = 1, DO NOT PRINT ERROR MESSAGES FOR DATA ERRORS OCCURING AT LOCATIONS DEFINED BY THE OPERATOR AS BAD PACK LOCATION.

IF PARAMETER = 0, PRINT ERROR MESSAGES ASSOCIATED WITH BAD PACK LOCATIONS.

NOTE: THE PROGRAM WILL SELECT A MAXIMUM BUFFER SIZE WHICH IS DETERMINED BY THE MEMORY AVAILABLE. THE MAXIMUM BUFFER SIZE ASSIGNED BY THE PROGRAM IS 5980 (10) WORDS. THE OPERATOR MAY SPECIFIY ANY OTHER MAXIMUM SIZE AS LONG AS THE VALUE SPECIFIED IS AT LEAST 4 WORDS BUT NO LARGER THAN THE INITIAL VALUE OF 'MAXDL' DETERMINED BY THE PROGRAM.

4.2.2 PERIPHERAL ADDRESSES AND OTHER LOCATIONS OF INTEREST

TO ALTER THESE LOCATIONS, THE OPERATOR MUST MAKE MANUAL ENTRIES BEFORE THE PROGRAM IS STARTED. THE KEYBOARD ENTRY ROUTINE DOES NOT PROVIDE ACCESS TO THESE LOCATIONS.

LOC	TAG	CONTENTS	FUNCTION
1144	$TKS	177560	TTY KEYBOARD STATUS REGISTER
1146	$TKB	177562	TTY KEYBOARD BUFFER REGISTER
1150	$TPS	177564	TTY PRINTER STATUS REGISTER
1152	$TPB	177566	TTY PRINTER BUFFER REGISTER
1174	$LKCSR	172540	ADDRESS OF KW11-P STATUS REGISTER
1176	$LKCSB	172542	ADDRESS OF KW11-P COUNTER BUFFER
1200	$LPVEC	104	KW11-P VECTOR ADDRESS
1202	$LKS	177546	ADDRESS OF KW11-L STATUS REGISTER
1204	$LLVEC	100	KW11-L VECTOR ADDRESS
1212	HZ	74	74 (60 DECIMAL) IF SYSTEM IS 60 HZ; 62 (50 DECIMAL) IF SYSTEM IS 50 HZ.

THE RH11-RH70 ADDRESS AND VECTOR MAY BE CHANGED WHEN THE PROGRAM IS STARTED FROM LOCATION 204(8) OR IF THE PROGRAM DOES NOT RECEIVE A RESPONSE WHEN IT ACCESSES THE DEFAULT RH11-RH70 ADDRESS.

4.2.3 PARAMETERS FOR THE FIRST OPERATION

THE FOLLOWING PARAMETERS ARE USED FOR THE INITIAL OPERATION (IN ADDITION TO THE 'MINIMUM' ADDRESS VALUES).

LOC	TAG	INITIAL VALUE	VALUE RANGE	FUNCTION
1412	BEGPAT	10	1 - 15	THE CODE FOR THE STARTING PATTERN. (IF A WRITE ORDER OR A WRITE CHECK ORDER IS SPECIFIED IN 'BEGCOD')
1414	BEGCOD	5	0 - 5	THE INITIAL COMMAND FOR EACH DRIVE EXERCISED. 0 = WRITE CHECK DATA 1 = WRITE CHECK HEADER & DATA

2 = WRITE DATA
3 = WRITE HEADER & DATA
4 = READ DATA
5 = READ HEADER & DATA

1416 BEGSIZ 404 4 - MAXDL THE BUFFER SIZE FOR THE FIRST DATA TRANSFER OPERATION.

4.3 SWITCH REGISTER SETTINGS

SW <15> = 1 HALT ON ERROR
SW <13> = 1 INHIBIT ERROR TYPEOUT
SW <10> = 1 RING THE TELETYPE BELL IF ERROR
SW <7> = 1 DISPLAY ALL DATA COMPARE ERRORS
SW <6> = 1 DO NOT ALTER THE CURRENT OPERATION PARAMETERS
SW <5> = 1 PARTIAL REGISTER DISPLAY IF ERROR; DO NOT DISPLAY ECC CORRECTION RESULTS
SW <4> = 1 INHIBIT MAXIMUM ERROR COUNT CHECK; DO NOT DEASSIGN DRIVES WHEN NORMAL END OF TEST REACHED.
SW <3> = 1 DISPLAY THE SECTOR IN ERROR (BEFORE RETRY ATTEMPTS) IF 'DCK', 'DTE', OR 'HCF' ERRORS OR AFTER THE 28TH RETRY IF UNCORRECTABLE 'DCK' ERROR.
IF DATA COMPARE ERRORS & SW<7> SET, DISPLAY REST OF BUFFER
SW <2> = 1 INHIBIT SUBSYSTEM STATUS TYPEOUT DURING STARTUP.
INHIBIT PERFORMANCE SUMMARY TYPEOUTS.
SW <1> = 1 INHIBIT DATA COMPARSION AFTER READ ORDERS
SW <0> = 1 READ ONLY MODE

IF THE PROGRAM IS BEING RUN ON A SWITCHLESS PROCESSOR (I.E. AN 11/34) THE PROGRAM WILL DETERMINE THAT THE HARDWARE SWITCH REGISTER IS NOT PRESENT AND WILL USE A 'SOFTWARE' SWITCH REGISTER. THE 'SOFTWARE' SWITCH REGISTER IS LOCATED AT LOCATION 176 (8). THE SETTINGS OF THE 'SOFTWARE' SWITCHES ARE CONTROLLED THROUGH A KEYBOARD ROUTINE WHICH IS CALLED BY TYPING A 'CONTROL G'. THE PROGRAM WILL RECOGNIZE THE 'CONTROL G' AT ANY TIME EXCEPT WHEN THE PROGRAM IS IN KEYBOARD ENTRY MODE, OR IS AT A HIGHER PRIORITY PROCESSING AN RP04/5/6 INTERRUPT. THE 'SOFTWARE' SWITCH VALUES ARE ENTERED AS AN OCTAL NUMBER IN RESPONSE TO THE PROMPT FROM THE SWITCH ENTRY ROUTINE:

'SWR = NNNNNN NEW ='

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

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

4.4 KEYBOARD COMMANDS

THROUGH THE KEYBOARD COMMANDS, THE OPERATOR MAY ASSIGN DRIVES

FOR TEST ('T' COMMAND), WRITE AND CHECK DATA PACKS ('W' COMMAND), PERFORM A SEQUENTIAL READ OF A PACK ('R' COMMAND), REQUEST A DRIVE PERFORMANCE SUMMARY ('S' COMMAND), OR DEASSIGN A DRIVE WHICH IS BEING TESTED, READING, OR WRITING ('D' COMMAND).

AFTER THE PROGRAM HAS BEEN INITIALIZED, THE FOLLOWING MESSAGE WILL BE TYPED:

```
'PROGRAM INITIALIZATION COMPLETE
'TYPE A CONTROL C TO ENTER COMMANDS'
```

KEYBOARD ENTRIES WILL NOT BE RECOGNIZED UNTIL THE OPERATOR TYPES A 'CONTROL C'. WHEN THE PROGRAM SEES A 'CONTROL C' ENTRY, IT WILL SUSPEND THE SCHEDULING OF FURTHER DEVICE OPERATIONS AND WAIT UNTIL ALL OUTSTANDING ORDERS HAVE TERMINATED. THE PROGRAM WILL ENTER COMMAND ENTRY MODE AND TYPE THE FOLLOWING PROMPTING MESSAGE:

```
'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; THE OPERATOR MUST TYPE ANOTHER 'CONTROL C' TO RETURN THE PROGRAM TO COMMAND MODE. IF THE COMMAND ENTERED SPECIFIED AN 'A' DRIVE NUMBER, THE PROGRAM WILL REMAIN IN COMMAND MODE UNTIL ALL AVAILABLE DRIVES HAVE BEEN PROCESSED.

THE 'T', 'W', AND 'R' COMMANDS REQUIRE ADDRESS LIMITS, DRIVE I.D, AND BAD LOCATION ADDRESS ENTRIES FOR THE DRIVE BEING REFERENCED.

THE PROGRAM WILL FIRST ASK FOR ADDRESS LIMITS WITH THE FOLLOWING TYPEOUT:

'ENTER ADDRESS LIMITS FOR DRV #N / RP04' (OR RP05 OR RP06)

THE PROGRAM WILL REQUEST VALUES FOR THE FOLLOWING ADDRESS LIMIT PARAMETERS.

NAME	BASE	DEFAULT VALUE	VALUE RANGE	FUNCTION
MINCYL	10		(SEE NOTE)	THE MINIMUM CYLINDER ADDRESS
MAXCYL	10		(SEE NOTE)	THE MAXIMUM CYLINDER ADDRESS
MINTRK	10	0	0 - 18	THE MINIMUM TRACK ADDRESS
MAXTRK	10	18	0 - 18	THE MAXIMUM TRACK ADDRESS
MINSEC	10	0	0 - 21	THE MINIMUM SECTOR ADDRESS
MAXSEC	10	21	0 - 21	THE MAXIMUM SECTOR ADDRESS

NOTE: 1. THE ADDRESS LIMITS ARE IN DECIMAL

2. THE MAXIMUM VALUES OF 'MINCYL' AND 'MAXCYL' WILL BE EITHER 410 (10) OR 814 (10) DEPENDING ON THE TYPE OF DRIVE.

3. THE MINIMUM CYLINDER, TRACK, OR SECTOR ADDRESS MAY BE SPECIFIED AS BEING LARGER THAN THE MAXIMUM ADDRESS. WHEN

THESE VALUES ARE INVERTED, THE PROGRAM WILL SELECT ADDRESSES BETWEEN THE 'MIN' ADDRESS AND THE UPPER PHYSICAL LIMIT FOR THAT ADDRESS AND BETWEEN '0' AND THE VALUE IN 'MAX'.

EACH COMMAND, EXCEPT THE 'S' AND THE 'D' COMMANDS, WILL ASK FOR A DRIVE IDENTIFICATION ENTRY. THE DRIVE IDENTIFICATION ENTRY REQUEST IS MADE BY THE FOLLOWING MESSAGE:

'ENTER I.D. FOR DRV #N:'

THE OPERATOR MAY ENTER AN I.D. NUMBER FOR THE DRIVE OF UP TO 6 CHARACTERS IN LENGTH. THIS I.D. WILL BE DISPLAYED, ALONG WITH THE DATE AND OPERATOR I.D. ENTRIES (SEE SECTION 4.1), WHEN THE 'SA' COMMAND IS EXECUTED. THE OPERATOR MAY ENTER ANY CHARACTER STRING, TERMINATED BY A 'CARRIAGE RETURN', OR A 'PERIOD' ONLY (NULL ENTRY) IN RESPONSE TO THE I.D. REQUEST.

THE PROGRAM WILL THEN ASK FOR THE ADDRESSES OF KNOWN BAD SPOTS ON THE DISK PACK USED, UP TO 16 ADDRESSES MAY BE ENTERED:

'BAD TRK/SEC ADRS FOR DRV #N ?'

THE OPERATOR MAY DECLARE UP TO 16 BAD LOCATIONS ON THE PACK BEING USED. THE LOCATION MAY BE AN ENTIRE CYLINDER, A BAD TRACK, OR A SINGLE SECTOR. THE FORMATS USED ARE AS FOLLOW:

FORMAT 1: C,T,S<CR>

A. THE PROGRAM WILL INHIBIT DATA ERROR MESSAGES OR WILL IDENTIFY DATA ERRORS WHICH OCCUR AT THE SPECIFIED ADDRESS, DEPENDING ON THE VALUE OF PARAMETER 'NOTPRT'.
B. LEADING ZEROS ARE NOT REQUIRED.

FORMAT 2: C,T<CR>

A. WHEN THIS FORMAT IS USED, THE ENTIRE TRACK WILL BE CONSIDERED BAD; DATA ERRORS WILL BE HANDLED AS IN 'FORMAT 1', ABOVE.
B. LEADING ZEROS ARE NOT REQUIRED.

FORMAT 3: C<CR.

A. WHEN THIS FORMAT IS USED, THE ENTIRE CYLINDER WILL BE CONSIDERED BAD; DATA ERRORS WILL BE HANDLED AS IN 'FORMAT 1' ABOVE.
B. LEADING ZEROS ARE NOT REQUIRED.

NOTE: CYLINDER, TRACK, AND SECTOR ENTRIES ARE IN DECIMAL.

THE OPERATOR MAY TERMINATE THE BAD ADDRESS ENTRY BY ENTERING A 'PERIOD' IN RESPONSE TO THE ENTRY REQUEST OR BY TERMINATING AN ENTRY WITH A 'PERIOD' INSTEAD OF A 'CARRIAGE RETURN'.

4.4.1 'T' COMMAND

USED TO ASSIGN A DRIVE(S) FOR TEST. THIS COMMAND IS REQUIRED
PERFORM THE TEST OF THE DRIVE(S). THE OTHER COMMANDS ARE CONVIENCE
COMMANDS OR SUPPORT COMMANDS.

FORMAT: TN<CR>

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

EXAMPLE: T0<CR> - ASSIGN DRIVE 0 FOR TEST
          TA<CR> - ASSIGN ALL AVAILABLE DRIVES FOR TEST

NOTE: DRIVE OPERATION BEGINS IMMEDIATELY
       AFTER COMMAND IS ENTERED.

4.4.2 '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 RETURN <CR>.

EXAMPLE: D0<CR> - DEASSIGN DRIVE 0
          DA<CR> - DEASSIGN ALL DRIVES BEING
                   TESTED.

NOTES: 1. IF THE 'D' COMMAND REFERENCES A
           DRIVE NOT ASSIGNED THE PROGRAM
           WILL TYPEOUT ''?DRIVE NOT ASSIGNED'

        2. THE DRIVES WILL BE DEASSIGNED AS THEIR
           OPERATIONS COMPLETE.

        3. IF 'DA' IS USED, ONLY DRIVES BEING
           TESTED WILL BE DEASSIGNED - THE
           ERROR MESSAGE IN (1) ABOVE WILL NOT
           BE DISPLAYED.

4.4.3 'S' COMMAND

USED TO REQUEST A PERFORMANCE SUMMARY TYPEOUT FOR THE REFERENCED
DRIVE(S).

FORMAT: SN<CR>

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

EXAMPLE: S0<CR> - TYPEOUT PERFORMANCE SUMMARY FOR DRIVE 0
          SA<CR> - TYPEOUT PERFORMANCE SUMMARY
                   FOR ALL DRIVES BEING TESTED.

NOTES: 1. IF 'SA' IS USED ONLY DRIVES BEING TESTED WILL BE DISPLAYED.

2. IF PARAMETER 'INTRVL' IS NOT ZERO, THE PROGRAM WILL AUTOMATICALLY DISPLAY A PERFORMANCE SUMMARY FOR EACH DRIVE BEING TESTED AT A RATE DETERMINED BY 'INTRVL'.

3. IF THE 'SA' COMMAND IS USED, THE PROGRAM WILL TYPEOUT THE OPERATOR ENTERED DATE, OPERATOR I.D., AND THE DRIVE I.D. FOR EACH DRIVE BEING TESTED. THE DATE AND OPERATOR I.D. WILL NOT BE TYPED OUT IF NO DRIVES ARE BEING TESTED.

4.4.4 'W' COMMAND

USED TO WRITE A DATA PACK WITH DATA ACCEPTABLE TO THE RP04/5/6 MULTI-DRIVE 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: WA<CR> - WRITE DATA PACKS ON ALL AVAILABLE DRIVES.
WO<CR> - GENERATE A DATA PACK ON DRIVE 0.

NOTES: 1. DATA PACKS GENERATED BY THE RP04/5/6 FORMATTER PROGRAM (MD-11-DZRJB) OR BY THE RP04/5/6 MECHANICAL & READ/WRITE PROGRAM (MD-11-DZRJA), TEST 20, ARE ACCEPTABLE. (PACKS WRITTEN BY TESTS 16, 17 OR 21 OF 'DZRJA' CANNOT BE USED AND MUST BE REWRITTEN.)

2. THE 'W' COMMAND SHOULD NOT BE USED UNLESS 'MAXDL' PARAMETER IS APPROXIMATELY 1 TRACK IN SIZE - 5000 (10). IF THE BUFFER SIZE IS MUCH LESS THAN 1 TRACK, THE TIME REQUIRED TO WRITE A DATA PACK IS TOO GREAT TO BE PRACTICAL.

3. THE 'W' COMMAND PERFORMS A SEQUENTIAL WRITE OF THE PACK USING A 'WRITE DATA' COMMAND. THE DATA PATTERN USED FOR EACH WRITE IS SELECTED RANDOMLY. HOWEVER, THE OPERATION OF THE COMMAND IS SEQUENTIAL, BEGINNING AT 'MINCYL', 'MINTRK' AND CONTINUING TO 'MAXCYL', 'MAXTRK'.

4. THE 'W' COMMAND DOES NOT WRITE HEADERS AND ASSUMES THAT THE FORMAT OF THE PACK IS GOOD.

5. THE 'W' COMMAND CANNOT BE STARTED IF SWITCH 0 (READ ONLY MODE) IS SET. IF SWITCH 0 SET SET DURING THE OPERATION OF THE 'W' COMMAND, THE DRIVE PERFORMING THE 'W' COMMAND WILL IGNORE THE SWITCH.

6. THE DATA WRITTEN IS VERIFIED BY A 'WRITE CHECK DATA' COMMAND.

4.4.5 'R' COMMAND

USED TO PERFORM A SEQUENTIAL READ OF THE PACK.

FORMAT: RN<CR>

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

EXAMPLE: RA<CR> - READ THE PACKS ON ALL OF THE ONLINE DRIVES.
RO<CR> - READ THE PACK ON DRIVE 0.

NOTES: 1. THE PROGRAM WILL PERFORMA NORMAL CHECK OF ALL DATA READ. HOWEVER, ALL OPERATIONS WILL BE SEQUENTIAL.

2. THE PROGRAM WILL READ THE PACK STARTING AT THE ADDRESS SPECIFIED BY 'MINCYL', 'MINTRK' TO THE ADDRESS SPECIFIED BY 'MAXCYL', 'MAXTRK'. THE READ WILL BE SEQUENTIAL.

4.4.6 GENERAL COMMAND INFORMATION

A. WHEN A COMMAND IS ENTERED, THE PROGRAM WILL TYPE OUT THE TIME

B. IF THE COMMAND ENTERED IS NOT VALID, THE PROGRAM WILL TYPE 'INVALID COMMAND'.

C. DRIVES ASSIGNED (WITH THE 'T' COMMAND) OR DEASSIGNED (WITH THE 'D' COMMAND) MAY BE ENTERED IN ANY SEQUENCE.

D. THE ERROR RESPONSES FROM THE PROGRAM ARE AS FOLLOWS

RESPONSE	COMMAND(S)
?UNIT N OFFLINE	T, W, R
?UNIT N NOT ASSIGNED	D, S
?UNIT N ALREADY ASSIGNED	T, W, R
?UNIT N NOT PRESENT	T, W, R
?UNIT N UNSAFE	T, W, R
?UNIT N NOT AN RP04/5/6	T, W, R

5. PERFORMANCE SUMMARY TYPEOUT

5.1 THE PROGRAM WILL DISPLAY A PERFORMANCE SUMMARY FOR THE DRIVES BEING EXERCISED. THIS SUMMARY WILL BE DISPLAYED AUTOMATICALLY IF THE PARAMETER 'INTRVL' IS NOT ZERO OR CAN BE DISPLAYED ON REQUEST BY THE OPERATOR THROUGH THE USE OF THE 'S' COMMAND. THE PERFORMANCE SUMMARY TYPEOUT CONTAINS THE FOLLOWING FIELDS:

'DRV' THE DRIVE NUMBER

'PASS'	THE PRESENT PASS COUNT FOR THE DRIVE
'ORDERS'	THE NUMBER OF ORDERS PERFORMED BY THE DRIVE
'SEEKS'	THE NUMBER OF SEEK OPERATIONS THE DRIVE PERFORMED
'WRDS XFER'	THE TOTAL NUMBER OF WORDS WRITTEN AND READ BY THE DRIVE
'WRDS READ'	THE TOTAL NUMBER OF WORDS READ BY THE DRIVE
'SOFT'	THE NUMBER OF SOFT DATA ERRORS
'HARD'	THE NUMBER OF HARD DATA ERRORS
'SKI'	THE NUMBER OF 'SKI' OR 'OCYL' ERRORS
'MISP'	THE NUMBER OF POSITIONING ERRORS
'OTHER'	THE 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 CORRECTED OR BECOMES 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 AT EACH OF THE FOLLOWING OFFSETS:

RP04/5	RP06
+400 MICRO-INCHES	+200 MICRO-INCHES
-400 MICRO-INCHES	-200 MICRO-INCHES
+800 MICRO-INCHES	+400 MICRO-INCHES
-800 MICRO-INCHES	-400 MICRO-INCHES
+1200 MICRO-INCHES	+800 MICRO-INCHES
-1200 MICRO-INCHES	-800 MICRO-INCHES

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 ORDERS.
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 'RDDAT' (READ DATA) OR 'RDHD' (READ HEADER AND DATA) OPERATION UNDER THE FOLLOWING CONDITIONS:

A. THE ORDER TERMINATED WITH NO ERROR.

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

AFTER EACH WRITE OPERATION, THE DATA WRITTEN IS CHECKED WITH THE APPROPRIATE WRITE CHECK COMMAND - IF PARAMETER 'AUTOCK' IS 1. (IF 'AUTOCK' IS 0, WRITE CHECKS WILL BE PERFORMED RANDOMLY.)

6.3 SECTOR REFORMATTING

THE PROGRAM WILL REFORMAT AN UNCORRECTABLE ERROR SECTOR IN THE FOLLOWING CASES (PARAMETER 'FORMAT' MUST BE SET AND SW<00> = 0). THIS PREVENTS THE SAME ERROR FROM BEING CONTINUOUSLY REPORTED.

A. DATA CHECK ERRORS - EM21
B. HEADER READ ERRORS - EM20, EM24, EM25, EM26, EM27
C. DRIVE TIMING ERRORS - EM31
D. OPERATION INCOMPLETE ERRORS - EM32
E. WRITE CHECK ERRORS - EM22, EM23

6.4 BAD TRACK/SECTOR FLAGGING

SINCE THE RP04 SUBSYSTEM DOES NOT HAVE AN AUTOMATIC BAD TRACK HANDLING CAPABILTIY, THE MULTIDRIVE EXERCISER ALLOWS THE OPERATOR TO IDENTIFY UP TO 8 BAD TRACK/SECTOR LOCATIONS WHEN THE DRIVE IS ASSIGNED FOR TEST. (SEE SECTION 4.1 FOR ADDITIONAL INFORMATION.)

IF ONE OF THE FOLLOWING ERRORS OCCURS AT A LOCATION IDENTIFIED BY THE OPERATOR, THE PROGRAM WILL INHIBIT THE ERROR REPORT FOR THAT ERROR.

DATA CHECK ERRORS ('DCK')
WRITE CHECK ERRORS ('WCE')
OPERATION INCOMPLETE ERRORS ('OPI')
DRIVE TIMING ERRORS ('DTE')
HEADER READ ERRORS ('FER' & 'HCRC', 'HCE' & 'HCRC', 'HCRC')

OPTIONALLY, THE OPERATOR MAY REQUEST AN ERROR REPORT FOR FLAGGED AREAS. (PARAMETER 'NOTPRT' MUST BE SET TO 0 AT STARTUP.) THE PROGRAM WILL IDENTIFY ERROR MESSAGES ASSOCIATED WITH THE PACK; THESE ERRORS WILL NOT BE ADDED TO THE ERROR TOTALS MAINTAINED BY THE PROGRAM.

7. ERROR MESSAGES

DRIVE ERRORS ARE REPORTED ON THE TELETYPE OR (IF AVAILABLE) A LINE PRINTER. ALL ERROR CONDITIONS ARE REPORTED IN ERROR MESSAGES, 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

MESSAGES EM1, EM2, EM3, EM4, EM5, EM10, EM11, & EM12 ARE ALWAYS DISPLAYED ON THE TTY. THE OTHER MESSAGES ARE DISPLAYED ON EITHER THE LINE PRINTER (IF AVAILABLE) OR THE TTY.

(THE MESSAGE TAGS ARE GIVEN FOR REFERENCE.)

MESSAGE TAG	TEXT
EM1	RH11 INTERRUPT OCCURRED (RPAS=0) THE RH11 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 RH11 DETECTED A CONTROL BUS PARITY ERROR WHEN READING THE INDICATED REGISTER FROM THE INDICATED DRIVE.
EM4	MASSBUS PARITY ERROR (PAR=1) THE INDICATED RP04 DETECTED A CONTROL BUS PARITY ERROR WHEN THE RH11 LOADED THE SPECIFIED REGISTER.
EM5	ADDRESS PLUG CHANGE BIT SET THE 'OPE' BIT WAS SET WHEN THE INDICATED DRIVE INTERRUPTED.
EM6	RH11 DIDN'T RESPOND TO ADDRESSING WHEN THE PROGRAM ADDRESSED THE RH11, 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 RH11 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.

EM13 OPERATION NOT COMPLETED WITHIN TIME LIMIT

THE DRIVE DID NOT COMPLETE THE OPERATION WITHIN 1 SECOND AFTER THE OPERATION WAS INITIATED.

EM14 UNIT WENT OFFLINE

THE DRIVE WENT OFFLINE DURING THE INDICATED OPERATION. (THE 'MOL' BIT BECAME ZERO.) THE PROGRAM WILL AUTOMATICALLY DEASSIGN THE DRIVE. THE OPERATOR MUST REASSIGN THE DRIVE WITH THE 'T' COMMAND TO RE-INITIATE TESTING.

EM15 NO RESPONSE TO PORT REQUEST

THE PROGRAM IS TESTING A DUAL PORT DRIVE WHICH HAS NOT SWITCHED TO THE REQUESTING PORT WITHIN 10 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 3 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 3 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 3 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 3 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 3 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 3 TIMES.

EM27 HEADER COMPARE ('HCE') ERROR

SIMILAR TO EM26 EXCEPT THAT THE 'HCE' BIT WAS SET INITIALLY. THE OPERATION WILL BE RETRIED 3 TIMES.

EM30 MISCELLANEOUS DRIVE ERROR

THIS MESSAGE IS GIVEN FOR THE FOLLOWING ERROR BITS:
'IXE', 'AOE', 'RMR', 'ILF', OR 'ILR'

EM31 OPERATION INCOMPLETE ('OPI') ERROR

AN OPERATION INCOMPLETE ERROR OCCURRED AT THE INDICATED SECTOR.

EM32 DRIVE TIMING ('DTE') ERROR

DRIVE TIMING ERROR OCCURRED ON THE INDICATED SECTOR. THE OPERATION WILL BE RETRIED 3 TIMES.

EM33 PARITY ('PAR') ERROR AFTER OPERATION STARTED

THE 'PAR' BIT WAS SET WHEN THE OPERATION WAS COMPLETED. THE OPERATION WILL BE RETRIED 3 TIMES.

EM34 WRITE CLOCK FAILURE ('WCF')

A WRITE CLOCK FAILURE OCCURRED DURING THE OPERATION. THE OPERATION WILL BE RETRIED 3 TIMES.

EM35 INVALID ADDRESS ('IAE') ERROR

AN INVALID ADDRESS ERROR OCCURRED DURING THE OPERATION.

EM36 WRITE LOCK ('WLE') ERROR

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

EM40 RH11 OR UNIBUS TRANSFER ERROR

'TRE' IS SET IN THE RH11 CONTROL REGISTER AND NO DRIVE ERROR WAS OCCURRED. THE OPERATION WILL BE RETRIED 3 TIMES IF THE ERROR WAS CAUSED BY 'DLT', 'DPE', 'MXF', 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

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

EM44 ERROR BIT(S) SET, BUT NO ERROR SIGNALED BY THE RH11

THE OPERATION COMPLETED NORMALLY; HOWEVER, THE PROGRAM FOUND EITHER ERROR BITS IN THE RP04 SET OR ERROR BITS IN THE RH11 SET.

EM45 ECC LOGIC FAILURE

THE CONTENTS OF EITHER THE ECC POSITION REGISTER (RPEC1) OR THE CONTENTS OF ECC PATTERN REGISTER (RPEC2) ARE NOT VALID. THE POSITION REIGSTER IS EITHER'0' OR > 10041 (8) OR THE PATTERN REGISTER CONTAINS ZEROS.

EM46 BUS ADDRESS OR WORD COUNT NOT CONSISTENT

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

EM50 SEEK INCOMPLETE OR OFF CYLINDER ERROR

THE DRIVE SIGNALED EITHER 'SKI' OR 'OCYL' ERROR BITS.

EM51 PROGRAM DETECTED POSITIONING ERROR

A HEADER COMPARE ERROR OCCURRED ('HCE'); HOWEVER, WHEN THE PROGRAM EXAMINED THE HEADER OF THE SECTOR IN ERROR, IT FOUND THAT THE CYLINDER FIELD DID NOT AGREE WITH THE CONTENTS OF 'RPCC' OF THE DRIVE. THE DRIVE WILL BE RECALIBRATED.

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

TT:TT:TT (DESCRIPTION OF ERROR)

'TT:TT:TT' IS THE TIME SINCE THE PROGRAM
WAS STARTED. TT:TT:TT IS GIVEN IN HOURS:
MINUTES: SECONDS.

LINE 2

'PRESENT ORDER = XXXX PREVIOUS ORDER = YYYY'

MNEMONICS USED FOR THE ORDERS ARE DEFINED BELOW:

UNLOAD - UNLOAD (OCTAL 3)
SEEK - SEEK (OCTAL 5)
RECAL - RECALIBRATE (OCTAL 7)
DRVCLR - DRIVE CLEAR (OCTAL 11)
RELSE - RELEASE (OCTAL 13)
OFFSET - OFFSET (OCTAL 15)
RTC - RETURN TO CENTERLINE (OCTAL 17)
READIN - READIN PRESET (OCTAL 21)
PACK - PACK ACKNOWLEDGE (OCTAL 23)
SEARCH - SEARCH (OCTAL 31)
WCKD - WRITE CHECK DATA (OCTAL 51)
WCKHD - WRITE CHECK HEADER & DATA (OCTAL 53)
WRTDAT - WRITE DATA (OCTAL 61)
WRTHD - WRITE CHECK HEADER & DATA (OCTAL 63)
RDDAT - READ DATA (OCTAL 71)
RDHD - READ HEADER & DATA (OCTAL 73)

(DISPLAY OF THE RH11/RP04/5/6 REGISTERS IN TWO GROUPS:
RPCS1,RPCS2,RPDS1,RPER1,RPER2,RPER3,RPEC1, & 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 PACK 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 RP04/5/6 REGISTERS. THE CONTENTS OF THIS WORD IDENTIFY HOW THE ERROR WAS PROCESSED BY THE RP04/5/6 DRIVE HANDLER ROUTINE. THE BITS IN THIS WORD ARE ENCODED AS FOLLOWS:

BIT #	MEANING IF BIT IS '1'
15	ERROR OCCURRED DONE (BIT07=0), BITS 14-9, 2, 1 SPECIFY TYPE DONE (BIT07=1), BITS 6-3 SPECIFY TYPE
14	DRIVE IS OFFLINE
12	PERSISTENT UNSAFE CONDITION EXISTS
11	UNCORRECTABLE PARITY ERROR OCCURRED
10	FATAL PARITY ERROR OCCURRED. MASSBUS CLEAR WAS PERFORMED
9	OPERATION NOT COMPLETED WITHIN 1 SECOND MASSBUS CLEAR PERFORMED. ALL OTHER OUTSTANDING OPERATIONS WERE RESTARTED.
7	DONE - OPERATION COMPLETED
6	DATA ERROR OCCURRED DURING THE TRANSFER
5	ERROR OCCURRED WHILE SEARCHING FOR THE 'TRANSFER' SECTOR OR DURING RECALIBRATE OR OFFSET COMMANDS
4	CORRECTABLE UNSAFE CONDITION OCCURRED
3	DRIVE ERROR OCCURRED THAT CAUSED AN AUTOMATIC RECALIBRATE SEQUENCE
2	PORT REQUEST TIMEOUT
1	NON-EXISTENT DRIVE REQUESTED

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

PRESENT 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
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 RH11 BUFFER ADDRESS
REGISTER AND THE RH11 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 RP04 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 SIZE = YYYY ACTUAL NUMBR WRDS XFRD = ZZZZ

THIS LINE GIVES THE STARTING ADDRESS OF THE BUFFER USED FOR THE
CURRENT DATA TRANSFER OPERATION, ITS SIZE, AND THE ACTUAL NUMBER

OF WORD TRANSFERED. THE STARTING ADDRESS OF THE BUFFER IS IN
OCTAL, THE SIZE AND WORD TRANSFERED VALUE ARE IN DECIMAL.

LINE 11

GOOD DATA = XXXX BAD DATA = YYYY SECT POS = ZZZ

THIS LINE GIVES THE GOOD DATA, THE ACTUAL DATA FROM THE DISK,
AND THE LOCATION IN THE SECTOR OF THE ACTUAL DATA. THE SECTOR
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 OFFSET X MICRO-INCHES

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

LINE 16

ECC CORRECTABLE AT OFFSET X MICRO-INCHES

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 COUNT 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 RH11/RP04 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 OFFSET VALUE FROM 'RPEC1' 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

ORDERS: WWW ERRORS: X WRDS XFR: YYYY WRDS READ: ZZZZ

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

'ORDERS' IS THE TOTAL NUMBER OF COMMANDS GIVEN TO THE DRIVE
WHICH REPORTED THE ERROR.

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

'WRDS XFR' IS THE TOTAL NUMBER OF WORDS WRITTEN AND READ BY
THE DRIVE.

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

LINE 28

ORDERS: WWWW TOTAL SEEKS: XXX TOTAL POS ERR = YYY TOTAL SKI, OCYL ERR = Z

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

'ORDERS' IS THE TOTAL NUMBER OF ORDERS GIVEN TO THE DRIVE WHICH
REPORTED THE ERROR.

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

'TOTAL POS ERR' IS THE TOTAL NUMBER OF POSITIONING ERRORS WHICH
THE PROGRAM DETECTED FOR THE DRIVE.

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

8. PROGRAM DESCRIPTION

8.1 PROGRAM OPERATION

WHEN THE PROGRAM IS STARTED, ALL TABLES AND
PARAMETERS ARE CLEARED OR INITIALIZED. THE PARAMETERS WHICH ARE
UNDER OPERATOR TTY ENTRY CONTROL ARE CHECKED FOR VALIDITY AND
CONSISTENCY. RH11 INTERRUPT ENABLE ('IE') IS SET, TTY KEYBOARD
INTERRUPT ENABLE IS SET, AND THE KW11-L OR KW11-P CLOCK IS STARTED.
WHEN THESE ACTIONS HAVE BEEN COMPLETED, THE PROGRAM TYPES OUT
'PROGRAM INTIALIZE COMPLETE'. 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 SUMMARY 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 RP04/5/6, AND IS ONLINE. THE ASSIGNMENT ROUTINE
THEN ISSUES A 'READIN PRESET' INSTRUCTION, SETS 'FMT22', 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 ORDER, THE ASSIGNED BUFFER
WILL BE FILLED WITH THE SELECTED PATTERN. (WRITE CHECK ORDERS ARE
ISSUED AFTER EACH WRITE ORDER. THE WRITE CHECK ORDER USES THE
PARAMETERS SELECTED FOR THE PRECEEDING WRITE ORDER.) 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
PERFORM THE OPERATION, THE PROGRAM ISSUES A SEARCH INSTURCTION
TO THE DRIVE WITH A 'TARGET' SECTOR WHICH IS 8 SECTORS 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 READS
THE LOOK AHEAD REGISTER (RPLA) OF THE INTERRUPTING DRIVE AND
COMPARES THE POSITION OF THE DISK WITH THAT OF THE DESIRED SECTOR.

IF OTHER DRIVES ARE WAITING ON CYLINDER, THEY ARE ALSO CHECKED. THE PROGRAM THEN ISSUES THE REQUESTED ORDER TO THE DRIVE NEAREST ITS TRANSFER SECTOR. THE DRIVES NOT SELECTED WILL HAVE ANOTHER SEARCH INITIATED. IF A DRIVE IS NOT SELECTED FOR TRANSFER AFTER THREE REVOLUTIONS OF ITS DISK, IT IS GIVEN PRIORITY OVER DRIVES WHICH HAVE NOTE BEEN ON CYLINDER AS LONG.

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 RH11 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 ORDER WAS A READ ORDER, 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.

    PERSISTENT UNSAFE CONDITION - EM12
    UNCORRECTABLE MASSBUS PARITY ERROR - EM10
    FATAL MASSBUS PARITY ERROR - EM11
    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 - EM51
    SEEK INCOMPLETE OR OFF CYLINDER ERROR - EM50
    WRITE CHECK WITHOUT 'DCK' SET - EM23
    RH11 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 RH11 - 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 ORDER TERMINATION.

WHEN THE DUAL PORT HANDLER ROUTINE IN THE MULTIDRIVE PROGRAM RECEIVES A REQUEST FOR A DRIVE, THE PROGRAM VERIFIES THAT THE DRIVE IS ONLINE. THE DRIVE IS SELECTED AND 0'S ARE WRITTEN INTO 'RPDS1': IF THE DRIVE IS IN NEUTRAL, THIS WILL SEIZE THE DRIVE. IF THE DRIVE IS SEIZED BY THE OTHER PORT, WRITING INTO 'RPDS1' WILL SET 'PORT REQUEST'. THE PROGRAM 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 20 SECOND TIMER FOR THE DRIVE. IF THE DRIVE HAS NOT SWITCHED TO THE REQUESTING SYSTEM WITHIN THE 20 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 MOD33 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 ORDER 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 EXCLUDE ALL ADDRESSES BETWEEN 'MAX' AND 'MIN' FROM THE SELECTION. FOR EXAMPLE: IF 'MINTRK' IS 18 AND 'MAXTRK' IS IS 5, THEN TRACK ADDRESS SELECTION WILL EXCLUDE TRACKS 6 - 17 FROM THE SELECTION AND SELECT AN ADDRESS FROM AMONG ADDRESSES 18, 0, 1, 2, 3, 4, 5.

B. THE BUFFER SIZE IS RANDOM SELECTED BETWEEN 4 (10) - AND THE VALUE IN 'MAXDL'. THE SIZE SELECTED IS WEIGHTED TO ENSURE THAT AT LEAST 4 WORDS ARE WRITTEN IN THE DATA AREA OF THE LAST SECTOR. 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.

C. THE DATA WRITTEN IS RANDOMLY SELECTED AMONG THE 15 STANDARD PATTERNS. THE KEYWORDS IN THE HEADER (WHEN PERFORMING A WRITE HEADER & DATA ORDER) ARE ZERO FILLED. THE PROGRAM EXPECTS TO FIND THAT THE KEYWORDS ARE ZERO.

D. THE ORDERS ARE SELECTED RANDOMLY. WRITE CHECK DATA AND WRITE CHECK HEADER & DATA ORDERS ARE PERFORMED ONLY IF THE PREVIOUS ORDER WAS THE APPROPRIATE DATA ORDER. IF THE 'FORMAT' PARAMETER IS ZERO, THE PROGRAM WILL NOT SELECT WRITE HEADER & DATA (AND WRITE CHECK HEADER & DATA) ORDERS. WHEN THE PROGRAM SELECTS A WRITE HEADER & DATA ORDER, THE BUFFER SIZE IS FORCED TO 260 (10); THE PROGRAM WILL NOT PERFORM A MULTI-SECTOR FORMAT WRITE OPERATION.

E. THE FIRST ORDER PERFORMED AFTER A UNIT IS ASSIGNED WITH A 'T', 'W', OR 'R' COMMAND IS NOT RANDOMLY SELECTED. THE PARAMETERS FOR THE FIRST OPERATION ARE THE MINIMUM OR STARTING VALUES OF THE VARIABLES.

8.4 DATA PATTERNS

THE PROGRAM SELECTS ONE OF THE FOLLOWING DATA PATTERNS TO WRITE WHEN A WRITE ORDER 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: FROM THE FIRST 4 DATA WORDS OF EACH SECTOR, THE PROGRAM MATCHES THE DATA TO ONE OF THE FOLLOWING PATTERNS. TO MAINTAIN COMPATIABILITY WITH PACKS WRITTEN BY THE FORMAT PROGRAM (MAINDEC-11-DZRJB), THE PROGRAM WILL ACCEPT ALL ZERO'S AND AND ALL ONE'S PATTERNS; HOWEVER, ALL ZERO'S AND ALL ONE'S PATTERNS ARE NOT WRITTEN BY THE EXERCISER PROGRAM.

PATTERN '8' IS DEFINED AS THE 'WORST CASE' PATTERN.

PAT 1	PAT 2	PAT 3	PAT 4	PAT 5	PAT 6	PAT 7	PAT 8
000001	177776	000000	000000	052525	007417	026455	165555
000003	177774	000000	010421	052525	007417	026455	133333
000007	177770	000000	021042	052525	007417	026455	165555
000017	177760	177777	031463	125252	170360	151322	133333
000037	177740	177777	042104	125252	170360	151322	165555
000077	177700	177777	052525	125252	170360	151322	133333
000177	177600	000000	063146	052525	007417	026455	165555
000377	177400	000000	073567	052525	007417	026455	133333
000777	177000	177777	104210	125252	170360	151322	165555
001777	176000	177777	114631	125252	170360	151322	133333
003777	174000	000000	125252	052525	007417	026455	165555
007777	170000	177777	135673	125252	170360	151322	133333
017777	160000	000000	146314	052525	007417	026455	165555

037777	140000	177777	156735	125252	170360	151322	133333
077777	100000	000000	167356	052525	007417	026455	165555
177777	000000	177777	177777	125252	170360	151322	133333

PAT 9	PAT 10	PAT 11	PAT 12	PAT 13	PAT 14	PAT 15
000001	177776	172666	077777	153333	000000	177777
000002	177775	155555	137777	066667	177777	000000
000004	177773	172666	157777	153333	177777	000000
000010	177767	155555	167777	066667	177777	000000
000020	177757	172666	173777	153333	177777	000000
000040	177737	155555	175777	066667	177777	000000
000100	177677	172666	176777	153333	177777	000000
000200	177577	155555	177377	066667	177777	000000
000400	177377	172666	177577	153333	177777	000000
001000	176777	155555	177677	066667	177777	000000
002000	175777	172666	177737	153333	177777	000000
004000	173777	155555	177757	066667	177777	000000
010000	167777	172666	177767	153333	177777	000000
020000	157777	155555	177773	066667	177777	000000
040000	137777	172666	177775	153333	177777	000000
100000	077777	155555	177776	066667	177777	000000

9. PROGRAM LISTING

```
1772
1773                    .TITLE  MD-11-DZRJD-B, RP04/5/6 MULTI-DRIVE EXERCISER
1774                    ;*COPYRIGHT (C) 1975,1977
1775                    ;*DIGITAL EQUIPMENT CORP.
1776                    ;*MAYNARD, MASS. 01754
1777                    ;*
1778                    ;*PROGRAM BY C. HESS
1779                    ;*
1780                    ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
1781                    ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
1782                    ;*
1783                    .SBTTL  OPERATIONAL SWITCH SETTINGS
1784                    ;*
1785                    ;*      SWITCH                  USE
1786                    ;*      ------          --------------------
1787                    ;*        15            HALT ON ERROR
1788                    ;*        13            INHIBIT ERROR TYPEOUTS
1789                    ;*        10            BELL ON ERROR
1790                    ;*         7            DISPLAY ALL DATA COMPARE ERRORS
1791                    ;*         6            DON'T CHANGE PARAMETERS (LOOP ON PRESENT VALUES)
1792                    ;*         5            A. PARTIAL REGISTER DISPLAY IF ERROR
1793                    ;*                      B. NO ECC CORRECTION RESULTS DISPLAYED IF ERROR
1794                    ;*         4            A. DO NOT CHECK FOR MAXIMUM ERROR COUNTS
1795                    ;*                      B. DO NOT DROP DRIVE WHEN NORMAL END OF TEST REACHED
```

```
                        ;*          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 & SW7 SET, DISPLAY
                        ;*                        REMAINDER OF BUFFER
                        ;*          2           A. DON'T TYPE SUBSYSTEM STAT'JS WHEN PROGRAM STARTED
                        ;*                      B. DON'T TYPE PERFORMANCE SUMMARY
                        ;*          1           INHIBIT DATA COMPARSION AFTER READ ORDERS
                        ;*          0           READ ONLY MODE
                        .SBTTL  BASIC DEFINITIONS

                        ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100                  STACK=  1100
                        .EQUIV  EMT,ERROR       ;;BASIC DEFINITION OF ERROR CALL
                        .EQUIV  IOT,SCOPE       ;;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
                        .EQUIV  PS,PSW
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
```

```
1852    002000          SW10=   2000
1853    001000          SW09=   1000
1854    000400          SW08=   400
1855    000200          SW07=   200
1856    000100          SW06=   100
1857    000040          SW05=   40
1858    000020          SW04=   20
1859    000010          SW03=   10
1860    000004          SW02=   4
1861    000002          SW01=   2
1862    000001          SW00=   1
1863                    .EQUIV  SW09,SW9
1864                    .EQUIV  SW08,SW8
1865                    .EQUIV  SW07,SW7
1866                    .EQUIV  SW06,SW6
1867                    .EQUIV  SW05,SW5
1868                    .EQUIV  SW04,SW4
1869                    .EQUIV  SW03,SW3
1870                    .EQUIV  SW02,SW2
1871                    .EQUIV  SW01,SW1
1872                    .EQUIV  SW00,SW0
1873
1874                    ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
1875    100000          BIT15=  100000
1876    040000          BIT14=  40000
1877    020000          BIT13=  20000
1878    010000          BIT12=  10000
1879    004000          BIT11=  4000
1880    002000          BIT10=  2000
1881    001000          BIT09=  1000
1882    000400          BIT08=  400
1883    000200          BIT07=  200
1884    000100          BIT06=  100
1885    000040          BIT05=  40
1886    000020          BIT04=  20
1887    000010          BIT03=  10
1888    000004          BIT02=  4
1889    000002          BIT01=  2
1890    000001          BIT00=  1
1891                    .EQUIV  BIT09,BIT9
1892                    .EQUIV  BIT08,BIT8
1893                    .EQUIV  BIT07,BIT7
1894                    .EQUIV  BIT06,BIT6
1895                    .EQUIV  BIT05,BIT5
1896                    .EQUIV  BIT04,BIT4
1897                    .EQUIV  BIT03,BIT3
1898                    .EQUIV  BIT02,BIT2
1899                    .EQUIV  BIT01,BIT1
1900                    .EQUIV  BIT00,BIT0
1901
1902                    ;*BASIC "CPU" TRAP VECTOR ADDRESSES
1903    000004          ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
1904    000010          RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
1905    000014          TBITVEC=14              ;;"T" BIT
1906    000014          TRTVEC= 14              ;;TRACE TRAP
1907    000014          BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
```

```
1908    000020          IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
1909    000024          PWRVEC= 24              ;;POWER FAIL
1910    000030          EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
1911    000034          TRAPVEC=34              ;;"TRAP" TRAP
1912    000060          TKVEC=  60              ;;TTY KEYBOARD VECTOR
1913    000064          TPVEC=  64              ;;TTY PRINTER VECTOR
1914    000240          PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
1915
1916                    ;;*****************************************************************
1917
1918                    .SBTTL  RH11 REGISTERS
1919
1920                    ;;*****************************************************************
1921
1922                    ;CONTROL AND STATUS REGISTER 1 (RPCS1)
1923
1924    000100          IE=     100                     ;INTERRUPT ENABLE (BIT #6)
1925    000200          RDY=    200                     ;READY (BIT #7)
1926    000400          A16=    400                     ;HIGH ORDER BUS ADDRESS BIT (BIT #8)
1927    001000          A17=    1000                    ;HIGH ORDER BUS ADDRESS BIT (BIT #9)
1928    002000          PSEL=   2000                    ;PORT SELECT (BIT #10)
1929    020000          MCPE=   20000                   ;MASSBUSS PARITY ERROR (BIT #13)
1930    040000          TRE=    40000                   ;TRANSFER ERROR (BIT #14)
1931                    ;SC=    100000                  ;SPECIAL CONDITION (BIT #15)
1932
1933                    ;WORD COUNT REGISTER (RPWC)
1934                    ;(EACH BIT IS CALLED BY BIT NUMBER)
1935
1936                    ;BUS ADDRESS REGISTER (RPBA)
1937                    ;(EACH BIT IS CALLED BY BIT NUMBER)
1938
1939                    ;CONTROL AND STATUS REGISTER 2 (RPCS2)
1940
1941    000001          US1=    1                       ;UNIT SELECT (BIT #0)
1942    000002          US2=    2                       ;UNIT SELECT (BIT #1)
1943    000004          US4=    4                       ;UNIT SELECT (BIT #2)
1944    000010          BAI=    10                      ;BUS ADDRESS INCREMENT INHIBIT (BIT #3)
1945    000020          PAT=    20                      ;MASSBUS PARITY TEST (BIT #4)
1946    000040          CLR=    40                      ;CLEAR (BIT #5)
1947    000100          IR=     100                     ;INPUT READY (BIT #6)
1948    000200          OR=     200                     ;OUTPUT READY (BIT #7)
1949    000400          MPE=    400                     ;MASS BUS PARITY ERROR (BIT #8)
1950    001000          MXF=    1000                    ;MISSED TRANSFER ERROR (BIT #9)
1951    002000          PGE=    2000                    ;PROGRAM ERROR (BIT #10)
1952    004000          NEM=    4000                    ;NON EXISTENT MEMORY (BIT #11)
1953    010000          NED=    10000                   ;NON EXISTENT DRIVE (BIT #12)
1954    020000          UPE=    20000                   ;UNIBUS PARITY ERROR (BIT #13)
1955    040000          WCE=    40000                   ;WRITE CHECK ERROR (BIT #14)
1956    100000          DLT=    100000                  ;DATA LATE (BIT #15)
1957
1958                    ;DATA BUFFER REGISTER (RPDB)
1959                    ;(EACH BIT IS CALLED BY BIT NUMBER)
1960
1961
1962                    ;;*****************************************************************
1963
```

```
                        .SBTTL  RP04/5/6 REGISTERS
                        ;;*****************************************************************
                        ;CONTROL AND STATUS 1 REGISTER. (#00)

000001                  GO=     1               ;GO BIT (BIT #0)
000002                  F1=     2               ;FUNCTION CODE BIT #1
000004                  F2=     4               ;FUNCTION CODE BIT #2
000010                  F3=     10              ;FUNCTION CODE BIT #3
000020                  F4=     20              ;FUNCTION CODE BIT #4
000040                  F5=     40              ;FUNCTION CODE BIT #5
004000                  DVA=    4000            ;DEVICE AVAILABLE (BIT #11)

                        ;DRIVE STATUS REGISTER (RPDS1) (#01)

                        ;DF5=   1               DRIVE FORWARD 5"/SEC.  (BIT #0)
000002                  DFF20=  2               ;DRIVE FORWARD 20"/SEC.  (BIT #1)
000004                  DIGB=   4               ;DRIVE TO INNER GUARD BAND (BIT #2)
000010                  GRV=    10              ;GO REVERSE (BIT #3)
000020                  DL64=   20              ;DIFFERENCE LESS THAN 64 (BIT #4)
000040                  DE1=    40              ;DIFFERENCE EQUALS 1 (BIT #5)
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                  LST=    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)
002000                  IAE=    2000            ;INVALID ADDRESS ERROR (BIT #10)
004000                  WLE=    4000            ;WRITE LOCK ERROR (BIT #11)
010000                  DTE=    10000           ;DRIVE TIMING ERROR (BIT #12)
020000                  OPI=    20000           ;OPERATION INCOMPLETE (BIT #13)
040000                  UNS=    40000           ;DRIVE UNSAFE (BIT #14)
100000                  DCK=    100000          ;DATA CHECK ERROR (BIT 15)

                        ;MAINTAINABILITY REGISTER (RPMR)(#03)

000001                  DMD=    1               ;DIAGINOSTIC MODE (BIT #0)
000002                  MCLK=   2               ;MAINTAINABILITY CLOCK (BIT #1)
```

```
                        MINX=   4               ;MAINTAINABILITY INDEX (BIT #2)
                        MSTCK=  10              ;MAINTAINABILITY SECTOR CLOCK (BIT #3)
                        MRD=    20              ;MAINTAINABILITY READ (BIT #4)
                        MWR=    40              ;MAINTAINABILITY WRITE (BIT #5)
                        DTSY=   200             ;MAINTAINABILITY SYNC DETECTED (BIT #7)

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

                        AT0=    1               ;DEVICE 0 (BIT #0)
                        AT1=    2               ;DEVICE 1 (BIT #1)
                        AT2=    4               ;DEVICE 2 (BIT #2)
                        AT3=    10              ;DEVICE 3 (BIT #3)
                        AT4=    20              ;DEVICE 4 (BIT #4)
                        AT5=    40              ;DEVICE 5 (BIT #5)
                        AT6=    100             ;DEVICE 6 (BIT #6)
                        AT7=    200             ;DEVICE 7 (BIT #7)

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

                        ;DRIVE TYPE REGISTER (RPDT) (#06)

                        DT00=   1               ;DRIVE TYPE NUMBER BIT 1
                        DT01=   2               ;DRIVE TYPE NUMBER BIT 2
                        DT02=   4               ;DRIVE TYPE NUMBER BIT 3
                        DT03=   10              ;DRIVE TYPE NUMBER BIT 4
                        DT04=   20              ;DRIVE TYPE NUMBER BIT 5
                        DT05=   40              ;DRIVE TYPE NUMBER BIT 6
                        DT06=   100             ;DRIVE TYPE NUMBER BIT 7
                        DT07=   200             ;DRIVE TYPE NUMBER BIT 8
                        DT08=   400             ;DRIVE TYPE NUMBER BIT 9
                        DRQ=    4000            ;DRIVE REQUEST REQUIRED (BIT #11)
                        MOH=    20000           ;MOVING HEAD (BIT #13)
                        TAP=    40000           ;TAPE DRIVE (BIT #14)
                        NBA=    100000          ;NOT BLOCK ADDRESSED (BIT #15)

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

                        EXT1=   1               ;EXTENSION 1 (BIT #0)
                        EXT2=   2               ;EXTENSION 2 (BIT #1)
                        EXT4=   4               ;EXTENSION 3 (BIT #2)
                        EXT10=  10              ;EXTENSION 4 (BIT #3)
                        EXT20=  20              ;EXTENSION 5 (BIT #4)
                        EXT40=  40              ;EXTENSION 6 (BIT #5)
                        SC1=    100             ;SECTOR COUNT FIELD 0 (BIT #6)
                        SC2=    200             ;SECTOR COUNT FIELD 1 (BIT #7)
                        ;SC4=   400             ;SECTOR COUNT FIELD 2 (BIT #8)
                        SC10=   1000            ;SECTOR COUNT FIELD 3 (BIT #9)
                        SC20=   2000            ;SECTOR COUNT FIELD 4 (BIT #10)
                        TRK1=   4000            ;TRACK FIELD 1 (BIT #11)
                        TRK2=   10000           ;TRACK FIELD 2 (BIT #12)
                        TRK4=   20000           ;TRACK FIELD 3 (BIT #13)
                        TRK10=  40000           ;TRACK FIELD 4 (BIT #14)
                        TRK20=  100000          ;TRACK FIELD 5 (BIT #15)

                        ;RP04 ERROR REGISTER #2 (RPER2) (#10)
```

```
2076
2077            000001          WCU=    1               ;WRITE CURRENT UNSAFE (BIT #0)
2078            000002          CSF=    2               ;CURRENT SINK FAILURE (BIT #1)
2079            000004          WSU=    4               ;WRITE SELECT UNSAFE (BIT #2)
2080            000010          CSU=    10              ;CURRENT SWITCH UNSAFE (BIT #3)
2081            000020          MSE=    20              ;MOTOR SEQUENCE ERROR (BIT #4)
2082            000040          TDF=    40              ;TRANSITIONS DETECTOR FAILURE (BIT #5)
2083            000100          TUF=    100             ;TRANSITIONS UNSAFE (BIT #6)
2084            000200          FEN=    200             ;FAILSAFE ENABLED (BIT #7)
2085            000400          WRU=    400             ;WRITE READY UNSAFE (BIT #8)
2086            001000          MHS=    1000            ;MULTIPLE HEAD SELECT (BIT #9)
2087            002000          NHS=    2000            ;NO HEAD SELECTION (BIT #10)
2088            004000          IXE=    4000            ;INDEX ERROR (BIT #11)
2089            010000          VU30=   10000           ;30VOLT UNSAFE (BIT #12)
2090            020000          PLU=    20000           ;PLO UNSAFE (BIT #13)
2091            100000          ACU=    100000          ;AC UNSAFE (BIT #15)
2092
2093                            ;RP05/6 ERROR REGISTER #02 (RPER2) (#10)
2094
2095            000001          WCU=    1               ;WRITE CURRENT UNSAFE (BIT #0)
2096            000002          CSF=    2               ;CURRENT SINK FAILURE (BIT #1)
2097            000004          WSU=    4               ;WRITE SELECT UNSAFE (BIT #2)
2098            000010          CSU=    10              ;CURRENT SWITCH UNSAFE (BIT #3)
2099            000020          RAW=    20              ;READ AND WRITE (BIT #4)
2100            000040          TDF=    40              ;TRANSITIONS DETECTOR FAILURE (BIT #5)
2101            000100          TUF=    100             ;TRANSITIONS UNSAFE (BIT #6)
2102            000200          ABS=    200             ;ABNORMAL STOP (BIT #7)
2103            000400          WRU=    400             ;WRITE READY UNSAFE (BIT #8)
2104            001000          MHS=    1000            ;MUTLTIPLE HEAD SELECT (BIT #9)
2105            002000          NHS=    2000            ;NO HEAD SELECTION (BIT #10)
2106            004000          IXE=    4000            ;INDEX ERROR (BIT #11)
2107            020000          PLU=    20000           ;PLO UNSAFE (BIT #12)
2108
2109                            ;OFFSET REGISTER (RPOF) (#11)
2110
2111            000001          OF25=   1               ;OFFSET 25 MICRO INCHES (BIT #0)
2112            000002          OF50=   2               ;OFFSET 50 MICRO INCHES (BIT #1)
2113            000004          OF100=  4               ;OFFSET 100 MICRO INCHES (BIT #2)
2114            000010          OF200=  10              ;OFFSET 200 MICRO INCHES (BIT #3)
2115            000020          OF400=  20              ;OFFSET 400 MICRO INCHES (BIT #4)
2116            000040          OF800=  40              ;OFFSET 800 MICRO INCHES (BIT #5)
2117            000200          OFREV=  200             ;OFFSET NEGATIVE (REVERSE) (BIT #5)
2118            002000          HCI=    2000            ;HEADER COMPARE INHIBIT (BIT #10)
2119            004000          ECI=    4000            ;ERROR CORRECTION CODE INHIBIT (BIT #11)
2120            010000          FMT22=  10000           ;FORMAT BIT (BIT #12
2121
2122                            ;DESIRED CYLINDER ADDRESS (RPCA) (#12)
2123                            ;(EACH BIT IS CALLED BY BIT NUMBER)
2124
2125                            ;CURRENT CYLINDER ADDRESS (RPCC) (#13)
2126                            ;(EACH BIT IS CALLED BY BIT NUMBER)
2127
2128                            ;SERIAL NUMBER REGISTER (RPSN) (#14)
2129                            ;(EACH IS CALLED BY BIT NUMBER)
2130
2131                            ;RP04 ERROR REGISTER #03 (RPER3) (#15)
```

```
2132
2133  000001                  PSU=    1                       ;PACK SPEED UNSAFE (BIT #0)
2134  000002                  VUF=    2                       ;VELOCITY UNSAFE (BIT #1)
2135  000010                  UWR=    10                      ;ANY UNSAFE EXCEPT READ/WRITE (BIT #3)
2136  000040                  ACL=    40                      ;AC LOW (BIT #5)
2137  000100                  DCL=    100                     ;DC LOW (BIT #6)
2138  040000                  SKI=    40000                   ;SEEK INCOMPLETE (BIT #14)
2139  100000                  OCYL=   100000                  ;OFF CYLINDER (BIT #15)
2140
2141                          ;RP05/6 ERROR REGISTER #03 (RPER3) (#15)
2142
2143  000001                  DCU=    1                       ;DC UNSAFE (BIT #0)
2144  000002                  WAO=    2                       ;WRITE AND OFFSET (BIT #1)
2145  000040                  ACL=    40                      ;AC LOW (BIT #5)
2146  000100                  DCL=    100                     ;DC LOW (BIT #6)
2147  020000                  OPE=    20000                   ;OPERATOR PLUG ERROR (BIT #13)
2148  040000                  SKI=    40000                   ;SEEK INCOMPLETE (BIT #14)
2149  100000                  OCYL=   100000                  ;OFF CYLINDER ERROR (BIT #15)
2150
2151                          ;ECC POSITION REGISTER (RPEC1) (#16)
2152                          ;(EACH BIT IS CALLED BY BIT NUMBER)
2153
2154                          ;ECC PATTERN REGISTER (RPEC2) (#17)
2155                          ;(EACH BIT IS CALLED BY BIT NUMBER)
2156
2157                          ;;*****************************************************************
2158
2159                          .SBTTL  RP04/5/6 DRIVER COMMANDS
2160
2161                          ;;*****************************************************************
2162
2163  000101                  RNOP    =       101             ;NO OPERATION
2164  000103                  UNLOAD  =       103             ;UNLOAD
2165  000105                  SEEK    =       105             ;SEEK
2166  000107                  RECAL   =       107             ;RECALIBRATE
2167  000111                  DRVCLR  =       111             ;DRIVE CLEAR
2168  000113                  RELSE   =       113             ;RELEASE
2169  000115                  OFFSET  =       115             ;OFFSET
2170  000117                  RTC     =       117             ;RETURN TO CENTER LINE
2171  000121                  READIN  =       121             ;READ IN PRESET
2172  000123                  ACK     =       123             ;PACK ACKNOWLEDGE
2173  000131                  SEARCH  =       131             ;SEARCH
2174  000141                  GETREG  =       141             ;GET REGISTERS
2175  000143                  SETFMT  =       143             ;SET FORMAT (8 ECI OR HCI)
2176  000145                  SELDRV  =       145             ;SELECT DRIVE
2177  000151                  WCKD    =       151             ;WRITE CHECK DATA
2178  000153                  WCKHD   =       153             ;WRITE CHECK HEADER & DATA
2179  000161                  WRTDAT  =       161             ;WRITE DATA
2180  000163                  WRTHD   =       163             ;WRITE HEADER & DATA
2181  000171                  RDDAT   =       171             ;READ DATA
2182  000173                  RDHD    =       173             ;READ HEADER & DATA
2183
2184                          .SBTTL  TRAP CATCHER
2185
2186  000000                          .=0
2187                          ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
```

```
2188                                    ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
2189                                    ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
2190            000174                          .=174
2191  000174   000000            DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
2192  000176   000000            SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
2193                                    .SBTTL  STARTING ADDRESS(ES)
2194  000200   000137  004116            JMP     @#START1        ;;JUMP TO STARTING ADDRESS OF PROGRAM
2195  000204   000137  004106            JMP     @#START         ;CHANGE THE RH11 UNIBUS ADDRESS
2196                                                            ;AFTER INITIAL START
2197
2198                                    .SBTTL  ACT11 HOOKS
2199
2200                                    ;;*****************************************************************
2201                                    ;HOOKS REQUIRED BY ACT11
2202            000210                          $SVPC=.                 ;SAVE PC
2203            000046                          .=46
2204  000046   005420                          $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
2205            000052                          .=52
2206  000052   040000                          .WORD   40000           ;;2)SET LOC.52 TO 40000
2207            000210                          .=$SVPC                 ;; RESTORE PC
```

```
                                .SBTTL  COMMON TAGS

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

                        001100          .=1100
001100                  $CMTAG:                         ;;START OF COMMON TAGS
001100  000000          $PASS:  .WORD   0               ;;CONTAINS PASS COUNT
001102  000             $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
001103  000             $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
001104  000000          $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
001106  000000          $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
001110  000000          $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
001112  000000          $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
001114  000             $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
001115  001             $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
001116  000000          $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001120  000000          $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
001122  000000          $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
001124  000000          $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
001126  000000          $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
001130  000000                  .WORD   0               ;;RESERVED--NOT TO BE USED
001132  000000                  .WORD   0
001134  000             $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
001135  000             $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
001136  000000                  .WORD   0
001140  177570          SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
001142  177570          DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
001144  177560          $TKS:   177560                  ;;TTY KBD STATUS
001146  177562          $TKB:   177562                  ;;TTY KBD BUFFER
001150  177564          $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
001152  177566          $TPB:   177566                  ;;TTY PRINTER BUFFER R_G. ADDRESS
001154  000             $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
001155  002             $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001156  012             $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001157  000             $TPFLG: .BYTE   0               ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
001160  177607  000377  $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
001164  077             $QUES:  .ASCII  /?/             ;;QUESTION MARK
001165  015             $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001166  000012          $LF:    .ASCIZ  <12>            ;;LINE FEED
                                ;;*****************************************************************
                        000015  CR      =       15
                        000012  LF      =       12
001170  176700          $RPADR: .WORD   176700          ;FIRST ADDRESS OF RH11/RP04/5/6 REGISTERS
001172  000254          $RPVEC: .WORD   254             ;RP04 VECTOR ADDRESS
001174  172540          $LKCSR: .WORD   172540          ;ADDR OF KW11-P STATUS REGISTER
001176  172542          $LKCSB: .WORD   172542          ;ADDR OF KW11-P COUNTER BUFFER
001200  000104          $LPVEC: .WORD   104             ;ADDR OF KW11-P VECTOR
001202  177546          $LKS:   .WORD   177546          ;ADDR OF KW11-L STATUS REGISTER
001204  000100          $LLVEC: .WORD   100             ;ADDR OF KW11-L VECTOR
001206  177777          PCLOCK: .WORD   -1              ;'0' IF KW11-P IS ON SYSTEM
001210  177777          CLKFLG: .WORD   -1              ;'0' IF A CLOCK IS AVAILABLE
001212  000074          HZ:     .WORD   74              ;74 (8) IF 60 HZ SYSTEM; 62 (8) IF 50 HZ SYSTEM
001214  000000          STATIN: .WORD   0               ;'TYPE STATISTICS' INDICATOR
001216  000000          PACK:   .WORD   0               ;'W ' COMMAND INDICATOR
```

```
2264  001220  000000  000000  000000  DATE:   .WORD   0,0,0,0,0       ;OPERATOR ENTERED DATE
2265  001226  000000  000000
2266  001232  000000  000000  000000  OPERID: .WORD   0,0,0,0         ;OPERATOR ID
2267  001240  000000
2268  001242  000000                  DRIVE:  .WORD   0               ;DRIVE # STORAGE: ERRORS 1-5 & 10
2269  001244  000000                  ATTN:   .WORD   0               ;ATTN REG STORAGE: ERRORS 1-5 & 10
2270  001246  000000                  UNIT:   .WORD   0               ;DRIVE # STORAGE FOR PRINTOUT
2271  001250  000000                  MASK:   .WORD   0               ;ERROR RETRY REGISTER MASK
2272  001252     000     000          RETRY:  .BYTE   0,0             ;ERROR RETRY LIMIT IN THE LOWER BYTE
2273                                                                  ;RETRY COUNT IN THE UPPER BYTE
2274  001254  000003                  FAIRNS: .WORD   3               ;MAXIMUM TIME IN QUEUE VALUE
2275  001256  000000                  LSTAD:  .WORD   0               ;STORE LAST MEMORY ADDRESS HERE
2276  001260  000000                  CHGADR: .WORD   0               ;CHANGE RH11 UNIBUS ADDRESS FLAG
2277  001262  000000                  CFLAG:  .WORD   0               ;'CONTROL C' FLAG
2278  001264  000000                  BADSEC: .WORD   0               ;BAD SECTOR/TRACK FLAG
2279  001266  000000                  HOUR:   .WORD   0               ;HOUR COUNT STORED HERE (MAXIMUM - 999.)
2280  001270  000000                  MINUTE: .WORD   0               ;MINUTE'S COUNT STORED HERE
2281  001272  000000                  SECOND: .WORD   0               ;SECOND'S COUNT STORED HERE
2282  001274  000000                  SIXTEE: .WORD   0               ;TIMER ROUTINE COUNTER (FOR ONE SECOND)
2283  001276  177777                  ZROIND: .WORD   -1              ;ZERO INDICATOR FOR THE DATA COMPARE ROUTINE
2284  001300     000                  FRSTER: .BYTE   0               ;DATA COMPARE ERROR FLAG
2285                                                                  ;IF > 0, PROCESSING 'DCKER' OR CAN'T MATCH PATTERN
2286                                                                  ;IF < 0, MISCOMPARSION FOUND
2287  001301     000                          .BYTE   0               ;MISCOMPARSION OR CAN'T MATCH PATTERN FLAG
2288                                                                  ;IF < 0, ERROR IN BUFFER
2289  001302  000000                  SAVER1: .WORD   0               ;SAVE R1 HERE
2290  001304  000000                  SAVER5: .WORD   0               ;SAVE R5 HERE
2291  001306  000000                  ERCTR:  .WORD   0               ;NUMBER OF ERRORS
2292  001310  000000                  LIMIT:  .WORD   0               ;DISPLAY LIMIT
2293  001312  000000                  CMCNT:  .WORD   0               ;WORD COUNT
2294  001314  000000                  CMCYL:  .WORD   0               ;CYLINDER ADDRESS
2295  001316     000                  CMSEC:  .BYTE   0               ;SECTOR ADDRESS
2296  001317     000                  CMTRK:  .BYTE   0               ;TRACK ADDRESS
2297  001320  000000                  ECBIT:  .WORD   0               ;ERROR BURST BIT OFFSET
2298  001322  000000                  ECSEC:  .WORD   0               ;ERROR BURST WORD OFFSET (RELATIVE TO SECTOR)
2299  001324  000000                  ECMSK0: .WORD   0               ;CORRECTION MASK FOR FIRST ERROR WORD
2300  001326  000000                  ECMSK1: .WORD   0               ;CORRECTION MASK FOR SECOND ERROR WORD
2301  001330  000000                  ECWRD:  .WORD   0               ;LOCATION OF FIRST ERROR WORD
2302  001332  000000                  ECGD:   .WORD   0               ;GOOD DATA, FIRST WORD
2303  001334  000000                  ECBAD0: .WORD   0               ;BAD DATA, FIRST WORD
2304  001336  000000                  ECWRD1: .WORD   0               ;LOCATION OF SECOND ERROR WORD
2305  001340  000000                  ECGD1:  .WORD   0               ;GOOD DATA, SECOND WORD
2306  001342  000000                  ECBAD1: .WORD   0               ;BAD DATA, SECOND WORD
2307  001344  000025                  SECLMT: .WORD   21.             ;SECTOR ADDRESS LIMIT
2308  001346  000022                  TRKLMT: .WORD   18.             ;TRACK ADDRESS LIMIT
2309  001350  000632                  CYLIMT: .WORD   410.            ;CYLINDER ADDRESS LIMIT FOR RP04/5'S
2310                                                                  ;(CHANGED TO 814. FOR RP06)
2311
2312                          ;;*****************************************************************
2313
2314                          .SBTTL  COMMON PARAMETERS
2315
2316                          ;;*****************************************************************
2317
2318                          ;PROGRAM USES THESE PARAMETERS TO DETERMINE REGULAR END OF PASS
2319  001352  002740          ENDCN:  .WORD   002740          ;1.875X10↑8 WORDS (10) [3X10↑9 BITS]
```

```
2320  001354  005455                          .WORD   005455          ;MSW
2321  001356  143300                ENDSK:    .WORD   143300          ;3 X 10↑6 SEEKS  (LSW)
2322  001360  000055                          .WORD   55              ;MSW
2323                                 ;PROGRAM USES THESE PARAMETERS TO DETERMINE Q.V. END OF PASS
2324  001362  120274                QVCON:    .WORD   120274          ;2.3437X10↑7 WORDS (10)
2325  001364  000005                          .WORD   5               ;MSW
2326  001366  134330                QVSEK:    .WORD   134330          ;3.75 X 10↑5 SEEKS (10)
2327  001370  000005                          .WORD   5               ;MSW
2328                                 ;THE NUMBERS TO DETERMINE END OF PASS ARE LOADED IN HERE BY THE PROGRAM.
2329                                 ;THE FIRST TIME THROUGH, THE QV PARAMETERS ARE USED, AFTER THAT THE
2330                                 ;REGULAR PARAMETERS ARE USED.
2331  001372  000000                ENDCON:   .WORD   0
2332  001374  000000                          .WORD   0
2333  001376  000000                ENDSEK:   .WORD   0
2334  001400  000000                          .WORD   0
2335
2336  001402  000001                PASCNT:   .WORD   1               ;NUMBER OF PASSES TO END OF TEST
2337  001404  000000                MAXDL:    .WORD   0               ;MAXIMUM DATA TRANFER SIZE IN WORDS
2338                                                                   ;(FILLED BY PROGRAM AT STARTUP OR BY OPERATOR
2339                                                                   ;DURING PARAMETER ENTRY DIALOG.)
2340  001406  000144                MAXER:    .WORD   100.            ;MAXIMUM ERRORS - 100(10)
2341  001410  000005  000000        INTRVL:   .WORD   5,0             ;FIRST WORD IS THE PERFORMANCE TYPEOUT INTERVAL
2342                                                                   ;(IN MINUTES).  SECOND WORD IS THE INTERVAL
2343                                                                   ;COUNTER
2344                                                                   ;COUNTER.  UPPER BYTE IS VALUE.
2345  001414  000004                CMPLMT:   .WORD   4               ;NUMBER OF COMPARE ERRORS TYPED OUT
2346  001416  000001                FORMAT:   .WORD   1               ;IF NOT EQ 0, ALLOW WRITE HEADER & DATA ORDERS
2347                                                                   ;IF EQ 0, DO NOT ALLOW WRITE HEADER & DATA ORDERS
2348  001420  000000                WCSEL:    .WORD   0               ;IF EQ TO 0, GENERATE A RANDOM WORD COUNT
2349                                                                   ;  FOR THE OPERATION.
2350                                                                   ;IF NOT EQ TO 0, USE THE VALUE IN 'MAXDL' FOR
2351                                                                   ;  THE WORD COUNT
2352  001422  000003                RATIO:    .WORD   3               ;READ/WRITE RATIO [RANGE 0 - 7]
2353                                                                   ;0 - 0/8        (READ/WRITE)
2354                                                                   ;1 - 7/1
2355                                                                   ;2 - 6/2
2356                                                                   ;3 - 5/3
2357                                                                   ;4 - 4/4
2358                                                                   ;5 - 3/5
2359                                                                   ;6 - 2/6
2360                                                                   ;7 - 1/7
2361  001424  000001                AUTOCK:   .WORD   1               ;IF NOT EQ 0, DO AN APPROPRITE WRITE
2362                                                                   ;  CHECK AFTER EACH WRITE ORDER.
2363                                                                   ;IF EQ 0, SELECT WRITE CHECK ORDERS
2364                                                                   ;  RANDOMLY.
2365  001426  000001                NOTPRT:   .WORD   1               ;IF EQ 1, DO NOT PRINT DATA ERROR MESSAGES
2366                                                                   ;  ASSOCIATED WITH OPERATOR SPECIFIED
2367                                                                   ;  BAD PACK AREAS.
2368                                                                   ;IF NOT EQ 0, PRINT ERROR MESSAGES RELATING TO
2369                                                                   ;  THESE AREAS.
2370  001430  000001                ENDET:    .WORD   1               ;IF NOT EQ 0, END OF PASS DETERMINED
2371                                                                   ;  BY THE 'WORDS READ' COUNT.
2372                                                                   ;IF EQ 0, END OF PASS DETERMINED
2373                                                                   ;  BY THE SEEK COUNT.
2374
2375                                 ;;*****************************************************************
```

```
2376
2377                                    .SBTTL  VALUES FOR FIRST OPERATION
2378
2379                                    ;;**********************************************************************
2380
2381  001432  000010                   BEGPAT: .WORD   10              ;STARTING PATTERN CODE [RANGE 1 - 17 (OCTAL)]
2382  001434  000005                   BEGCOD: .WORD   5               ;STARTING COMMAND CODE [RANGE 0 - 5]
2383                                                                    ;0 = WRITE CHECK DATA ('WCKD')
2384                                                                    ;1 = WRITE CHECK HEADER & DATA ('WCHKHD')
2385                                                                    ;2 = WRITE DATA ('WRTDAT')
2386                                                                    ;3 = WRITE HEADER & DATA ('WRTHD')
2387                                                                    ;4 = READ DATA ('RDDAT')
2388                                                                    ;5 = READ HEADER & DATA ('RDHD')
2389  001436  000404                   BEGSIZ: .WORD   404             ;STARTING RECORD SIZE [RANGE 4 - MAXMEM]
2390                                                                    ;NOTE:  THE SIZE MUST BE AT LEAST 4 IF
2391                                                                    ;WRITE DATA OR READ DATA; THE SIZE MUST
2392                                                                    ;BE AT LEAST 8 IF WRITE HEADER AND
2393                                                                    ;DATA OR READ HEADER AND DATA.
2394                                                                    ;IF THE SIZE IS GREATER THAN 1 SECTOR, THE
2395                                                                    ;SIZE MUST ALLOW FOR OVERLAPPING 4 OR 8
2396                                                                    ;WORDS INTO THE LAST SECTOR USED.
2397
2398                                    ;;**********************************************************************
2399
2400                                    .SBTTL  TABLES, CONSTANTS, AND VARIABLE LOCATIONS
2401
2402                                    ;;**********************************************************************
2403
2404  001440  000000  000000  000000   ORDERQ: .WORD   0,0,0,0,0,0,0,0,0  ;LIST OF DRIVES PERFORMING COMMANDS
2405  001446  000000  000000  000000
2406  001454  000000  000000  000000
2407
2408  001462  000000                   ASNLST: .WORD   0               ;A BIT SET IS AN ASSIGNED DRIVE
2409
2410  001464  000000  000000  000000   DUNIT:  .WORD   0,0,0,0,0,0,0,0,0       ;ADDRESSES OF DRIVES TO BE DEASSIGNED
2411  001472  000000  000000  000000
2412  001500  000000  000000  000000
2413
2414  001506  000000  000000  000000   NEWUNT: .WORD   0,0,0,0,0,0,0,0,0       ;ADDRESSES OF NEWLY ASSIGNED DRIVES
2415  001514  000000  000000  000000
2416  001522  000000  000000  000000
2417
2418  001530  000000  000000  000000   AVAIL:  .WORD   0,0,0,0,0,0,0,0,0       ;LIST OF DRIVES WAITING FOR BUFFERS/PARAMETERS
2419  001536  000000  000000  000000
2420  001544  000000  000000  000000
2421
2422  001552  000000  000000  000000   WAIT:   .WORD   0,0,0,0,0,0,0,0,0       ;LIST OF DRIVES WAITING FOR BUFFERS
2423  001560  000000  000000  000000
2424  001566  000000  000000  000000
2425
2426  001574  000000  000000  000000   PARQ:   .WORD   0,0,0,0,0,0,0,0,0       ;LIST OF DRIVES WAITING FOR NEXT PARAMETERS
2427  001602  000000  000000  000000
2428  001610  000000  000000  000000
2429
2430  001616  000000                   BUFTBL: .WORD   0               ;BUFFER ALLOCATION TABLE ENTRY COUNT
2431  001620  000000  000000                   .WORD   0,0
```

```
2432  001624  000000  000000                .WORD   0,0
2433  001630  000000  000000                .WORD   0,0
2434  001634  000000  000000                .WORD   0,0
2435  001640  000000  000000                .WORD   0,0
2436  001644  000000  000000                .WORD   0,0
2437  001650  000000  000000                .WORD   0,0
2438  001654  000000  000000                .WORD   0,0
2439  001660  000000  000000                .WORD   0,0
2440  001664  000000  000000                .WORD   0,0
2441  001670  000000  000000                .WORD   0,0
2442  001674  000000  000000                .WORD   0,0
2443  001700  000000  000000                .WORD   0,0
2444  001704  000000  000000                .WORD   0,0
2445  001710  000000  000000                .WORD   0,0
2446  001714  000000  000000                .WORD   0,0
2447  001720  000000  000000                .WORD   0,0
2448  001724  000000  000000                .WORD   0,0
2449  001730  000000  000000                .WORD   0,0
2450  001734  000000  000000                .WORD   0,0
2451
2452  001740  041730              BLKADR:   .WORD   DRIVE0          ;ADDRESS OF THE BLOCK FOR DRIVE 0
2453  001742  042234                        .WORD   DRIVE1          ;ADDRESS OF THE BLOCK FOR DRIVE 1
2454  001744  042540                        .WORD   DRIVE2          ;ADDRESS OF THE BLOCK FOR DRIVE 2
2455  001746  043044                        .WORD   DRIVE3          ;ADDRESS OF THE BLOCK FOR DRIVE 3
2456  001750  043350                        .WORD   DRIVE4          ;ADDRESS OF THE BLOCK FOR DRIVE 4
2457  001752  043654                        .WORD   DRIVE5          ;ADDRESS OF THE BLOCK FOR DRIVE 5
2458  001754  044160                        .WORD   DRIVE6          ;ADDRESS OF THE BLOCK FOR DRIVE 6
2459  001756  044464                        .WORD   DRIVE7          ;ADDRESS OF THE BLOCK FOR DRIVE 7
2460
2461  001760     151              COMTBL:   .BYTE   WCKD            ;WRITE CHECK DATA
2462  001761     153                        .BYTE   WCKHD           ;WRITE CHECK HEADER AND DATA
2463  001762     161                        .BYTE   WRTDAT          ;WRITE DATA
2464  001763     163                        .BYTE   WRTHD           ;WRITE HEADER AND DATA
2465  001764     171                        .BYTE   RDDAT           ;READ DATA
2466  001765     173                        .BYTE   RDHD            ;READ HEADER AND DATA
2467
2468  001766     002              OPTBL:    .BYTE   2               ;UNLOAD
2469  001767     004                        .BYTE   4               ;SEEK
2470  001770     006                        .BYTE   6               ;RECAL
2471  001771     010                        .BYTE   10              ;DRIVE CLEAR
2472  001772     012                        .BYTE   12              ;RELEASE
2473  001773     014                        .BYTE   14              ;OFFSET
2474  001774     016                        .BYTE   16              ;RETURN TO CENTERLINE
2475  001775     020                        .BYTE   20              ;READIN PRESET
2476  001776     022                        .BYTE   22              ;PACK ACKNOWLEDGE
2477  001777     030                        .BYTE   30              ;SEARCH
2478  002000     050                        .BYTE   50              ;WRITE CHECK DATA
2479  002001     052                        .BYTE   52              ;WRITE CHECK HEADER AND DATA
2480  002002     060                        .BYTE   60              ;WRITE DATA
2481  002003     062                        .BYTE   62              ;WRITE HEADER AND DATA
2482  002004     070                        .BYTE   70              ;READ DATA
2483  002005     072                        .BYTE   72              ;READ HEADER AND DATA
2484  002006     377                        .BYTE   -1              ;TERMINATOR
2485
2486          002010              .EVEN
2487
```

```
2488  002010  047125  047514  042101  MNTBL:  .ASCIZ  /UNLOAD /
2489  002016  000040
2490  002020  042523  045505  020040          .ASCIZ  /SEEK   /
2491  002026  000040
2492  002030  042522  040503  020114          .ASCIZ  /RECAL  /
2493  002036  000040
2494  002040  051104  041526  051114          .ASCIZ  /DRVCLR /
2495  002046  000040
2496  002050  042522  051514  020105          .ASCIZ  /RELSE  /
2497  002056  000040
2498  002060  043117  051506  052105          .ASCIZ  /OFFSET /
2499  002066  000040
2500  002070  052122  020103  020040          .ASCIZ  /RTC    /
2501  002076  000040
2502  002100  042522  042101  047111          .ASCIZ  /READIN /
2503  002106  000040
2504  002110  040520  045503  020040          .ASCIZ  /PACK   /
2505  002116  000040
2506  002120  042523  051101  044103          .ASCIZ  /SEARCH /
2507  002126  000040
2508  002130  041527  042113  020040          .ASCIZ  /WCKD   /
2509  002136  000040
2510  002140  041527  044113  020104          .ASCIZ  /WCKHD  /
2511  002146  000040
2512  002150  051127  042124  052101          .ASCIZ  /WRTDAT /
2513  002156  000040
2514  002160  051127  044124  020104          .ASCIZ  /WRTHD  /
2515  002166  000040
2516  002170  042122  040504  020124          .ASCIZ  /RDDAT  /
2517  002176  000040
2518  002200  042122  042110  020040          .ASCIZ  /RDHD   /
2519  002206  000040
2520  002210  047516  042516  020040          .ASCIZ  /NONE   /
2521  002216  000040
2522
2523  002220      000         OFFCOD: .BYTE   0               ;OFFSET CODE TABLE
2524  002221      010                 .BYTE   10              ;+200 U INCHES
2525  002222      210                 .BYTE   210             ;-200 U INCHES
2526  002223      020                 .BYTE   20              ;+400 U INCHES
2527  002224      220                 .BYTE   220             ;-400 U INCHES
2528  002225      030                 .BYTE   30              ;+600 U INCHES
2529  002226      230     000         .BYTE   230,0           ;-600 U INCHES, TERMINATOR
2530  002230      020                 .BYTE   20              ;+400 U INCHES
2531  002231      220                 .BYTE   220             ;-400 U INCHES
2532  002232      040                 .BYTE   40              ;+800 U INCHES
2533  002233      240                 .BYTE   240             ;-800 U INCHES
2534  002234      060                 .BYTE   60              ;+1200 U INCHES
2535  002235      260     000         .BYTE   260,0           ;-1200 U INCHES, TERMINATOR
2536          002240          .EVEN
2537
2538  002240  002274          OFMTBL: .WORD   OFMSG0          ;1ST OFFSET MESSAGE
2539  002242  002327                  .WORD   OFMSG1          ;2ND OFFSET MESSAGE
2540  002244  002363                  .WORD   OFMSG2          ;3RD OFFSET MESSAGE
2541  002246  002417                  .WORD   OFMSG3          ;4TH OFFSET MESSAGE
2542  002250  002453                  .WORD   OFMSG4          ;5TH OFFSET MESSAGE
2543  002252  002507                  .WORD   OFMSG5          ;6TH OFFSET MESSAGE
```

```
2544  002254  002543                          .WORD   OFMSG6          ;7TH OFFSET MESSAGE
2545  002256  002274                          .WORD   OFMSG0          ;1ST OFFSET MESSAGE
2546  002260  002417                          .WORD   OFMSG3          ;4TH OFFSET MESSAGE
2547  002262  002453                          .WORD   OFMSG4          ;5TH OFFSET MESSAGE
2548  002264  002577                          .WORD   OFMSG7          ;8TH OFFSET MESSAGE
2549  002266  002633                          .WORD   OFMSG8          ;9TH OFFSET MESSAGE
2550  002270  002667                          .WORD   OFMSG9          ;10TH OFFSET MESSAGE
2551  002272  002724                          .WORD   OFMSGA          ;11TH OFFSET MESSAGE
2552
2553  002274  043101  042524  020122  OFMSG0: .ASCIZ  /AFTER RETRY WITHOUT OFFSET/
2554  002302  042522  051124  020131
2555  002310  044527  044124  052517
2556  002316  020124  043117  051506
2557  002324  052105     000
2558  002327     101  020124  043117  OFMSG1: .ASCIZ  /AT OFFSET +200 MICRO-INCHES/
2559  002334  051506  052105  025440
2560  002342  030062  020060  044515
2561  002350  051103  026517  047111
2562  002356  044103  051505     000
2563  002363     101  020124  043117  OFMSG2: .ASCIZ  /AT OFFSET -200 MICRO-INCHES/
2564  002370  051506  052105  026440
2565  002376  030062  020060  044515
2566  002404  051103  026517  047111
2567  002412  044103  051505     000
2568  002417     101  020124  043117  OFMSG3: .ASCIZ  /AT OFFSET +400 MICRO-INCHES/
2569  002424  051506  052105  025440
2570  002432  030064  020060  044515
2571  002440  051103  026517  047111
2572  002446  044103  051505     000
2573  002453     101  020124  043117  OFMSG4: .ASCIZ  /AT OFFSET -400 MICRO-INCHES/
2574  002460  051506  052105  026440
2575  002466  030064  020060  044515
2576  002474  051103  026517  047111
2577  002502  044103  051505     000
2578  002507     101  020124  043117  OFMSG5: .ASCIZ  /AT OFFSET +600 MICRO-INCHES/
2579  002514  051506  052105  025440
2580  002522  030066  020060  044515
2581  002530  051103  026517  047111
2582  002536  044103  051505     000
2583  002543     101  020124  043117  OFMSG6: .ASCIZ  /AT OFFSET -600 MICRO-INCHES/
2584  002550  051506  052105  026440
2585  002556  030066  020060  044515
2586  002564  051103  026517  047111
2587  002572  044103  051505     000
2588  002577     101  020124  043117  OFMSG7: .ASCIZ  /AT OFFSET +800 MICRO-INCHES/
2589  002604  051506  052105  025440
2590  002612  030070  020060  044515
2591  002620  051103  026517  047111
2592  002626  044103  051505     000
2593  002633     101  020124  043117  OFMSG8: .ASCIZ  /AT OFFSET -800 MICRO-INCHES/
2594  002640  051506  052105  026440
2595  002646  030070  020060  044515
2596  002654  051103  026517  047111
2597  002662  044103  051505     000
2598  002667     101  020124  043117  OFMSG9: .ASCIZ  /AT OFFSET +1200 MICPO-INCHES/
2599  002674  051506  052105  025440
```

```
2600  002702  031061  030060  046440
2601  002710  041511  047522  044455
2602  002716  041516  042510  000123
2603  002724  052101  047440  043106  OFMSGA: .ASCIZ  /AT OFFSET -1200 MICRO-INCHES/
2604  002732  042523  020124  030455
2605  002740  030062  020060  044515
2606  002746  051103  026517  047111
2607  002754  044103  051505     000
2608
2609          002762                          .EVEN
2610
2611                                  ;;*****************************************************************
2612
2613                                  .SBTTL  DATA PATTERNS
2614
2615                                  ;;*****************************************************************
2616
2617  002762  000000                 STNDAT: .WORD   0               ;STANDARD DATA PATTERN POINTER TABLE
2618  002764  003066                         .WORD   DATA1
2619  002766  003126                         .WORD   DATA1+40
2620  002770  003166                         .WORD   DATA1+100
2621  002772  003226                         .WORD   DATA1+140
2622  002774  003266                         .WORD   DATA1+200
2623  002776  003326                         .WORD   DATA1+240
2624  003000  003366                         .WORD   DATA1+300
2625  003002  003426                         .WORD   DATA1+340
2626  003004  003466                         .WORD   DATA1+400
2627  003006  003526                         .WORD   DATA1+440
2628  003010  003566                         .WORD   DATA1+500
2629  003012  003626                         .WORD   DATA1+540
2630  003014  003666                         .WORD   DATA1+600
2631  003016  003726                         .WORD   DATA1+640
2632  003020  003766                         .WORD   DATA1+700
2633  003022  003026                         .WORD   DATA0           ;ZEROES
2634  003024  003730                         .WORD   DATA1+642       ;ONES
2635
2636  003026  000000                 DATA0:  .WORD   0               ;DUMMY DATA PATTERN
2637  003030  000000                         .WORD   0
2638  003032  000000                         .WORD   0
2639  003034  000000                         .WORD   0
2640  003036  000000                         .WORD   0
2641  003040  000000                         .WORD   0
2642  003042  000000                         .WORD   0
2643  003044  000000                         .WORD   0
2644  003046  000000                         .WORD   0
2645  003050  000000                         .WORD   0
2646  003052  000000                         .WORD   0
2647  003054  000000                         .WORD   0
2648  003056  000000                         .WORD   0
2649  003060  000000                         .WORD   0
2650  003062  000000                         .WORD   0
2651  003064  000000                         .WORD   0
2652
2653  003066  000001                 DATA1:  .WORD   000001  ;STANDARD PATTERN 1
2654  003070  000003                         .WORD   000003
2655  003072  000007                         .WORD   000007
```

```
2656  003074  000017                          .WORD   000017
2657  003076  000037                          .WORD   000037
2658  003100  000077                          .WORD   000077
2659  003102  000177                          .WORD   000177
2660  003104  000377                          .WORD   000377
2661  003106  000777                          .WORD   000777
2662  003110  001777                          .WORD   001777
2663  003112  003777                          .WORD   003777
2664  003114  007777                          .WORD   007777
2665  003116  017777                          .WORD   017777
2666  003120  037777                          .WORD   037777
2667  003122  077777                          .WORD   077777
2668  003124  177777                          .WORD   177777
2669
2670  003126  177776                          .WORD   177776  ;STANDARD PATTERN 2
2671  003130  177774                          .WORD   177774
2672  003132  177770                          .WORD   177770
2673  003134  177760                          .WORD   177760
2674  003136  177740                          .WORD   177740
2675  003140  177700                          .WORD   177700
2676  003142  177600                          .WORD   177600
2677  003144  177400                          .WORD   177400
2678  003146  177000                          .WORD   177000
2679  003150  176000                          .WORD   176000
2680  003152  174000                          .WORD   174000
2681  003154  170000                          .WORD   170000
2682  003156  160000                          .WORD   160000
2683  003160  140000                          .WORD   140000
2684  003162  100000                          .WORD   100000
2685  003164  000000                          .WORD   000000
2686
2687  003166  000000                          .WORD   000000  ;STANDARD PATTERN 3
2688  003170  000000                          .WORD   000000
2689  003172  000000                          .WORD   000000
2690  003174  177777                          .WORD   177777
2691  003176  177777                          .WORD   177777
2692  003200  177777                          .WORD   177777
2693  003202  000000                          .WORD   000000
2694  003204  000000                          .WORD   000000
2695  003206  177777                          .WORD   177777
2696  003210  177777                          .WORD   177777
2697  003212  000000                          .WORD   000000
2698  003214  177777                          .WORD   177777
2699  003216  000000                          .WORD   000000
2700  003220  177777                          .WORD   177777
2701  003222  000000                          .WORD   000000
2702  003224  177777                          .WORD   177777
2703
2704  003226  000000                          .WORD   000000  ;STANDARD PATTERN 4
2705  003230  010421                          .WORD   010421
2706  003232  021042                          .WORD   021042
2707  003234  031463                          .WORD   031463
2708  003236  042104                          .WORD   042104
2709  003240  052525                          .WORD   052525
2710  003242  063146                          .WORD   063146
2711  003244  073567                          .WORD   073567
```

```
2712  003246  104210                          .WORD   104210
2713  003250  114631                          .WORD   114631
2714  003252  125252                          .WORD   125252
2715  003254  135673                          .WORD   135673
2716  003256  146314                          .WORD   146314
2717  003260  156735                          .WORD   156735
2718  003262  167356                          .WORD   167356
2719  003264  177777                          .WORD   177777
2720
2721  003266  052525                          .WORD   052525  ;STANDARD PATTERN 5
2722  003270  052525                          .WORD   052525
2723  003272  052525                          .WORD   052525
2724  003274  125252                          .WORD   125252
2725  003276  125252                          .WORD   125252
2726  003300  125252                          .WORD   125252
2727  003302  052525                          .WORD   052525
2728  003304  052525                          .WORD   052525
2729  003306  125252                          .WORD   125252
2730  003310  125252                          .WORD   125252
2731  003312  052525                          .WORD   052525
2732  003314  125252                          .WORD   125252
2733  003316  052525                          .WORD   052525
2734  003320  125252                          .WORD   125252
2735  003322  052525                          .WORD   052525
2736  003324  125252                          .WORD   125252
2737
2738  003326  007417                          .WORD   007417  ;STANDARD PATTERN 6
2739  003330  007417                          .WORD   007417
2740  003332  007417                          .WORD   007417
2741  003334  170360                          .WORD   170360
2742  003336  170360                          .WORD   170360
2743  003340  170360                          .WORD   170360
2744  003342  007417                          .WORD   007417
2745  003344  007417                          .WORD   007417
2746  003346  170360                          .WORD   170360
2747  003350  170360                          .WORD   170360
2748  003352  007417                          .WORD   007417
2749  003354  170360                          .WORD   170360
2750  003356  007417                          .WORD   007417
2751  003360  170360                          .WORD   170360
2752  003362  007417                          .WORD   007417
2753  003364  170360                          .WORD   170360
2754
2755  003366  026455                          .WORD   026455  ;STANDARD PATTERN 7
2756  003370  026455                          .WORD   026455
2757  003372  026455                          .WORD   026455
2758  003374  151322                          .WORD   151322
2759  003376  151322                          .WORD   151322
2760  003400  151322                          .WORD   151322
2761  003402  026455                          .WORD   026455
2762  003404  026455                          .WORD   026455
2763  003406  151322                          .WORD   151322
2764  003410  151322                          .WORD   151322
2765  003412  026455                          .WORD   026455
2766  003414  151322                          .WGRD   151322
2767  003416  026455                          .WORD   026455
```

N04


```
2768  003420  151322                  .WORD   151322
2769  003422  026455                  .WORD   026455
2770  003424  151322                  .WORD   151322
2771
2772  003426  165555                  .WORD   165555  ;STANDARD PATTERN 8
2773  003430  133333                  .WORD   133333
2774  003432  165555                  .WORD   165555
2775  003434  133333                  .WORD   133333
2776  003436  165555                  .WORD   165555
2777  003440  133333                  .WORD   133333
2778  003442  165555                  .WORD   165555
2779  003444  133333                  .WORD   133333
2780  003446  165555                  .WORD   165555
2781  003450  133333                  .WORD   133333
2782  003452  165555                  .WORD   165555
2783  003454  133333                  .WORD   133333
2784  003456  165555                  .WORD   165555
2785  003460  133333                  .WORD   133333
2786  003462  165555                  .WORD   165555
2787  003464  133333                  .WORD   133333
2788
2789  003466  000001                  .WORD   000001  ;STANDARD PATTERN 9
2790  003470  000002                  .WORD   000002
2791  003472  000004                  .WORD   000004
2792  003474  000010                  .WORD   000010
2793  003476  000020                  .WORD   000020
2794  003500  000040                  .WORD   000040
2795  003502  000100                  .WORD   000100
2796  003504  000200                  .WORD   000200
2797  003506  000400                  .WORD   000400
2798  003510  001000                  .WORD   001000
2799  003512  002000                  .WORD   002000
2800  003514  004000                  .WORD   004000
2801  003516  010000                  .WORD   010000
2802  003520  020000                  .WORD   020000
2803  003522  040000                  .WORD   040000
2804  003524  100000                  .WORD   100000
2805
2806  003526  177776                  .WORD   177776  ;STANDARD PATTERN 10
2807  003530  177775                  .WORD   177775
2808  003532  177773                  .WORD   177773
2809  003534  177767                  .WORD   177767
2810  003536  177757                  .WORD   177757
2811  003540  177737                  .WORD   177737
2812  003542  177677                  .WORD   177677
2813  003544  177577                  .WORD   177577
2814  003546  177377                  .WORD   177377
2815  003550  176777                  .WORD   176777
2816  003552  175777                  .WORD   175777
2817  003554  173777                  .WORD   173777
2818  003556  167777                  .WORD   167777
2819  003560  157777                  .WORD   157777
2820  003562  137777                  .WORD   137777
2821  003564  077777                  .WORD   077777
2822
2823  003566  172666                  .WORD   172666  ;STANDARD PATTERN 11
```

```
2824  003570  155555                          .WORD   155555
2825  003572  172666                          .WORD   172666
2826  003574  155555                          .WORD   155555
2827  003576  172666                          .WORD   172666
2828  003600  155555                          .WORD   155555
2829  003602  172666                          .WORD   172666
2830  003604  155555                          .WORD   155555
2831  003606  172666                          .WORD   172666
2832  003610  155555                          .WORD   155555
2833  003612  172666                          .WORD   172666
2834  003614  155555                          .WORD   155555
2835  003616  172666                          .WORD   172666
2836  003620  155555                          .WORD   155555
2837  003622  172666                          .WORD   172666
2838  003624  155555                          .WORD   155555
2839
2840  003626  077777                          .WORD   077777  ;STANDARD PATTERN 12
2841  003630  137777                          .WORD   137777
2842  003632  157777                          .WORD   157777
2843  003634  167777                          .WORD   167777
2844  003636  173777                          .WORD   173777
2845  003640  175777                          .WORD   175777
2846  003642  176777                          .WORD   176777
2847  003644  177377                          .WORD   177377
2848  003646  177577                          .WORD   177577
2849  003650  177677                          .WORD   177677
2850  003652  177737                          .WORD   177737
2851  003654  177757                          .WORD   177757
2852  003656  177767                          .WORD   177767
2853  003660  177773                          .WORD   177773
2854  003662  177775                          .WORD   177775
2855  003664  177776                          .WORD   177776
2856
2857  003666  153333                          .WORD   153333  ;STANDARD PATTERN 13
2858  003670  066667                          .WORD   066667
2859  003672  153333                          .WORD   153333
2860  003674  066667                          .WORD   066667
2861  003676  153333                          .WORD   153333
2862  003700  066667                          .WORD   066667
2863  003702  153333                          .WORD   153333
2864  003704  066667                          .WORD   066667
2865  003706  153333                          .WORD   153333
2866  003710  066667                          .WORD   066667
2867  003712  153333                          .WORD   153333
2868  003714  066667                          .WORD   066667
2869  003716  153333                          .WORD   153333
2870  003720  066667                          .WORD   066667
2871  003722  153333                          .WORD   153333
2872  003724  066667                          .WORD   066667
2873
2874  003726  000000                          .WORD   000000  ;STANDARD PATTERN 14
2875  003730  177777                          .WORD   177777
2876  003732  177777                          .WORD   177777
2877  003734  177777                          .WORD   177777
2878  003736  177777                          .WORD   177777
2879  003740  177777                          .WORD   177777
```

```
2880  003742  177777                    .WORD   177777
2881  003744  177777                    .WORD   177777
2882  003746  177777                    .WORD   177777
2883  003750  177777                    .WORD   177777
2884  003752  177777                    .WORD   177777
2885  003754  177777                    .WORD   177777
2886  003756  177777                    .WORD   177777
2887  003760  177777                    .WORD   177777
2888  003762  177777                    .WORD   177777
2889  003764  177777                    .WORD   177777
2890
2891  003766  177777                    .WORD   177777  ;STANDARD PATTERN 15
2892  003770  000000                    .WORD   000000
2893  003772  000000                    .WORD   000000
2894  003774  000000                    .WORD   000000
2895  003776  000000                    .WORD   000000
2896  004000  000000                    .WORD   000000
2897  004002  000000                    .WORD   000000
2898  004004  000000                    .WORD   000000
2899  004006  000000                    .WORD   000000
2900  004010  000000                    .WORD   000000
2901  004012  000000                    .WORD   000000
2902  004014  000000                    .WORD   000000
2903  004016  000000                    .WORD   000000
2904  004020  000000                    .WORD   000000
2905  004022  000000                    .WORD   000000
2906  004024  000000                    .WORD   000000
2907
```

```
2908                                    .SBTTL  ERROR POINTER TABLE
2909
2910                                    ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
2911                                    ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
2912                                    ;*LOCATION SITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
2913                                    ;*NOTE1:        IF SITEMB IS 0 THE ONLY PERTINENT DATA IS (SERRPC).
2914                                    ;*NOTE2:        EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
2915
2916                                    ;*      EM              ;;POINTS TO THE ERROR MESSAGE
2917                                    ;*      DH              ;;POINTS TO THE DATA HEADER
2918                                    ;*      DT              ;;POINTS TO THE DATA
2919                                    ;*      DF              ;;POINTS TO THE DATA FORMAT
2920
2921
2922  004026                            $ERRTB:
2923                                    ;ERROR 1
2924
2925                                    ;ERROR 1
2926
2927  004026  045060                            EM1                     ;RH11 INTERRUPT OCCURRED (RPAS = 0)
2928  004030  047520                            DH1
2929  004032  050152                            DT1
2930  004034  000000                            0
2931
2932                                    ;ERROR 2
2933
2934  004036  045123                            EM2                     ;UNEXPECTED ATTENTION OCCURRED
2935  004040  047525                            DH2
2936  004042  050156                            DT2
2937  004044  000000                            0
2938
2939                                    ;ERROR 3
2940
2941  004046  045161                            EM3                     ;MASSBUS PARITY ERROR (MCPE=1)
2942  004050  047602                            DH3
2943  004052  050174                            DT3
2944  004054  000000                            0
2945
2946                                    ;ERROR 4
2947
2948  004056  045217                            EM4                     ;MASSBUS PARITY ERROR (PAR=1)
2949  004060  047630                            DH4
2950  004062  050204                            DT4
2951  004064  000000                            0
2952
2953                                    ;ERROR 5
2954
2955  004066  045254                            EM5                     ;ADDRESS PLUG BIT CHANGED
2956  004070  047525                            DH2
2957  004072  050156                            DT2
2958  004074  000000                            0
2959
2960                                    ;ERROR 6
2961
2962  004076  045310                            EM6                     ;RH11 DIDN'T RESPOND TO ADDRESSING
2963  004100  047667                            DH6
```

```
2964  004102  050216                        DT6
2965  004104  000000                        0
2966
2967                                        ;;**********.**********.**********.**********.**********.**********.**********.**********
2968
2969                                        .SBTTL  SETUP AND INITIALIZATION ROUTINE
2970
2971                                        ;       START ADDRESS = 200
2972                                        ;       ADDRESS TO CHANGE RH11 UNIBUS ADDRESS = 204
2973
2974                                        ;;**********.**********.**********.**********.**********.**********.**********.**********
2975
2976  004106  012737  177777  001260  START:  MOV     #-1,CHGADR      ;SET RH11 ADDRESS CHANGE FLAG
2977  004114  000402                        BR      START2          ;START THE PROGRAM
2978  004116  005037  001260          START1: CLR     CHGADR          ;CLEAR THE RH11 ADDRESS CHANGE FLAG
2979  004122  000005                  START2: RESET                   ;CLEAR THE BUS
2980  004124  005227  177777                INC     #-1             ;FIRST TIME THROUGH?
2981  004130  001015                        BNE     REGPAS          ;BRANCH IF NOT
2982  004132  013737  001362  001372        MOV     QVCON,ENDCON            ;SET UP FOR QV PASS
2983  004140  013737  001364  001374        MOV     QVCON+2,ENDCON+2
2984  004146  013737  001366  001376        MOV     QVSEK,ENDSEK
2985  004154  013737  001370  001400        MOV     QVSEK+2,ENDSEK+2
2986  004162  000414                        BR      SETUP
2987  004164  013737  001352  001372  REGPAS: MOV     ENDCN,ENDCON            ;SET UP FOR NORMAL PASS
2988  004172  013737  001354  001374        MOV     ENDCN+2,ENDCON+2
2989  004200  013737  001356  001376        MOV     ENDSK,ENDSEK
2990  004206  013737  001360  001400        MOV     ENDSK+2,ENDSEK+2
2991  004214                          SETUP:
2992                                        .SBTTL  INITIALIZE THE COMMON TAGS
2993                                        ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
2994  004214  012706  001100                MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
2995  004220  005C26                        CLR     (R6)+           ;;CLEAR MEMORY LOCATION
2996  004222  022706  001140                CMP     #SWR,R6 ;;DONE?
2997  004226  001374                        BNE     .-6             ;;LOOP BACK IF NO
2998  004230  012706  001100                MOV     #STACK,SP       ;;SETUP THE STACK POINTER
2999                                        ;;INITIALIZE A FEW VECTORS
3000  004234  012737  030570  000030        MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR E    ROUTINE
3001  004242  012737  000340  000032        MOV     #340,@#EMTVEC+2 ;;LEVEL 7
3002  004250  012737  033136  000034        MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
3003  004256  012737  000340  000036        MOV     #340,@#TRAPVEC+2;LEVEL 7
3004  004264  012737  176543  032522        MOV     #176543,$HINUM  ;;PRIME THE RANDOM NUMBER GENERATOR
3005  004272  012737  123456  032524        MOV     #123456,$LONUM  ;;BOTH HIGH AND LOW WORDS
3006                                        ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
3007                                        ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
3008  004300  013746  000004                MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
3009  004304  012737  004340  000004        MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
3010  004312  012737  177570  001140        MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
3011  004320  012737  177570  001142        MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
3012  004326  022777  177777  174604        CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
3013  004334  001012                        BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
3014                                                                ;;AND  THE HARDWARE SWR IS NOT = -1
3015  004336  000403                        BR      65$             ;;BRANCH IF NO TIMEOUT
3016  004340  012716  004346          64$:  MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
3017  004344  000002                        RTI
3018  004346  012737  000176  001140  65$:  MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
3019  004354  012737  000174  001142        MOV     #DISPREG,DISPLAY
```

```
3020  004362  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
3021
3022  004366  012737  000240  000032          MOV     #240,@#EMTVEC+2   ;CHANGE EMT PRIORITY TO 5
3023  004374  012737  000240  000034          MOV     #240,@#TRAPVEC+2  ;CHANGE TRAP PRIORITY TO 5
3024  004402  005227  177777                  INC     #-1             ;FIRST START ?
3025  004406  001017                          BNE     1$              ;BR IF NOT
3026  004410  023737  000042  000046          CMP     @#42,@#46       ;ARE WE IN ACT-11 AUTO MODE?
3027  004416  001413                          BEQ     1$              ;BRANCH IF YES
3028  004420  104401  055376                  TYPE    ,TITLE          ;NAME AND MAINDEC NUMBER
3029  004424  005737  000042                  TST     42              ;AUTO ACCEPT OR CHAIN MODE ?
3030  004430  001006                          BNE     1$              ;BR IF EITHER
3031  004432  122737  000011  000041          CMPB    #11,41          ;LOADED FROM AN RP04/5/6 ?
3032  004440  001002                          BNE     1$              ;BR IF NOT
3033  004442  104401  055476                  TYPE    ,LOADRV         ;INSTRUCT THE OPERATOR ON HOW TO TEST DRIVE 0
3034  004446  004737  030110          1$:     JSR     PC,$TKINT       ;TURN ON THE KEYBOARD INTERRUPT
3035                                  .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
3036  004452  005737  000042                  TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
3037  004456  001006                          BNE     67$             ;;BRANCH IF YES
3038  004460  023727  001140  000176          CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
3039  004466  001005                          BNE     68$             ;;BRANCH IF NO
3040  004470  104406                          GTSWR                   ;;GET SOFT-SWR SETTINGS
3041  004472  000403                          BR      68$
3042  004474  112737  000001  001134  67$:    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
3043  004502                          68$:
3044  004502  005227  177777                  INC     #-1             ;FIRST START ?
3045  004506  001015                          BNE     2$              ;BR IF NOT
3046  004510  004737  055004                  JSR     PC,BUSADR       ;CHECK RH11 BUS ADDRESS
3047  004514  013737  001170  033372          MOV     $RPADR,RPADR    ;RH11 ADDRESS
3048  004522  013737  001172  033374          MOV     $RPVEC,RPVEC    ;RH11 VECTOR ADDRESS
3049  004530  005737  000042                  TST     @#42            ;ACT-11 AUTO OR CHAIN MODE?
3050  004534  001002                          BNE     2$              ;BRANCH IF EITHER, SKIP
3051                                                                  ;DATE & OPERATOR ID INPUT
3052  004536  004737  054526                  JSR     PC,OPRDAT       ;GET THE DATE AND OPERATOR ID
3053  004542  005037  001214          2$:     CLR     STATIN          ;CLEAR PERFORMANCE SUMMARY TYPEOUT FLAG
3054  004546  012705  001440                  MOV     #ORDERQ,R5      ;START OF AREA TO CLEAR
3055  004552  005025                  3$:     CLR     (R5)+
3056  004554  022705  001740                  CMP     #BLKADR,R5      ;LOOK FOR END OF CLEAR AREA
3057  004560  001374                          BNE     3$              ;BR IF NOT FINISHED
3058  004562  012706  001100                  MOV     #STACK,SP       ;SETUP THE STACK POINTER
3059  004566  005037  177776                  CLR     PS              ;CLEAR THE PROCESSOR STATUS WORD
3060  004572  013737  001212  001274          MOV     HZ,SIXTEE       ;1/60 TH OR 1/50 TH SECOND COUNTER VALUE
3061  004600  005037  001266                  CLR     HOUR            ;CLEAR THE HOUR'S COUNTER
3062  004604  005037  001270                  CLR     MINUTE          ;CLEAR THE MINUTE'S COUNTER
3063  004610  005037  001272                  CLR     SECOND          ;CLEAR THE SECOND'S COUNTER
3064  004614  005037  001412                  CLR     INTRVL+2        ;CLEAR INTERVAL COUNTER
3065  004620  005037  001216                  CLR     PACK            ;CLEAR THE 'R' OR 'W' COMMAND FLAG
3066  004624  005037  001262                  CLR     CFLAG           ;CLEAR THE 'CONTROL C' FLAG
3067  004630  042737  170000  001406          BIC     #170000,MAXER   ;MAKE SURE ERROR LIMITS ARE NOT TOO HIGH
3068
3069                          ;ROUTINE TO DETERMINE BUFFER AREA SIZE
3070
3071  004636  005227  177777  SIZMEM: INC     #-1             ;SEE IF TIME TO SIZE MEMORY
3072  004642  001005                  BNE     1$              ;BR IF NOT
3073  004644  004737  054706                  JSR     PC,$SIZE        ;SEE HOW MUCH MEMORY ON SYSTEM
3074  004650  013737  055002  001256          MOV     $LSTAD,LSTAD    ;SAVE THE LAST ADDRESS
3075  004656  012737  000001  001616  1$:     MOV     #1,BUFTBL       ;LOAD NUMBER OF BUFFERS
```

```
                                ;GET VALUE FOR SOFTWARE SWITCH REGISTER

3076  004664  012737  054526  001620          MOV     #ENDPGM,BUFTBL+2  ;STARTING ADDRESS OF BUFFER
3077  004672  013737  001256  001622          MOV     LSTAD,BUFTBL+4  ;LAST ADDR TO BUFFER ALLOCATION TABLE
3078  004700  162737  054526  001622          SUB     #ENDPGM,BUFTBL+4  ;SUBTRACT PROGRAM SPACE
3079  004706  000241                          CLC                     ;CLEAR THE 'C' BIT
3080  004710  006037  001622                  ROR     BUFTBL+4        ;CONVERT TO WORD COUNT
3081  004714  162737  000144  001622          SUB     #100.,BUFTBL+4  ;SAVE ROOM FOR THE 'ABS' LOADER
3082  004722  023727  001256  100000          CMP     LSTAD,#100000   ;16K ON THE SYSTEM ?
3083  004730  103406                          BLO     3$              ;BR IF YES
3084  004732  105737  000041                  TSTB    41              ;SEE WHO LOADED THE PROGRAM
3085  004736  001403                          BEQ     3$              ;BR IF LOADED BY PAPER TAPE
3086  004740  162737  002570  001622          SUB     #1400.,BUFTBL+4  ;SUBTRACT 'XXDP' LOADER SIZE
3087  004746  005737  001404          3$:     TST     MAXDL           ;VALUE IN 'MAXDL' ?
3088  004752  001012                          BNE     4$              ;BR IF VALUE IS
3089  004754  012737  013534  001404          MOV     #5980.,MAXDL    ;ASSUME FULL TRACK + 1 SEC MAXIMUM
3090  004762  023737  001404  001622          CMP     MAXDL,BUFTBL+4  ;IS THAT TOO LARGE ?
3091  004770  103403                          BLO     4$              ;BR IF NOT
3092  004772  013737  001622  001404          MOV     BUFTBL+4,MAXDL  ;USE MAX AVAIL MEMORY AS MAX BUFFER SIZE
3093  005000  013737  001622  053516  4$:     MOV     BUFTBL+4,PARLST+2  ;VALUE FOR THE PARAMETER TABLE
3094
3095                                  ;SEE IF THE OPERATOR WANTS TO CHANGE ANY PARAMETERS
3096
3097  005006  005737  000042          LKPAR:  TST     @#42            ;'XXDP' CHAIN MODE OR 'ACT11' OPERATION ?
3098  005012  001022                          BNE     SETVEC          ;BR IF YES
3099  005014  104401  053612                  TYPE    ,ASKPAR         ;ASK FOR PARMETERS
3100  005020  104411                          RDLIN                   ;READ THE ENTRY
3101  005022  012605                          MOV     (SP)+,R5        ;ADDRESS OF ENTRY TO R5
3102  005024  122715  000131                  CMPB    #'Y,(R5)        ;WAS ENTRY A 'Y' (YES)
3103  005030  001013                          BNE     SETVEC          ;BR IF NOT 'Y'
3104
3105  005032  012703  053514          ENTPR:  MOV     #PARLST,R3      ;PARAMETER TABLE ADDRESS
3106  005036  004737  026332                  JSR     PC,PARENT       ;GET THE PARAMETER ENTRY
3107  005042  023727  001404  000004          CMP     MAXDL,#4        ;IS THE 'MAXDL' VALUE OK ?
3108  005050  103003                          BHIS    SETVEC          ;BR IF IT IS
3109  005052  012737  000004  001404          MOV     #4,MAXDL        ;SET 'MAXDL' TO THE MINIMUM VALUE
3110
3111                                  ;DISPLAY DRIVE STATUS AND SET UP THE OTHER SYSTEM DEVICES THAT
3112                                  ;       THE PROGRAM WILL USE
3113
3114  005060  004737  022452          SETVEC: JSR     PC,CKCLK        ;START THE CLOCK
3115  005064  004737  033410                  JSR     PC,RPINIT       ;INITIALIZE THE RP04/5/6 DRIVER
3116  005070  012737  177777  033332          MOV     #-1,SAVEFG      ;SET THE SAVE REGISTERS FLAG
3117  005076  062727  177777  000000          ADD     #-1,#0          ;CHECK FOR FIRST START
3118  005104  103004                          BCC     11$             ;BR IF FIRST START
3119  005106  032777  000004  174024          BIT     #SW02,@SWR      ;TYPEOUT THE DRIVE STATUS TABLE ?
3120  005114  001076                          BNE     10$             ;BR IF NOT
3121  005116  012737  000340  177776  11$:    MOV     #PR7,PS         ;SET PRIORITY TO 7
3122  005124  005004                          CLR     R4              ;DRIVE TABLE POINTER
3123  005126  104401  001165                  TYPE    ,$CRLF          ;CR-LF
3124  005132  104401  052404                  TYPE    ,SYSTAT         ;TYPE STATUS HEADING
3125  005136                          1$:
3126  005136  010446                          MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
3127                                                                  ;;TYPE DRIVE NUMBER
3128  005140  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
3129  005142     002                          .BYTE   2               ;;TYPE 2 DIGIT(S)
3130  005143     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
3131  005144  104401  052213                  TYPE    ,LIN4SP         ;SPACES
```

```
3132  005150  105764  033244                TSTB    DRVSTA(R4)          ;CHECK DRIVE'S STATUS
3133  005154  100416                      BMI     4$                  ;BR IF UNSAFE
3134  005156  001020                      BNE     5$                  ;BR IF ONLINE
3135  005160  105764  033254                TSTB    DRVTYP(R4)          ;SEE IF OFFLINE OR NONEXISTENT
3136  005164  001404                      BEQ     2$                  ;BR IF NONEXISTENT
3137  005166  100006                      BPL     3$                  ;BR IF OFFLINE
3138  005170  104401  052317                TYPE    ,NOTRP              ;DRIVE NOT AN RP04/5/6
3139  005174  000440                      BR      9$                  ;CHECK NEXT DRIVE
3140  005176  104401  052340          2$:   TYPE    ,NOTPRS             ;DRIVE NOT PRESENT
3141  005202  000435                      BR      9$                  ;CHECK NEXT DRIVE
3142  005204  104401  052226          3$:   TYPE    ,UNTOFF             ;DRIVE OFFLINE
3143  005210  000405                      BR      6$                  ;PRINT DRIVE TYPE
3144  005212  104401  052374          4$:   TYPE    ,NOTSAF             ;DRIVE UNSAFE
3145  005216  000402                      BR      6$                  ;PRINT DRIVE TYPE
3146  005220  104401  052237          5$:   TYPE    ,UNTON              ;DRIVE ONLINE
3147  005224  104401  052215          6$:   TYPE    ,LINSP              ;SPACES
3148  005230  012737  052424  005274        MOV     #RP04B,8$           ;ADDRESS OF RP04 MESSAGE
3149  005236  132764  000001  033254        BITB    #BIT00,DRVTYP(R4)   ;RP04 ?
3150  005244  001012                      BNE     7$                  ;BR IF YES
3151  005246  012737  052431  005274        MOV     #RP05,8$            ;ADDRESS OF RP05 MESSAGE
3152  005254  132764  000002  033254        BITB    #BIT01,DRVTYP(R4)   ;RP05 ?
3153  005262  001003                      BNE     7$                  ;BR IF YES
3154  005264  012737  052436  005274        MOV     #RP06,8$            ;ADDRESS OF RP06 MESSAGE
3155  005272  104401                7$:   TYPE                        ;TYPE THE DRIVE TYPE MESSAGE
3156  005274  000000                8$:   .WORD   0                   ;MESSAGE ADDRESS HERE
3157  005276  104401  001165          9$:   TYPE    ,$CRLF              ;CR-LF
3158  005302  005204                      INC     R4                  ;INCREMENT DRIVE NUMBER/TABLE POINTER
3159  005304  020427  000010                CMP     R4,#8.              ;FINISHED ?
3160  005310  001312                      BNE     1$                  ;BR IF NOT
3161  005312  104401  001165          10$:  TYPE    ,$CRLF              ;CR-LF
3162  005316  005037  177776                CLR     @PS                 ;SET PRIORITY BACK TO '0'
3163  005322  000137  005326                JMP     MONTR               ;CHECK FOR 'XXDP' OR 'ACT11' MONITOR
3164
3165                                  ;SETUP IF 'XXDP' OR 'ACT11' OPERATION
3166
3167  005326  005737  000042          MONTR:  TST     @42                 ;'XXDP' CHAIN MODE OR 'ACT11' ALTO ACCEPT
3168  005332  001402                      BEQ     1$                  ;BR IF NEITHER
3169  005334  004737  024334                JSR     PC,ASGN2            ;ASSIGN DRIVES
3170  005340  005227  177777          1$:   INC     #-1                 ;FIRST START ?
3171  005344  001011                      BNE     2$                  ;BR IF NOT
3172  005346  105737  000041                TSTB    @#41                ;LOADED FROM PAPER TAPE ?
3173  005352  001406                      BEQ     2$                  ;BR IF YES
3174  005354  023727  001256  100000        CMP     LSTAD,#100000       ;MORE THAN 16K ON THE SYSTEM ?
3175  005362  103002                      BHIS    2$                  ;BR IF YES
3176  005364  104401  055675                TYPE    ,NOLOAD             ;TELL THE OPERATOR THAT THE 'XXDP' LOADER
3177                                                                    ;WILL BE OVERWRITTEN
3178  005370  004737  030110          2$:   JSR     PC,$TKINT           ;INITIALIZE THE KEYBOARD INTERRUPT HANDLER
3179  005374  104401  053401                TYPE    ,INTDON             ;TYPE 'INITIALIZE COMPLETE'
3180  005400  000137  005562                JMP     @#MAIN1             ;START THE PROGRAM
3181
3182                                  ;'XXDP' OR 'ACT11' END OF TEST ROUTINE
3183
3184  005404  013700  000042          $GET42: MOV     @42,R0              ;MONITOR ADDRESS
3185  005410  001002                      BNE     1$                  ;BR IF MONITOR
3186  005412  000137  005562                JMP     @#MAIN1             ;NONE, CONTINUE
3187  005416  000005                1$:   RESET                       ;CLEAR EVERYTHING
```

```
3188  005420  004710                SENDAD: JSR     PC,(R0)         ;GO TO THE MONITOR
3189  005422  000240                        NOP                     ;SAVE ROOM
3190  005424  000240                        NOP                     ;FOR
3191  005426  000240                        NOP                     ;ACT11
3192  005430  000137  004116        $DOAGN: JMP     START1          ;START AGAIN
3193
3194                                ;;*****************************************************************
3195
3196                                .SBTTL  MAIN PROGRAM
3197
3198                                ;;*****************************************************************
3199
3200  005434  012703  000010        MAIN:   MOV     #8.,R3          ;DRIVE COUNTER
3201  005440  012705  001464                MOV     #DUNIT,R5       ;ADDRESS OF 'DROP DRIVE' TABLE
3202  005444  005715                1$:     TST     (R5)            ;SEE IF ENTRY AT PRESENT POSITION
3203  005446  001011                        BNE     3$              ;BR IF THERE IS ONE
3204  005450  062705  000002        2$:     ADD     #2,R5           ;INCREMENT TO NEXT TABLE POSITION
3205  005454  005303                        DEC     R3              ;DECREMENT DRIVE COUNTER
3206  005456  001372                        BNE     1$              ;BR IF MORE TO CHECK
3207  005460  005737  001462                TST     ASNLST          ;ANY DRIVES ACTIVE ?
3208  005464  001036                        BNE     MAIN1           ;BR IF YES
3209  005466  000137  005404                JMP     $GET42          ;CHECK FOR MONITOR RETURN
3210  005472  012701  001530        3$:     MOV     #AVAIL,R1       ;ADDRESS OF 'AVAILABLE DRIVES' TABLE
3211  005476  005711                4$:     TST     (R1)            ;SEE IF AT END OF TABLE
3212  005500  001405                        BEQ     5$              ;BR IF AT END; GO CHECK 'WAIT' TABLE
3213  005502  021115                        CMP     (R1),(R5)       ;IS DRIVE IN 'AVAIL' THE ONE TO BE DROPPED
3214  005504  001414                        BEQ     7$              ;BR IF YES
3215  005506  062701  000002                ADD     #2,R1           ;INCREMENT 'AVAIL' TABLE ADDRESS
3216  005512  000771                        BR      4$              ;CONTINUE LOOKING
3217  005514  012701  001552        5$:     MOV     #WAIT,R1        ;MOVE THE ADDRESS OF THE BUFFER WAIT TABLE
3218  005520  005711                6$:     TST     (R1)            ;AT THE END OF THE 'WAIT' TABLE ?
3219  005522  001752                        BEQ     2$              ;BR IF YES: SEE IF ANY MORE 'DROP' REQUESTS
3220  005524  021115                        CMP     (R1),(R5)       ;DRIVE IN THE 'WAIT' TABLE ?
3221  005526  001403                        BEQ     7$              ;BR IF IT IS
3222  005530  062701  000002                ADD     #2,R1           ;INCREMENT 'WAIT' TABLE ADDRESS
3223  005534  000771                        BR      6$              ;CONTINUE LOOK THROUGH THE 'WAIT' TABLE
3224  005536  011100                7$:     MOV     (R1),R0         ;PUT THE DRIVE'S BLOCK ADDRESS IN R0
3225  005540  104401  052710                TYPE    ,DEASSG         ;TYPE 'DRIVE DEASSIGNED'
3226  005544  004737  022736                JSR     PC,TYPES        ;TYPE THE DRIVE'S PERFORMANCE SUMMARY
3227  005550  005015                        CLR     (R5)            ;CLEAR THE 'DROP DRIVE' TABLE ENTRY
3228  005552  005011                        CLR     (R1)            ;REMOVE THE DRIVE FROM THE 'AVAIL' OR 'WAIT' TABLE
3229  005554  004737  017570                JSR     PC,CMPRES       ;COMPRESS THE RESPECTIVE TABLE
3230  005560  000733                        BR      2$              ;SEE IF ANY MORE DRIVES
3231
3232                                ;LOOK FOR DRIVES TO BE ASSIGNED
3233
3234  005562  012703  000010        MAIN1:  MOV     #8.,R3          ;DRIVE COUNT
3235  005566  005002                        CLR     R2              ;'AVAIL' INDEX
3236  005570  005004                        CLR     R4              ;ASSIGN LIST INDEX
3237  005572  005005                        CLR     R5              ;NEW DRIVE INDEX
3238  005574  005765  001506        1$:     TST     NEWUNT(R5)      ;NEW DRIVE IN THIS POSITION
3239  005600  001006                        BNE     3$              ;BR IF THERE IS
3240  005602  062705  000002        2$:     ADD     #2,R5           ;INCREMENT R5
3241  005606  005204                        INC     R4              ;INCREMENT ASSIGN INDEX
3242  005610  005303                        DEC     R3              ;DECREMENT DRIVE COUNT
3243  005612  001370                        BNE     1$              ;BR IF MORE DRIVES
```

```
3244  005614  000432                          BR      MAIN2           ;START OPERATIONS FOR THE AVAILABLE DRIVES
3245  005616  104401  052220          3$:     TYPE    ,UNTMSG         ;'DRIVE'
3246  005622  010446                          MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
3247                                                                  ;;TYPE DRIVE NUMBER
3248  005624  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
3249  005626     002                          .BYTE   2               ;;TYPE 2 DIGIT(S)
3250  005627     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
3251  005630  104401  052760                  TYPE    ,ASGND          ;'ASSIGNED'
3252  005634  005762  001530          4$:     TST     AVAIL(R2)       ;AT END OF AVAILABLE TABLE
3253  005640  001403                          BEQ     5$              ;BR IF YES
3254  005642  062702  000002                  ADD     #2,R2           ;INCREMENT AVAILABLE TABLE INDEX
3255  005646  000772                          BR      4$              ;CONTINUE LOOKING FOR END OF TABLE
3256  005650  016562  001506  001530  5$:     MOV     NEWUNT(R5),AVAIL(R2)  ;MOVE ADDR OF DRIVE INTO AVAIL LST
3257  005656  005065  001506                  CLR     NEWUNT(R5)      ;TAKE DRIVE OUT OF NEW DRIVE TABLE
3258  005662  156437  033360  001462          BISB    ATABIT(R4),ASNLST ;SET DRIVE ASSIGNED INDICATOR
3259  005670  016200  001530                  MOV     AVAIL(R2),R0    ;PUT STARTING ADDRESS OF BLOCK IN R0
3260  005674  062702  000002                  ADD     #2,R2           ;INCREMENT AVAILABLE TABLE POINTER
3261  005700  000740                          BR      2$              ;LOOK FOR MORE DRIVES
3262
3263                                  ;GET PARAMETERS, BUFFER SPACE, AND START ORDERS FOR DRIVES IN
3264                                  ;       THE 'AVAILABLE' QUEUE
3265
3266  005702  005737  001552          MAIN2:  TST     WAIT            ;OUTSTANDING BUFFER REQUESTS
3267  005706  001113                          BNE     MAIN3           ;BR IF THERE ARE
3268  005710  005002                          CLR     R2              ;CLEAR DRIVE TABLE POINTER
3269  005712  005762  001530          1$:     TST     AVAIL(R2)       ;ANY DRIVES WAITING FOR PARAMETERS
3270  005716  001551                          BEQ     IDLE            ;BRANCH IF NONE
3271  005720  016200  001530                  MOV     AVAIL(R2),R0    ;CONTROL BLOCK ADDR IN R0
3272  005724  005760  000104                  TST     $NEXT(R0)       ;PARAMETERS BEEN SELECTED ?
3273  005730  001021                          BNE     6$              ;BR IF THEY HAVE
3274  005732  105760  000026                  TSTB    $PACK(R0)       ;'R' OR 'W' COMMAND FOR THE DRIVE ?
3275  005736  001403                          BEQ     2$              ;BR IF NOT
3276  005740  004737  017606                  JSR     PC,WRTPK        ;GET DATA PACK PARAMETERS
3277  005744  000415                          BR      7$              ;GET THE BUFFER
3278  005746  012701  001574          2$:     MOV     #PARQ,R1        ;ADDRESS OF THE PARAMETER QUEUE
3279  005752  020011          3$:     CMP     R0,(R1)         ;IS CURRENT DRIVE IN THE QUEUE ?
3280  005754  001403                          BEQ     4$              ;BR IF IT IS
3281  005756  005721                          TST     (R1)+           ;AT END OF THE QUEUE
3282  005760  001403                          BEQ     5$              ;BR IF AT END
3283  005762  000773                          BR      3$              ;CONTINUE LOOKING
3284  005764  004737  017570          4$:     JSR     PC,CMPRES       ;COMPRESS THE TABLE
3285  005770  004737  016506          5$:     JSR     PC,SELPAR       ;SELECT THE PARAMETERS
3286  005774  004737  017362          6$:     JSR     PC,GETPAR       ;LOAD NEW PARAMETERS
3287  006000  005046                  7$:     CLR     -(SP)           ;MAKE ROOM ON THE STACK FOR THE BUFFER ADDR
3288  006002  004737  015732                  JSR     PC,GETBUF       ;GET BUFFER
3289  006006  012660  000006                  MOV     (SP)+,$BUF(R0)  ;MOVE BUFFER ADDR TO DPB
3290  006012  001424                          BEQ     8$              ;BR IF '0' ADDR (NO BUFFER)
3291  006014  004737  016302                  JSR     PC,FILBUF       ;FILL THE BUFFER
3292  006020  005060  000072                  CLR     $FAIR(R0)       ;CLEAR THE 'FAIRNESS' COUNT
3293  006024  004737  016430                  JSR     PC,GODRIV       ;PUT CURRENT DPB IN DRIVER
3294  006030  012705  001440                  MOV     #ORDERQ,R5      ;ADDRESS OF ORDER QUEUE IN R5
3295  006034  005725                          TST     (R5)+           ;END OF QUEUE ?
3296  006036  001376                          BNE     .-2             ;BR IF NOT
3297  006040  010045                          MOV     R0,-(R5)        ;PUT BLOCK ADDRESS INTO QUEUE
3298  006042  105760  000026                  TSTB    $PACK(R0)       ;'R' OR 'W' COMMAND FOR DRIVE ?
3299  006046  001025                          BNE     10$             ;BR IF EITHER
```

```
3300  006050  012705  001574            MOV     #PARQ,R5        ;PUT BLOCK INTO THE PARAMETER QUEUE
3301  006054  005725                    TSI     (R5)+           ;FIND THE END OF THE QUEUE
3302  006056  001376                    BNE     .-2             ;BR IF NOT AT END OF QUEUE
3303  006060  010045                    MOV     R0,-(R5)        ;PUT BLOCK ADDRESS INTO THE QUEUE
3304  006062  000417                    BR      10$             ;CONTINUE LOOKING
3305  006064  026037  000072  001254 8$:  CMP   $FAIR(R0),FAIRNS  ;ENTRY BEEN IN THE QUEUE LONG ENOUGH ?
3306  006072  001405                    BEQ     9$              ;BR IF YES
3307  006074  005260  000072            INC     $FAIR(R0)       ;INCREMENT THE ENTRY COUNT
3308  006100  062702  000002            ADD     #2,R2           ;INCREMENT THE POINTER
3309  006104  000702                    BR      1$              ;LOOK FOR SOME MORE DRIVES
3310  006106  012705  001552        9$: MOV     #WAIT,R5        ;'WAIT' QUEUE ADDRESS
3311  006112  005725                    TST     (R5)+           ;LOOK FOR AN OPENING
3312  006114  001376                    BNE     .-2             ;BR IF NONE YET
3313  006116  016245  001530            MOV     AVAIL(R2),-(R5)  ;MOVE DRIVE'S BLOCK ADDRESS TO QUEUE
3314  006122  012701  001530        10$: MOV    #AVAIL,R1       ;'AVAILABLE' TABLE ADDRESS
3315  006126  060201                    ADD     R2,R1           ;FORM ADDRESS OF LAST ENTRY
3316  006130  004737  017570            JSR     PC,CMPRES       ;COMPRESS THE TABLE
3317  006134  000666                    BR      1$              ;CONTINUE LOOKING
3318
3319                                ;GET BUFFER ASSIGNMENTS FOR DRIVES IN THE 'BUFFER WAIT' QUEUE
3320
3321  006136  013700  001552        MAIN3: MOV  WAIT,R0         ;MOVE THE 'WAIT' ENTRY TO R0
3322  006142  005046                    CLR     -(SP)           ;MAKE ROOM ON THE STACK FOR THE BUFFER ADDR
3323  006144  004737  015732            JSR     PC,GETBUF       ;TRY TO GET A BUFFER
3324  006150  012660  000006            MOV     (SP)+,$BUF(R0)  ;MOVE THE BUFFER ADDR TO THE DPB
3325  006154  001002                    BNE     1$              ;BR IF A BUFFER WAS ASSIGNED
3326  006156  000137  006242            JMP     IDLE            ;NO BUFFER AVAILABLE YET
3327  006162  004737  016302        1$: JSR     PC,FILBUF       ;FILL THE BUFFER
3328  006166  004737  016430            JSR     PC,GODRIV       ;PUT THE ENTRY IN THE DRIVER
3329  006172  005060  000072            CLR     $FAIR(R0)       ;CLEAR THE 'FAIRNESS' COUNT
3330  006176  012705  001440            MOV     #ORDERQ,R5      ;ADDRESS OF ORDER QUEUE IN R5
3331  006202  005725                    TST     (R5)+           ;AT END OF THE QUEUE
3332  006204  001376                    BNE     .-2             ;BR IF NOT
3333  006206  010045                    MOV     R0,-(R5)        ;PUT BLOCK ADDRESS IN QUEUE
3334  006210  105760  000026            TSTB    $PACK(R0)       ;'R' OR 'W' COMMAND FOR DRIVE ?
3335  006214  001005                    BNE     2$              ;BR IF YES, DON'T PUT BLOCK INTO 'PARQ'
3336  006216  012705  001574            MOV     #PARQ,R5        ;FIND THE END OF THE PARAMETER QUEUE
3337  006222  005725                    TST     (R5)+           ;OPEN SLOT IN THE QUEUE ?
3338  006224  001376                    BNE     .-2             ;BR IF NOT
3339  006226  010045                    MOV     R0,-(R5)        ;PUT BLOCK ADDRESS INTO THE QUEUE
3340  006230  012701  001552        2$: MOV     #WAIT,R1        ;ADDRESS OF TABLE TO TO COMPRESS
3341  006234  004737  017570            JSR     PC,CMPRES       ;COMPRESS THE WAIT TABLE
3342  006240  000620                    BR      MAIN2           ;LOOK FOR MORE ENTRIES
3343
3344                                ;WAIT FOR AN ORDER TO FINISH
3345
3346  006242  012701  001440        IDLE: MOV   #ORDERQ,R1      ;ADDRESS OF THE ORDER QUEUE IN R1
3347  006246  012100                1$: MOV     (R1)+,R0        ;PUT BLOCK ADDRESS INTO R0
3348  006250  001433                    BEQ     IDLE1           ;BR IF END OF QUEUE
3349  006252  005760  000016            TST     $STATUS(R0)     ;SEE IF DRIVE FINISHED
3350  006256  001773                    BEQ     1$              ;BR IF DRIVE NOT FINISHED
3351  006260  162701  000002            SUB     #2,R1           ;CORRECT THE QUEUE POINTER
3352  006264  010146                    MOV     R1,-(SP)        ;SAVE THE QUEUE ADDRESS
3353  006266  004737  015566            JSR     PC,STATIS       ;ACCUMULATE STATISTICS FOR DRIVE IN R0
3354  006272  000240                    NOP                     ;DEBUGGING AID
3355  006274  004737  006600            JSR     PC,PR' 'ES      ;PROCESS END OF ORDER
```

```
3356  006300  005037  001264                  CLR     BADSEC          ;CLEAR THE BAD TRK/SEC ERROR INDICATOR
3357  006304  004737  026564                  JSR     PC,ABNRML       ;SEE IF ANY DRIVES HAVE TOO MANY ERRORS
3358  006310  004737  026612                  JSR     PC,EOP          ;SEE IF ANY DRIVE HAS XFERED 3X1019 BITS
3359  006314  012601                          MOV     (SP)+,R1        ;RESTORE THE ORDER TABLE INDEX
3360  006316  012705  001530                  MOV     #AVAIL,R5       ;FIND THE END OF THE 'AVAILABLE' TABLE
3361  006322  005725                  2$:     TST     (R5)+           ;END OF THE TABLE ?
3362  006324  001376                          BNE     2$              ;BR IF NOT AT END OF LIST
3363  006326  011145                          MOV     (R1),-(R5)      ;MOVE THE BLOCK ADDRESS INTO THE TABLE
3364  006330  004737  017570                  JSR     PC,CMPRES       ;COMPRESS THE ORDER QUEUE
3365  006334  004737  016066                  JSR     PC,RELBUF       ;RESTORE BUFFER
3366  006340  005737  001262          IDLE1:  TST     CFLAG           ;'CONTROL C' FLAG ENTERED ?
3367  006344  001403                          BEQ     1$              ;BR IF IT WAS
3368  006346  004737  023750                  JSR     PC,KSR          ;SERVICE THE KEYBOARD
3369  006352  000733                          BR      IDLE            ;SYSTEM WAS BUSY
3370  006354  032777  000004  172556  1$:     BIT     #SW02,@SWR      ;TYPE PERFORMANCE SUMMARY
3371  006362  001007                          BNE     2$              ;BR IF NOT
3372  006364  005737  001214                  TST     STATIN          ;TIME TO TYPE THE PERFORMANCE SUMMARY ?
3373  006370  001404                          BEQ     2$              ;BR IF NOT
3374  006372  005037  001214                  CLR     STATIN          ;CLEAR THE INDICATOR
3375  006376  004737  022654                  JSR     PC,STATPR       ;TYPE THE SUMMARY
3376  006402  005737  001574          2$:     TST     PARQ            ;ENTRY IN THE PARAMETER QUEUE ?
3377  006406  001410                          BEQ     3$              ;BR IF NOT
3378  006410  013700  001574                  MOV     PARQ,R0         ;PUT THE BLOCK ADDRESS INTO R0
3379  006414  004737  016506                  JSR     PC,SELPAR       ;GET THE PARAMETERS FOR NEXT OPERATION
3380  006420  012701  001574                  MOV     #PARQ,R1        ;SETUP TO COMPRESS THE TABLE
3381  006424  004737  017570                  JSR     PC,CMPRES       ;COMPRESS THE PARAMETER QUEUE
3382  006430  000137  005434          3$:     JMP     MAIN            ;CONTINUE THE LOOP
3383
3384                                          ;SETUP TO REFORMAT AN ERROR SECTOR
3385
3386  006434  032777  000001  172476  REFMT:  BIT     #SW0,@SWR       ;READ ONLY SWITCH SET ?
3387  006442  001055                          BNE     REFMTX          ;BR IF IT IS
3388  006444  032777  000200  172466          BIT     #SW7,@SWR       ;SWITCH 7 SET ?
3389  006452  001051                          BNE     REFMTX          ;BR IF IT IS
3390  006454  005737  001416                  TST     FORMAT          ;WRITE HEADER & DATA ORDERS ALLOWED ?
3391  006460  001446                          BEQ     REFMTX          ;BR IF NOT
3392  006462  016060  000272  000100          MOV     $RPCC(R0),$NCYL(R0)  ;USE PRESENT CYLINDER
3393  006470  004737  022354                  JSR     PC,READOR       ;GET CORRECTED SECTOR-TRACK ADDRESSES
3394  006474  112660  000077                  MOVB    (SP)+,$NTRK(R0) ;TRACK ADDR TO DPB
3395  006500  112660  000076                  MOVB    (SP)+,$NSEC(R0) ;SECTOR ADDR TO DPB
3396  006504  012760  000404  000102          MOV     #260.,$NWROL(R0)  ;WORD COUNT FOR FORMAT
3397  006512  023727  001404  000404          CMP     MAXDL,#260.     ;CAN A FULL SECTOR BE WRITTEN ?
3398  006520  103003                          BHIS    1$              ;BR IF IT CAN
3399  006522  013760  001404  000102          MOV     MAXDL,$NWROL(R0)  ;PUT TRANSFER SIZE INTO THE DPB
3400  006530  112760  000003  000074  1$:     MOVB    #3,$NCODE(R0)   ;COMMAND CODE
3401  006536  004737  017334                  JSR     PC,GETPAT       ;GET A PATTERN
3402  006542  110560  000075                  MOVB    R5,$NPBIC(R0)   ;PATTERN CODE TO CONTROL BLOCK
3403  006546  012760  177777  000104          MOV     #-1,$NEXT(R0)   ;SET PARAMETERS SELECTED INDICATOR
3404  006554  012701  001574                  MOV     #PARQ,R1        ;SET UP TO SEE IF BLOCK IN THE PARAMETER QUEUE
3405  006560  005711                  2$:     TST     (R1)            ;SEE IF AT END OF TABLE
3406  006562  001405                          BEQ     REFMTX          ;BR IF AT END
3407  006564  020021                          CMP     R0,(R1)+        ;SEE IF BLOCK AT PRESENT POSITION
3408  006566  001374                          BNE     2$              ;BR IF NOT
3409  006570  005041                          CLR     -(R1)           ;CLEAR THE ENTRY
3410  006572  004737  017570                  JSR     PC,CMPRES       ;COMPRESS THE TABLE
3411  006576  000207                  REFMTX: RTS     PC              ;RETURN
```

```
3412
3413                                    ;PROCESS THE ORDER TERMINATION
3414
3415  006600  111037  001246    PROCES: MOVB    (R0),UNIT       ;DRIVE NUMBER FOR ANY ERROR MESSAGES
3416  006604  005760  000016            TST     $STATUS(R0)     ;SEE IF DRIVER SIGNALED AN ERROR
3417  006610  100427                    BMI     ERPROC          ;BR IF ERROR
3418  006612  032760  100000  000234    BIT     #BIT15,$RPCS1(R0)  ;SEE IF 'SC' SET
3419  006620  001410                    BEQ     1$              ;BR IF NOT SET
3420  006622  032760  040000  000234    BIT     #BIT14,$RPCS1(R0)  ;SEE IF 'TRE' SET
3421  006630  001017                    BNE     ERPROC          ;BR IF SET
3422  006632  032760  040000  000246    BIT     #BIT14,$RPDS1(R0)  ;SEE IF 'ERR' SET
3423  006640  001013                    BNE     ERPROC          ;BR IF SET
3424  006642  004737  013030    1$:     JSR     PC,CKERR        ;NO ERROR, CHECK ERROR BITS ANYWAY
3425  006646  004737  013130            JSR     PC,CKBUS        ;NO ERROR, CHECK BUS ADDR & WC
3426  006652  032777  000002  172260    BIT     #SW01,@SWR      ;DATA COMPARE ?
3427  006660  001002                    BNE     2$              ;BR IF NOT
3428  006662  004737  013214            JSR     PC,CMPAR        ;NO ERROR, COMPARE DATA
3429  006666  000207            2$:     RTS     PC              ;RETURN
3430
3431                                    ;ORDER TERMINATED WITH AN ERROR - PROCESS THE ERROR
3432
3433  006670  032760  000200  000016  ERPROC: BIT     #BIT07,$STATUS(R0) ;DONE BIT SET ?
3434  006676  001402                    BEQ     ERPRC1          ;BR IF ORDER DIDN'T COMPLETE NORMALLY
3435  006700  000137  007324            JMP     DONE            ;PROCESS ERROR WITH 'DONE' BIT SET
3436
3437                                    ;PROCESS ORDER COMPLETION WITH 'ERROR' & 'DONE NOT' BITS
3438
3439  006704  032760  010000  000016  ERPRC1: BIT     #BIT12,$STATUS(R0) ;SEE IF DRIVE WAS UNSAFE
3440  006712  001025                    BNE     PUNSAF          ;BR IF YES
3441  006714  032760  004000  000016            BIT     #BIT11,$STATUS(R0) ;PARITY ERROR OCCURRED
3442  006722  001055                    BNE     UCPAR           ;BR IF IT DID
3443  006724  032760  002000  000016            BIT     #BIT10,$STATUS(R0) ;FATAL PARITY ERROR?
3444  006732  001056                    BNE     FALPAR          ;BR IF THERE IS ONE
3445  006734  032760  001000  000016            BIT     #BIT09,$STATUS(R0) ;TIMEOUT?
3446  006742  001076                    BNE     SWTIM           ;BR IF YES
3447  006744  032760  040002  000016            BIT     #BIT14!BIT01,$STATUS(R0)  ;DRIVE WENT OFFLINE ?
3448  006752  001111                    BNE     OFLIN           ;BR IF IT DID
3449  006754  032760  000004  000016            BIT     #BIT2,$STATUS(R0) ;PORT REQUEST TIME OUT ?
3450  006762  001141                    BNE     PRTIM           ;BR IF IT DID
3451  006764  000207                    RTS     PC              ;ERROR. RETURN
3452
3453                                    ;DRIVE IS PERSISTENTLY UNSAFE
3454
3455  006766  104401  001165    PUNSAF: TYPE    ,$CRLF          ;CR-LF
3456  006772  104401  045450            TYPE    ,EM12           ;'DRIVE UNSAFE' MESSAGE
3457  006776  104401  052733            TYPE    ,DRNUM          ;DRIVE NUMBER
3458  007002  013746  001246            MOV     UNIT,-(SP)      ;;SAVE UNIT FOR TYPEOUT
3459                                                            ;;TYPE DRIVE NUMBER
3460  007006  104403                    TYPOS                   ;;GO TYPE--OCTAL ASCII
3461  007010     002                    .BYTE   2               ;;TYPE 2 DIGIT(S)
3462  007011     000                    .BYTE   0               ;;SUPPRESS LEADING ZEROS
3463  007012  104401  001165            TYPE    ,$CRLF          ;CR-LF
3464  007016  004737  020240            JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3465  007022  104414  045450            DISPLY  ,EM12           ;PERSISTENT DEVICE UNSAFE MESSAGE
3466  007026  004737  020304            JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
3467  007032  004737  020712            JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
```

```
3468  007036  004737  021366          JSR     PC,LINE4        ;PRINT LINE 4 OF THE ERROR MESSAGE
3469  007042  004737  023464          JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3470  007046  004737  022022          JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3471  007052  000137  026506          JMP     DROP            ;DROP THE DRIVE
3472
3473                                  ;UNCORRECTABLE MASSBUS PARITY ERROR OCCURRED
3474
3475  007056  104401  001165  UCPAR:  TYPE    ,$CRLF          ;CR-LF
3476  007062  104401  045352          TYPE    ,EM10           ;'UNCORRECTABLE PARITY ERROR' MESSAGE
3477  007066  000404                  BR      FALPR1          ;FINISH PROCESSING THE ERROR
3478
3479                                  ;'FATAL' MASSBUS PARITY ERROR OCCURRED
3480
3481  007070  104401  001165  FALPAR: TYPE    ,$CRLF          ;CR-LF
3482  007074  104401  045415          TYPE    ,EM11           ;'FATAL PARITY ERROR' MESSAGE
3483  007100  104401  052733  FALPR1: TYPE    ,DRNUM          ;DRIVE NUMBER
3484  007104  013746  001246          MOV     UNIT,-(SP)      ;;SAVE UNIT FOR TYPEOUT
3485                                                          ;;TYPE DRIVE NUMBER
3486  007110  104403                  TYPOS                   ;;GO TYPE--OCTAL ASCII
3487  007112     002                  .BYTE   2               ;;TYPE 2 DIGIT(S)
3488  007113     000                  .BYTE   0               ;;SUPPRESS LEADING ZEROS
3489  007114  104401  001165          TYPE    ,$CRLF          ;CR-LF
3490  007120  004737  023464          JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3491  007124  032777  100000  172006  BIT     #SW15,@SWR      ;HALT ON ERROR ?
3492  007132  001401                  BEQ     1$              ;BR IF NOT
3493  007134  000000                  HALT                    ;ERROR HALT
3494  007136  000207          1$:     RTS     PC
3495
3496                                  ;SOFTWARE TIMEOUT OCCURRED
3497
3498  007140  004737  020240  SWTIM:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3499  007144  104414  045501          DISPLY  ,EM13           ;PRINT THE TIME OUT MESSAGE
3500  007150  004737  020304          JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
3501  007154  004737  020712          JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
3502  007160  004737  021366          JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
3503  007164  004737  023464          JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3504  007170  004737  022022          JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3505  007174  000207                  RTS     PC              ;RETURN
3506
3507                                  ;DRIVE WENT OFFLINE
3508
3509  007176  104401  001165  OFLIN:  TYPE    ,$CRLF          ;CR-LF
3510  007202  104401  045553          TYPE    ,EM14           ;'DRIVE WENT OFFLINE' MESSAGE
3511  007206  104401  052733          TYPE    ,DRNUM          ;DRIVE NUMBER
3512  007212  013746  001246          MOV     UNIT,-(SP)      ;;SAVE UNIT FOR TYPEOUT
3513                                                          ;;TYPE DRIVE NUMBER
3514  007216  104403                  TYPOS                   ;;GO TYPE--OCTAL ASCII
3515  007220     002                  .BYTE   2               ;;TYPE 2 DIGIT(S)
3516  007221     000                  .BYTE   0               ;;SUPPRESS LEADING ZEROS
3517  007222  104401  001165          TYPE    ,$CRLF          ;CR-LF
3518  007226  004737  020240          JSR     PC,LINE1        ;PRINT LINE 1 OF THE ERROR MESSAGE
3519  007232  104414  045553          DISPLY  ,EM14           ;PRINT OFFLINE MESSAGE
3520  007236  004737  020304          JSR     PC,LINE2        ;PRINT LINE 2 OF THE ERROR MESSAGE
3521  007242  004737  020712          JSR     PC,LINE3        ;PRINT LINE 3 OF THE ERROR MESSAGE
3522  007246  004737  021366          JSR     PC,LINE4        ;PRINT LINE 4 OF THE ERROR MESSAGE
3523  007252  004737  023464          JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
```

```
3524  007256  004737  022022                  JSR     PC,LINE7        ;PRINT LINE 7 OF THE ERROR MESSAGE
3525  007262  000137  026506                  JMP     DROP            ;DROP THE DRIVE
3526
3527                                  ;PORT REQUEST TIMEOUT ERROR
3528
3529  007266  004737  020240          PRTIM:  JSR     PC,LINE1        ;TYPE LINE 1 OF THE ERROR MESSAGE
3530  007272  104414  045576                  DISPLY  .EM15           ;PRINT PORT TIME OUT MESSAGE
3531  007276  004737  020304                  JSR     PC,LINE2        ;TYPE LINE 2 OF THE ERROR MESSAGE
3532  007302  004737  020712                  JSR     PC,LINE3        ;TYPE LINE 3 OF THE ERROR MESSAGE
3533  007306  004737  021366                  JSR     PC,LINE4        ;TYPE LINE 4 OF THE ERROR MESSAGE
3534  007312  004737  023464                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3535  007316  004737  022022                  JSR     PC,LINE7        ;TYPE LINE 7 OF THE ERROR MESSAGE
3536  007322  000207                          RTS     PC              ;RETURN
3537
3538                                  ;PROCESS ORDER COMPLETION WITH 'ERROR' & 'DONE' BITS SET
3539
3540  007324  032760  000030  000016  DONE:   BIT     #BIT04!BIT03,STATUS(R0) ;UNSAFE OCCURRED
3541  007332  001402                          BEQ     .+6             ;BR IF NOT
3542  007334  000137  012526                  JMP     UNSAF           ;REPORT UNSAFE
3543  007340  032760  040000  000244          BIT     #BIT14,$RPCS2(R0)  ;IS 'NCE' SET ?
3544  007346  001006                          BNE     1$              ;BR IF SET
3545  007350  032760  040000  000246          BIT     #BIT14,$RPDS1(R0) ;CHECK 'ERR'
3546  007356  001002                          BNE     1$              ;BR IF SET
3547  007360  000137  012272                  JMP     TRFER           ;PROCESS 'TRE'
3548  007364  032760  000400  000250  1$:     BIT     #BIT08,$RPER1(R0) ;'NCRC' SET?
3549  007372  001402                          BEQ     .+6             ;BR IF NOT
3550  007374  000137  010750                  JMP     NCRCER          ;PROCESS 'NCRC'
3551  007400  032760  000020  000250          BIT     #BIT04,$RPER1(R0) ;'FMT' SET?
3552  007406  001402                          BEQ     .+6             ;BR IF NOT SET
3553  007410  000137  011132                  JMP     CKFMT           ;CHECK FORMAT ERROR
3554  007414  032760  000200  000250          BIT     #BIT07,$RPER1(R0) ;'NCE' SET?
3555  007422  001402                          BEQ     .+6             ;BR IF NOT SET
3556  007424  000137  011326                  JMP     CKNCE           ;CHECK 'NCE' ERROR
3557  007430  032760  020000  000250          BIT     #BIT13,$RPER1(R0) ;'OPI' SET?
3558  007436  001402                          BEQ     .+6             ;BR IF NOT SET
3559  007440  000137  011626                  JMP     OPIER           ;REPORT 'OPI'
3560  007444  032760  000010  000250          BIT     #BIT3,$RPER1(R0) ;'PAR' SET?
3561  007452  001402                          BEQ     .+6             ;BR IF NOT SET
3562  007454  000137  011760                  JMP     PARER           ;REPORT 'PAR'
3563  007460  032760  000040  000250          BIT     #BIT5,$RPER1(R0) ;'WCF' SET?
3564  007466  001402                          BEQ     .+6             ;BR IF NOT SET
3565  007470  000137  012430                  JMP     WCFER           ;REPORT 'WCF'
3566  007474  032760  002000  000250          BIT     #BIT10,$RPER1(R0) ;'IAE' SET?
3567  007502  001402                          BEQ     .+6             ;BR IF NOT SET
3568  007504  000137  012052                  JMP     IAEER           ;REPORT 'IAE'
3569  007510  032760  004000  000250          BIT     #BIT11,$RPER1(R0) ;'WLE' SET?
3570  007516  001402                          BEQ     .+6             ;BR IF NOT SET
3571  007520  000137  012104                  JMP     WLEER           ;REPORT 'WLE'
3572  007524  032760  001000  000250          BIT     #BIT9,$RPER1(R0) ;'AOE' SET?
3573  007532  001405                          BEQ     2$              ;BR IF NOT SET
3574  007534  032760  002000  000246          BIT     #BIT10,$RPDS1(R0) ;'LST' SET?
3575  007542  001401                          BEQ     2$              ;BR IF NOT SET
3576  007544  000207                          RTS     PC              ;'AOE' & 'LST' SET, EXIT
3577  007546  032760  010000  000250  2$:     BIT     #BIT12,$RPER1(R0)  ;SEE IF 'DTE' SET
3578  007554  001402                          BEQ     .+6             ;BR IF NOT
3579  007556  000137  011736                  JMP     DTEER           ;REPORT 'DTE' ERROR
```

```
3580  007562  032760  040000  000244          BIT     #BIT14,$RPCS2(R0)  ;SEE IF 'WCK' SET
3581  007570  001402                          BEQ     .+6                ;BR IF NOT SET
3582  007572  000137  010422                  JMP     WCKER              ;REPORT 'WCK'
3583  007576  005760  000250                  TST     $RPER1(R0)         ;SEE IF 'DCK' SET
3584  007602  100002                          BPL     .+6                ;BR IF NOT
3585  007604  000137  007630                  JMP     DCKER              ;PROCESS 'DCK'
3586  007610  032760  140000  000276          BIT     #BIT15!BIT14,$RPER3(R0) ;'SKI' OR 'OCYL' SET
3587  007616  001402                          BEQ     .+6                ;BR IF NOT SET
3588  007620  000137  012372                  JMP     SKIER              ;REPORT ERROR
3589  007624  000137  011100                  JMP     DRVER              ;REPORT DRIVE ERROR
3590
3591                                  ;PROCESS DATA ('DCK') CHECK ERROR
3592
3593  007630  022760  010042  000300  DCKER:  CMP     #10042,$RPEC1(R0)  ;VALID POSITION COUNT ?
3594  007636  101406                          BLOS    1$                 ;BR IF NOT VALID
3595  007640  005760  000300                  TST     $RPEC1(R0)         ;POSITION COUNT 0 ?
3596  007644  001403                          BEQ     1$                 ;BR IF 0'S
3597  007646  005760  000302                  TST     $RPEC2(R0)         ;VALUE IN PATTERN REGISTER ?
3598  007652  001026                          BNE     4$                 ;BR IF YES
3599  007654  004737  020240          1$:     JSR     PC,LINE1           ;TYPE FIRST LINE OF ERROR MESSAGE
3600  007660  104414  047203                  DISPLY  ,EM45              ;TYPE 'ECC LOGIC ERROR'
3601  007664  004737  020304                  JSR     PC,LINE2           ;TYPE LINE 2 OF ERROR MESSAGE
3602  007670  004737  023464                  JSR     PC,INCTOT          ;INCREMENT TOTAL ERROR COUNT
3603  007674  012737  000003  001252          MOV     #3,RETRY           ;RETRY COUNT
3604  007702  004737  015440                  JSR     PC,$RETRY          ;RETRY THE ORDER
3605  007706  000403                          BR      2$                 ;RETRY WAS NOT SUCCESSFUL
3606  007710  004737  021740                  JSR     PC,LINE6C          ;TYPE LINE 6C OF ERROR MESSAGE
3607  007714  000402                          BR      3$                 ;FINISH THE ERROR REPORT
3608  007716  004737  021746          2$:     JSR     PC,LINE6D          ;TYPE LINE 6D OF ERROR MESSAGE
3609  007722  004737  022022          3$:     JSR     PC,LINE7           ;TYPE LINE 7 OF ERROR MESSAGE
3610  007726  000402                          BR      5$                 ;EXIT
3611  007730  004737  020102          4$:     JSR     PC,SPOTCK          ;SEE IF ERROR AT A BAD SPOT ON THE PACK
3612  007734  000207                  5$:     RTS     PC                 ;IT IS, DON'T REPORT IT
3613  007736  126027  000024  000001          CMPB    $CODE(R0),#1       ;IS ORDER A WRITE CHECK ?
3614  007744  101002                          BHI     6$                 ;BR IF NOT
3615  007746  000137  010422                  JMP     WCKER              ;REPORT ERROR UNDER WRITE CHECK PROCESSING
3616  007752  004737  020240          6$:     JSR     PC,LINE1           ;PRINT LINE 1 OF ERROR MESSAGE
3617  007756  104414  045653                  DISPLY  ,EM21              ;DATA CHECK ERROR
3618  007762  004737  020304          DCKER1: JSR     PC,LINE2           ;PRINT LINE 2 OF ERROR MESSAGE
3619  007766  004737  020712                  JSR     PC,LINE3           ;PRINT LINE 3 OF ERROR MESSAGE
3620  007772  004737  021366                  JSR     PC,LINE4           ;PRINT LINE 4 OF ERROR MESSAGE
3621  007776  004737  015102                  JSR     PC,PRTBAD          ;SEE IF BAD SECTOR TO BE PRINTED
3622  010002  012737  110100  001250          MOV     #BIT15!BIT12!BIT06,MASK  ;LOAD ERROR MASK
3623  010010  032760  010100  000250          BIT     #BIT12!BIT06,$RPER1(R0)  ;CHECK 'DTE' & 'ECH'
3624  010016  001003                          BNE     1$                 ;BR IF SET
3625  010020  004737  021712                  JSR     PC,LINE6           ;PRINT LINE 6 OF ERROR MESSAGE
3626  010024  000541                          BR      8$                 ;FINISH THE ERROR REPORT
3627  010026  012737  000020  001252  1$:     MOV     #16.,RETRY         ;RETRY COUNT
3628  010034  005001                          CLR     R1                 ;R1 IS OFFSET CODE POINTER
3629  010036  032760  000002  000262          BIT     #BIT01,$RPDT(R0)   ;IS DRIVE AN RP06 ?
3630  010044  001002                          BNE     16$                ;BR IF IT IS
3631  010046  012701  000007                  MOV     #7,R1              ;INCREMENT PAST RP06 OFFSET CODES
3632  010052  004737  016066          16$:    JSR     PC,RELBUF          ;RELEASE THE BUFFER
3633  010056  004737  022354                  JSR     PC,READDR          ;GET THE ADDRESS OF THE ERROR SECTOR
3634  010062  112660  000011                  MOVB    (SP)+,$TRK(R0)     ;TRACK ADDRESS OF ERROR SECTOR
3635  010066  112660  000010                  MOVB    (SP)+,$SEC(R0)     ;SECTOR ADDRESS OF ERROR SECTOR
```

```
3636  010072  016060  000272  000012          MOV     $RPCC(R0),$CYL(R0)  ;PRESENT CYLINDER
3637  010100  026060  000022  000020          CMP     $SSEC(R0),$WRDL(R0) ;SEE IF TRANSFER LENGTH LESS THAN 1 SECTOR
3638  010106  103010                          BHIS    15$                 ;BR IF IT IS; USE PRESENT TRANSFER LENGTH
3639  010110  016060  000022  000020          MOV     $SSEC(R0),$WRDL(R0) ;CHANGE TRANSFER SIZE TO 1 SECTOR
3640  010116  016060  000020  000004          MOV     $WRDL(R0),$WRDM(R0) ;SETUP WORD COUNT FOR OPERATION
3641  010124  005460  000004                  NEG     $WRDM(R0)           ;CHANGE COUNT TO 2'S COMP
3642  010130  005046                  15$:    CLR     -(SP)               ;SPACE FOR NEW BUFFER ADDRESS
3643  010132  004737  015732                  JSR     PC,GETBUF           ;GET A BUFFER ADDRESS
3644  010136  012660  000006                  MOV     (SP)+,$BUF(R0)      ;NEW BUFFER ADDRESS TO DPB
3645  010142  004737  016430          2$:     JSR     PC,GODRIV           ;RETRY
3646  010146  005760  000016          3$:     TST     $STATUS(R0)         ;TEST FOR DONE
3647  010152  001775                          BEQ     3$                  ;BR IF NOT DONE
3648  010154  100075                          BPL     10$                 ;BR IF NOT ERROR
3649  010156  032760  000200  000016          BIT     #BIT7,$STATUS(R0)   ;SEE IF ORDER TERMINIATED NORMALLY
3650  010164  001006                          BNE     14$                 ;BR IF NOT
3651  010166  004737  023464                  JSR     PC,INCTOT           ;INCREMENT TOTAL ERROR COUNT
3652  010172  104414  051370                  DISPLY  ,LIN8M              ;'DIFFERENT ERROR DURING RETRY'
3653  010176  000137  006704                  JMP     ERPRC1              ;SEE WHICH ERROR
3654  010202  033760  001250  000250  14$:    BIT     MASK,$RPER1(R0)     ;LOOK AT CURRENT ERROR
3655  010210  001430                          BEQ     5$                  ;BR IF DIFFERENT ERROR
3656  010212  032760  010100  000250          BIT     #BIT12!BIT6,$RPER1(R0)  ;'ECH' OR 'DTE' STILL SET ?
3657  010220  001437                          BEQ     7$                  ;BR IF NEITHER SET
3658  010222  105237  001253                  INCB    RETRY+1             ;INCREMENT RETRY COUNT
3659  010226  123737  001252  001253          CMPB    RETRY,RETRY+1       ;DONE ?
3660  010234  001342                          BNE     2$                  ;BR IF NOT
3661  010236  005201                          INC     R1                  ;INCREMENT TABLE INDEX
3662  010240  116137  002220  044771          MOVB    OFFCOD(R1),GENDPB+$FMT  ;OFFSET CODE
3663  010246  001435                          BEQ     9$                  ;BR IF END OF OFFSET TABLE
3664  010250  062737  000002  001252          ADD     #2,RETRY            ;NEW RETRY LIMIT
3665  010256  004737  015332                  JSR     PC,OFFST            ;OFFSET
3666  010262  005737  045006          4$:     TST     GENDPB+$STATUS      ;SEE IF FINISHED WITH OFFSET
3667  010266  001775                          BEQ     4$                  ;BR IF NOT
3668  010270  100324                          BPL     2$                  ;BR IF NO ERROR PERFORMING OFFSET
3669  010272  004737  022270          5$:     JSR     PC,LINE8            ;PRINT LINE 8 OF ERROR MESSAGE
3670  010276  004737  023370          6$:     JSR     PC,INCHRD           ;INCREMENT 'HARD' ERROR COUNT
3671  010302  004737  023464                  JSR     PC,INCTOT           ;INCREMENT TOTAL ERROR COUNT
3672  010306  004737  022022                  JSR     PC,LINE7            ;PRINT LINE 7 OF ERROR MESSAGE
3673  010312  004737  015102                  JSR     PC,PRTBAD           ;PRINT THE BAD SECTOR
3674  010316  000436                          BR      13$                 ;CLEAN UP AND RETURN
3675  010320  004737  021732          7$:     JSR     PC,LINE6B           ;PRINT LINE 6B OF ERROR MESSAGE
3676  010324  004737  021650                  JSR     PC,LINE5B           ;PRINT LINE 5B OF THE ERROR MESSAGE
3677  010330  004737  023344          8$:     JSR     PC,INCSOF           ;INCREMENT 'SOFT' ERROR COUNT
3678  010334  004737  014342                  JSR     PC,ECC              ;CORRECT THE ERROR USING ECC AND CHECK IT
3679  010340  000407                          BR      11$                 ;COMPARE THE BUFFER
3680  010342  004737  021746          9$:     JSR     PC,LINE6D           ;PRINT LINE 6D OF ERROR MESSAGE
3681  010346  000753                          BR      6$                  ;INCREMENT ERROR COUNT
3682  010350  004737  021724          10$:    JSR     PC,LINE6A           ;PRINT LINE 6A OF ERROR MESSAGE
3683  010354  004737  023344                  JSR     PC,INCSOF           ;INCREMENT 'SOFT' ERROR COUNT
3684  010360  012737  000001  001300  11$:    MOV     #1,FRSTER           ;SET PROCESSING 'DCKER' INDICATOR
3685  010366  004737  013232                  JSR     PC,CMPARD           ;COMPARE THE BUFFER
3686  010372  105737  001301                  TSTB    FRSTER+1            ;ERROR IN COMPARE ?
3687  010376  100406                          BMI     13$                 ;BRANCH IF ERROR
3688  010400  004737  023464                  JSR     PC,INCTOT           ;INCREMENT TOTAL ERROR COUNT
3689  010404  104414  051602                  DISPLY  ,LIN9G              ;'DATA COMPARE OK' MESSAGE
3690  010410  004737  022022          12$:    JSR     PC,LINE7            ;PRINT LINE 7 OF ERROR MESSAGE
3691  010414  004737  006434          13$:    JSR     PC,REFMT            ;REFORMAT THE ERROR SECTOR
```

```
3692  010420  000207                          RTS     PC              ;RETURN
3693
3694                                  ;WRITE CHECK ERROR PROCESSING
3695
3696  010422  032760  100000  000250  WCKER:  BIT     #BIT15,$RPER1(R0)  ;SEE IF 'DCK' SET ALSO
3697  010430  001034                          BNE     2$              ;BR IF IT IS
3698  010432  004737  020240                  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3699  010436  104414  045757                  DISPLY  ,EM23           ;PRINT WCE & DCK NOT
3700  010442  005037  001250                  CLR     MASK            ;CLEAR ERROR MASK
3701  010446  004737  020304                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
3702  010452  004737  020712                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
3703  010456  004737  021366                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
3704  010462  004737  021456                  JSR     PC,LINE5        ;PRINT LINE 5 OF ERROR MESSAGE
3705  010466  004737  023464                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3706  010472  012737  000003  001252          MOV     #3,RETRY        ;RETRY LIMIT
3707  010500  004737  015440                  JSR     PC,$RETRY       ;RETRY THE OPERATION
3708  010504  000403                          BR      1$              ;RETRY UNSUCESSFUL
3709  010506  004737  021740                  JSR     PC,LINE6C       ;PRINT LINE 6C OF ERROR MESSAGE
3710  010512  000502                          BR      8$              ;FINISH PROCESSING THE ERROR
3711  010514  004737  021746          1$:     JSR     PC,LINE6D       ;PRINT LINE 6D OF ERROR MESSAGE
3712  010520  000506                          BR      10$             ;FINISH PROCESSING THE ERROR
3713  010522  004737  020102          2$:     JSR     PC,SPOTCK       ;SEE IF ERROR AT BAD SPOT ON THE PACK
3714  010526  000507                          BR      11$             ;EXIT IF AT BAD SPOT ON PACK
3715  010530  004737  020240                  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3716  010534  012737  045704  010562          MOV     #EM22,13$       ;ASSUME THAT EM22 WILL BE PRINTED
3717  010542  032760  040000  000244          BIT     #BIT14,$RPCS2(R0)  ;DID 'WCK' ALSO SET ?
3718  010550  001003                          BNE     12$             ;BR IF IT DID
3719  010552  012737  046605  010562          MOV     #EM37,13$       ;MESSAGE FOR 'DCK' AND 'WCK' NOT DURING
3720                                                                  ;WRITE CHECK
3721  010560  104414                  12$:    DISPLY                  ;TYPE THE ERROR MESSAGE
3722  010562  000000                  13$:    .WORD   0               ;MESSAGE ADDRESS GOES HERE
3723  010564  004737  020304                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
3724  010570  004737  020712                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
3725  010574  004737  021366                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
3726  010600  004737  021456                  JSR     PC,LINE5        ;PRINT LINE 5 OF ERROR MESSAGE
3727  010604  032760  000100  000250          BIT     #BIT06,$RPER1(R0)  ;ECH SET ALSO ?
3728  010612  001442                          BEQ     8$              ;FINISH PROCESSING THE ERROR
3729  010614  012737  000020  001252  3$:     MOV     #16.,RETRY      ;RETRY LIMIT - 16 (10)
3730  010622  004737  016430          4$:     JSR     PC,GODRIV       ;RETRY THE ORDER
3731  010626  005760  000016          5$:     TST     $TATUS(R0)      ;ORDER FINISHED ?
3732  010632  001775                          BEQ     5$              ;BR IF NOT
3733  010634  100405                          BMI     6$              ;BR IF ERROR ON ORDER
3734  010636  105237  001253                  INCB    RETRY+1         ;INCREMENT RETRY COUNT
3735  010642  004737  021740                  JSR     PC,LINE6C       ;PRINT LINE 6C OF ERROR MESSAGE
3736  010646  000431                          BR      9$              ;FINISH ERROR PROCESSING
3737  010650  105237  001253          6$:     INCB    RETRY+1         ;INCREMENT RETRY COUNT
3738  010654  123737  001252  001253          CMPB    RETRY,RETRY+1   ;DONE ?
3739  010662  001714                          BEQ     1$              ;BR IF AT RETRY LIMIT
3740  010664  032760  100000  000250          BIT     #BIT15,$RPER1(R0)  ;'DCK' SET
3741  010672  001407                          BEQ     7$              ;BR IF NOT - DIFFERENT ERROR
3742  010674  032760  000100  000250          BIT     #BIT06,$RPER1(R0)  ;'ECH' ALSO SET ?
3743  010702  001347                          BNE     4$              ;BR IF IT IS, RETRY ORDER
3744  010704  004737  021740                  JSR     PC,LINE6C       ;PRINT LINE 6C OF ERROR MESSAGE
3745  010710  000403                          BR      8$              ;FINISH PROCESSING ERROR
3746  010712  004737  022270          7$:     JSR     PC,LINE8        ;PRINT LINE 8 - 'DIFFERENT ERROR '
3747  010716  000405                          BR      9$              ;FINISH PROCESSING ERROR
```

```
3745  010720  004737  023464          8$:     JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3749  010724  004737  022022                  JSR     PC,LINE7        ;FINISH THE ERROR MESSAGE
3750  010730  000406                          BR      11$             ;EXIT
3751  010732  004737  023464          9$:     JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3752  010736  004737  022022          10$:    JSR     PC,LINE7        ;FINISH THE ERROR MESSAGE
3753  010742  004737  006434                  JSR     PC,REFMT        ;REFORMAT THE SECTOR IN ERROR
3754  010746  000207                  11$:    RTS     PC              ;RETURN
3755
3756                                  ;REPORT 'HCRC' ERROR
3757
3758  010750  004737  020102          HCRCER: JSR     PC,SPOTCK       ;SEE IF ERROR AT PACK BAD SPOT
3759  010754  000450                          BR      3$              ;EXIT IF IT IS
3760  010756  004737  020240                  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3761  010762  104414  045632                  DISPLY  .EM20           ;REPORT 'HCRC'
3762  010766  004737  020304                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
3763  010772  004737  020712                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
3764  010776  004737  021366                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
3765  011002  032760  040000  000244          BIT     #BIT14,$RPCS2(R0)  ;'WCE' ERROR ALSO ?
3766  011010  001402                          BEQ     1$              ;BR IF NOT
3767  011012  004737  021456                  JSR     PC,LINE5        ;DISPLAY WORDS WHICH CAUSED 'WCE'
3768  011016  004737  023344          1$:     JSR     PC,INCSOF       ;INCREMENT 'SOFT' ERROR COUNT
3769  011022  004737  023464                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3770  011026  012737  000400  001250          MOV     #BIT8,MASK      ;SET ERROR MASK
3771  011034  012737  000003  001252          MOV     #3,RETRY        ;RETRY LIMIT
3772  011042  004737  015440                  JSR     PC,$RETRY       ;RETRY ORDER
3773  011046  000405                          BR      2$              ;RETRY NOT SUCESSFUL
3774  011050  004737  021740                  JSR     PC,LINE6C       ;PRINT LINE 6C OF ERROR MESSAGE
3775  011054  004737  022022                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3776  011060  000406                          BR      3$              ;EXIT
3777  011062  004737  021746          2$:     JSR     PC,LINE6D       ;PRINT LINE 6D OF ERROR MESSAGE
3778  011066  004737  022022                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3779  011072  004737  006434                  JSR     PC,REFMT        ;REFORMAT THE ERROR SECTOR
3780  011076  000207                  3$:     RTS     PC              ;RETURN
3781
3782                                  ;REPORT DRIVE ERROR
3783
3784  011100  004737  020240          DRVER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3785  011104  104414  046247                  DISPLY  .EM30           ;REPORT DRIVE ERROR
3786  011110  004737  020304                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
3787  011114  004737  020712                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
3788  011120  004737  023464                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3789  011124  004737  022022                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3790  011130  000207                          RTS     PC              ;RETURN
3791
3792                                  ;PROCESS FORMAT ('FER') ERROR
3793
3794  011132  032760  000400  000250  CKFMT:  BIT     #BIT8,$RPER1(R0) ;'HCRC' SET ON ORIGINAL ERROR ?
3795  011140  001402                          BEQ     1$              ;BR IF NOT SET
3796  011142  000137  010750                  JMP     HCRCER          ;REPORT HCRC ERROR
3797  011146  004737  022354          1$:     JSR     PC,READDR       ;GET CORRECTED TRACK & SECTOR ADDRSSES
3798  011152  004737  015356                  JSR     PC,READHD       ;READ HEADER
3799  011156  032737  000400  045024          BIT     #BIT8,GENREG+RPER1 ;'HCRC' SET WHEN HEADER READ?
3800  011164  001002                          BNE     2$              ;BR IF 'HCRC' SET
3801  011166  000137  012132                  JMP     FMTER           ;NO, ERROR IS 'FMT' ONLY
3802  011172  004737  020102          2$:     JSR     PC,SPOTCK       ;SEE IF ERROR AT BAD SPOT ON THE PACK
3803  011176  000452                          BR      5$              ;EXIT IF IT IS
```

```
3804  011200  004737  020240                      JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3805  011204  104414  046036                      DISPLY  EM24            ;HEADER READ ERROR - FMT BIT DROPPED UP
3806  011210  004737  020304                      JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
3807  011214  004737  020712                      JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
3808  011220  004737  021366                      JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
3809  011224  032760  040000  000244              BIT     #BIT14,$RPCS2(R0)  ;'WCE' ERROR ALSO ?
3810  011232  001402                              BEQ     3$              ;BR IF NOT
3811  011234  004737  021456                      JSR     PC,LINE5        ;DISPLAY WORDS WHICH CAUSED 'WCE'
3812  011240  004737  021556          3$:         JSR     PC,LINE5A       ;DISPLAY HEADER
3813  011244  004737  023344                      JSR     PC,INCSOF       ;INCREMENT SOFT ERROR COUNT
3814  011250  004737  023464                      JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3815  011254  012737  000020  001250              MOV     #BIT4,MASK      ;SET ERROR MASK
3816  011262  012737  000003  001252              MOV     #3,RETRY        ;RETRY LIMIT
3817  011270  004737  015440                      JSR     PC,$RETRY       ;RETRY THE ORDER
3818  011274  000405                              BR      4$              ;RETRY NOT SUCESSFUL
3819  011276  004737  021740                      JSR     PC,LINE6C       ;PRINT LINE 6C OF ERROR MESSAGE
3820  011302  004737  022022                      JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3821  011306  000406                              BR      5$              ;EXIT
3822  011310  004737  021746          4$:         JSR     PC,LINE6D       ;PRINT LINE 6D OF ERROR MESSAGE
3823  011314  004737  022022                      JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3824  011320  004737  006434                      JSR     PC,REFMT        ;REFORMAT THE ERROR SECTOR
3825  011324  000207                  5$:         RTS     PC              ;RETURN
3826
3827                                   ;PROCESS HEADER COMPARE ('HCE') ERROR
3828
3829  011326  032760  000400  000250  CKHCE:     BIT     #BIT8,$RPER1(R0) ;HCRC SET ON ORIGINAL ERROR ?
3830  011334  001402                              BEQ     1$              ;BR IF NOT SET
3831  011336  000137  010750                      JMP     HCRCER          ;REPORT HEADER CRC ERROR
3832  011342  004737  022354          1$:         JSR     PC,READDR       ;GET CURRENT SECTOR & TRACK ADDRS
3833  011346  004737  015356                      JSR     PC,READHD       ;READ HEADER OF CURRENT SECTOR
3834  011352  032737  000400  045024              BIT     #BIT8,GENREG+RPER1 ;'HCRC' SET ?
3835  011360  001016                              BNE     3$              ;BR IF SET
3836  011362  042737  010000  054516              BIC     #BIT12,CYLDER   ;CLEAR FORMAT BIT FROM HEADER
3837  011370  026037  000272  054516              CMP     $RPCC(R0),CYLDER  ;CORRECT CYLINDER ?
3838  011376  001402                              BEQ     2$              ;BR IF IT IS
3839  011400  000137  011552                      JMP     POSER           ;REPORT POSITIONING ERROR
3840  011404  052737  010000  054516  2$:         BIS     #BIT12,CYLDER   ;RESTORE THE FORMAT BIT
3841  011412  000137  012210                      JMP     HCEER           ;REPORT 'HCE' ERROR
3842  011416  004737  020102          3$:         JSR     PC,SPOTCK       ;SEE IF ERROR AT BAD SPOT
3843  011422  000452                              BR      6$              ;EXIT IF IT IS
3844  011424  004737  020240                      JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3845  011430  104414  046104                      DISPLY  EM25            ;HEADER READ ERROR - 'HCE' SET
3846  011434  004737  020304                      JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
3847  011440  004737  020712                      JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
3848  011444  004737  021366                      JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
3849  011450  032760  040000  000244              BIT     #BIT14,$RPCS2(R0)  ;'WCE' ERROR ALSO ?
3850  011456  001402                              BEQ     4$              ;BR IF NOT
3851  011460  004737  021456                      JSR     PC,LINE5        ;DISPLAY WORDS WHICH CAUSED 'WCE'
3852  011464  004737  021556          4$:         JSR     PC,LINE5A       ;PRINT LINE 5 OF ERROR MESSAGE
3853  011470  004737  023344                      JSR     PC,INCSOF       ;INCREMENT SOFT ERROR COUNT
3854  011474  004737  023464                      JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3855  011500  012737  000200  001250              MOV     #BIT7,MASK      ;SET ERROR MASK
3856  011506  012737  000003  001252              MOV     #3,RETRY        ;RETRY LIMIT
3857  011514  004737  015440                      JSR     PC,$RETRY       ;RETRY THE ORDER
3858  011520  000405                              BR      5$              ;RETRY NOT SUCESSFUL
3859  011522  004737  021740                      JSR     PC,LINE6C       ;PRINT LINE 6C OF ERROR MESSAGE
```

```
3860  011526  004737  022022          JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3861  011532  000406                  BR      6$              ;EXIT
3862  011534  004737  021746    5$:   JSR     PC,LINE6D       ;PRINT LINE 6D OF ERROR MESSAGE
3863  011540  004737  022022          JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3864  011544  004737  006434          JSR     PC,REFMT        ;REFORMAT THE ERROR SECTOR
3865  011550  000207            6$:   RTS     PC              ;RETURN
3866
3867                                  ;REPORT POSSIBLE POSITIONING ERROR
3868
3869  011552  004737  015300    POSER:JSR     PC,RECALT       ;RECALIBRATE
3870  011556  004737  020240          JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3871  011562  104414  047432          DISPLY  ,EM51           ;PROGRAM DETECTED POSITIONING ERROR
3872  011566  004737  020304          JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
3873  011572  004737  020740          JSR     PC,LINE3C       ;PRINT LINE 3C OF ERROR MESSAGE
3874  011576  052737  010000  054516  BIS     #BIT12,CYLDER   ;RESTORE THE FORMAT BIT
3875  011604  004737  021556          JSR     PC,LINE5A       ;PRINT LINE 5A OF THE ERROR MESSAGE
3876  011610  004737  023440          JSR     PC,INCMIS       ;INCREMENT MISPOSITIONING COUNT
3877  011614  004737  023464          JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3878  011620  004737  022150          JSR     PC,LINE7A       ;PRINT LINE 7A OF ERROR MESSAGE
3879  011624  000207                  RTS     PC              ;EXIT
3880
3881                                  ;REPORT 'OPI' ERROR
3882
3883  011626  004737  020102    OPIER:JSR     PC,SPOTCK       ;SEE IF ERROR AT BAD SPOT
3884  011632  000207                  RTS     PC              ;RETURN IF IT IS
3885  011634  004737  020240          JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3886  011640  104414  046301          DISPLY  ,EM31           ;'OPI' ERROR
3887  011644  004737  020304          JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
3888  011650  004737  020712          JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
3889  011654  004737  021366          JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
3890  011660  004737  023464          JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3891  011664  012737  020000  001250  MOV     #BIT13,MASK     ;ERROR MASK
3892  011672  012737  000003  001252  OPIER1: MOV     #3,RETRY        ;RETRY LIMIT
3893  011700  004737  015440          JSR     PC,$RETRY       ;RETRY THE ORDER
3894  011704  000405                  BR      1$              ;RETRY UNSUCESSFUL
3895  011706  004737  021740          JSR     PC,LINE6C       ;PRINT LINE 6C OF ERROR MESSAGE
3896  011712  004737  022022          JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3897  011716  000207                  RTS     PC              ;EXIT
3898  011720  004737  021746    1$:   JSR     PC,LINE6D       ;PRINT LINE 6D OF ERROR MESSAGE
3899  011724  004737  022022          JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3900  011730  004737  006434          JSR     PC,REFMT        ;REFORMAT THE ERROR SECTOR
3901  011734  000207                  RTS     PC              ;RETURN
3902
3903                                  ;REPORT 'DTE' ERROR
3904
3905  011736  004737  020102    DTEER:JSR     PC,SPOTCK       ;SEE IF ERROR AT BAD SPOT
3906  011742  000207                  RTS     PC              ;RETURN IF IT IS
3907  011744  004737  020240          JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3908  011750  104414  046344          DISPLY  ,EM32           ;'DTE' ERROR
3909  011754  000137  007762          JMP     DCKER1          ;FINISH PROCESSING THE 'DTE' ERROR
3910
3911                                  ;REPORT 'PAR' ERROR
3912
3913  011760  004737  020240    PARER:JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3914  011764  104414  046377          DISPLY  ,EM33           ;REPORT 'PAR'
3915  011770  004737  020304          JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
```

```
3916  011774  004737  021016                  JSR     PC,LINE3E       ;PRINT LINE 3E OF ERROR MESSAGE
3917  012000  004737  021366                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
3918  012004  004737  023464                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3919  012010  012737  000010  001250          MOV     #BIT03,MASK     ;ERROR MASK
3920  012016  012737  000003  001252          MOV     #3,RETRY        ;RETRY LIMIT
3921  012024  004737  015440                  JSR     PC,$RETRY       ;RETRY ORDER
3922  012030  000405                          BR      2$              ;RETRY UNSUCESSFUL
3923  012032  004737  021740                  JSR     PC,LINE6C       ;RETRY SUCESSFUL
3924  012036  004737  022022          1$:     JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3925  012042  000207                          RTS     PC              ;EXIT
3926  012044  004737  021746          2$:     JSR     PC,LINE6D       ;PRINT LINE 6D OF ERROR MESSAGE
3927  012050  000772                          BR      1$              ;FINISH ERROR MESSAGE
3928
3929                                  ;REPORT 'IAE' ERROR
3930
3931  012052  004737  020240          IAEER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3932  012056  104414  046516                  DISPLY  ,EM35           ;REPORT 'IAE'
3933  012062  004737  020304                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
3934  012066  004737  021104                  JSR     PC,LINE3F       ;PRINT LINE 3F OF ERROR MESSAGE
3935  012072  004737  023464                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3936  012076  004737  022022                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3937  012102  000207                          RTS     PC              ;RETURN
3938
3939                                  ;REPORT 'WLE' ERROR
3940
3941  012104  004737  020240          WLEER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3942  012110  104414  046554                  DISPLY  ,EM36           ;REPORT 'WLE'
3943  012114  004737  020304                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
3944  012120  004737  023464                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3945  012124  004737  022022                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3946  012130  000207                          RTS     PC              ;RETURN
3947
3948                                  ;REPORT FORMAT ERROR
3949
3950  012132  004737  020240          FMTER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3951  012136  104414  046165                  DISPLY  ,EM26           ;FORMAT ERROR
3952  012142  004737  020304                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
3953  012146  004737  020712                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
3954  012152  004737  021366                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
3955  012156  032760  040000  000244          BIT     #BIT14,$RPCS2(R0)  ;'WCE' ERROR ALSO ?
3956  012164  001402                          BEQ     1$              ;BR IF NOT
3957  012166  004737  021456                  JSR     PC,LINE5        ;DISPLAY WORDS WHICH CAUSED 'WCE'
3958  012172  004737  021556          1$:     JSR     PC,LINE5A       ;PRINT LINE 5A OF ERROR MESSAGE
3959  012176  004737  023464                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3960  012202  004737  022022                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3961  012206  000207                          RTS     PC
3962
3963                                  ;REPORT HEADER COMPARE ERROR
3964
3965  012210  004737  020240          HCEER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3966  012214  104414  046212                  DISPLY  ,EM27           ;HEADER COMPARE ERROR
3967  012220  004737  020304                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
3968  012224  004737  020712                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
3969  012230  004737  021366                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
3970  012234  032760  040000  000244          BIT     #BIT14,$RPCS2(R0)  ;'WCE' ERROR ALSO ?
3971  012242  001402                          BEQ     1$              ;BR IF NOT
```

```
3972  012244  004737  021456            JSR     PC,LINE5        ;DISPLAY WORDS WHICH CAUSED 'WCE'
3973  012250  004737  021556      1$:   JSR     PC,LINE5A       ;PRINT LINE 5A OF ERROR MESSAGE
3974  012254  004737  023464            JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3975  012260  004737  022022            JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3976  012264  004737  006434            JSR     PC,REFMT        ;REFORMAT THE ERROR SECTOR
3977  012270  000207                    RTS     PC              ;RETURN
3978
3979                                ;PROCESS CONTROL/INTERFACE TRANSFER ERROR
3980
3981  012272  004737  020240      TRFER:JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3982  012276  104414  046667            DISPLY  .EM40           ;RH11 OR UNIBUS TRANSFER ERROR
3983  012302  004737  020304            JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
3984  012306  004737  020712            JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
3985  012312  004737  021366            JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
3986  012316  004737  023464            JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3987  012322  032760  121400  000244    BIT     #BIT15!BIT13!BIT9!BIT8,$RPCS2(R0)  ;'DLT','UPE','MXF', 'MDPE' SET ?
3988  012330  001415                    BEQ     2$              ;BR IF NONE SET
3989  012332  012737  000003  001252    MOV     #3,RETRY        ;RETRY LIMIT
3990  012340  005037  001250            CLR     MASK            ;CLEAR ERROR MASK
3991  012344  004737  015440            JSR     PC,$RETRY       ;RETRY THE OPERATION
3992  012350  000403                    BR      1$              ;RETURN HERE IF RETRY UNSUCESSFUL
3993  012352  004737  021740            JSR     PC,LINE6C       ;PRINT LINE 6C OF ERROR MESSAGE
3994  012356  000402                    BR      2$              ;FINISH THE ERROR REPORT
3995  012360  004737  021746      1$:   JSR     PC,LINE6D       ;PRINT LINE 6D OF ERROR MESSAGE
3996  012364  004737  022022      2$:   JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3997  012370  000207                    RTS     PC
3998
3999                                ;PROCESS 'SKI' OR 'OCYL' ERRORS
4000
4001  012372  004737  020240      SKIER:JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
4002  012376  104414  047343            DISPLY  .EM50           ;'SKI' OR 'OCYL' ERROR
4003  012402  004737  020304            JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
4004  012406  004737  020726            JSR     PC,LINE3B       ;PRINT LINE 3B OF ERROR MESSAGE
4005  012412  004737  023464            JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
4006  012416  004737  023414            JSR     PC,INCSKI       ;INCREMENT 'SKI' OR 'OCYL' ERROR COUNT
4007  012422  004737  022150            JSR     PC,LINE7A       ;PRINT LINE 7A OF ERROR MESSAGE
4008  012426  000207                    RTS     PC
4009
4010                                ;REPORT WRITE CLOCK FAILURE ('WCF')
4011
4012  012430  004737  020240      WCFER:JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
4013  012434  104414  046454            DISPLY  .EM34           ;REPORT WRITE CLOCK FAILURE
4014  012440  004737  020304            JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
4015  012444  004737  020720            JSR     PC,LINE3A       ;PRINT LINE 3A OF ERROR MESSAGE
4016  012450  004737  021366            JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
4017  012454  004737  023464            JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
4018  012460  004737  015102            JSR     PC,PRTBAD       ;SEE IF BAD SECTOR TO BE PRINTED
4019  012464  012737  000003  001252    MOV     #3,RETRY        ;RETRY COUNT
4020  012472  012737  000040  001250    MOV     #BIT05,MASK     ;ERROR MASK
4021  012500  004737  015440            JSR     PC,$RETRY       ;RETRY THE ORDER
4022  012504  000405                    BR      2$              ;RETURN HERE IF RETRY UNSUCESSFUL
4023  012506  004737  021740            JSR     PC,LINE6C       ;PRINT LINE 6C OF ERROR MESSAGE
4024  012512  004737  022022      1$:   JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
4025  012516  000207                    RTS     PC
4026  012520  004737  021746      2$:   JSR     PC,LINE6D       ;PRINT LINE 6D OF ERROR MESSAGE
4027  012524  000772                    BR      1$
```

```
4028
4029                                    ;PROCESS DRIVE UNSAFE ERROR
4030
4031  012526  004737  020240            UNSAF:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
4032  012532  104414  047975                    DISPLY  ,EM60           ;REPORT DRIVE UNSAFE
4033  012536  004737  020304                    JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
4034  012542  004737  020712                    JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
4035  012546  004737  023464                    JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
4036  012552  012737  000003  001252            MOV     #3,RETRY        ;RETRY COUNT
4037  012560  004737  015440                    JSR     PC,$RETRY       ;RETRY THE ORDER
4038  012564  000403                            BR      1$              ;RETRY WAS UNSUCCESSFUL
4039  012566  004737  021740                    JSR     PC,LINE6C       ;PRINT LINE 6C OF ERROR MESSAGE
4040  012572  000402                            BR      2$              ;CONTINUE WITH ERROR REPORT
4041  012574  004737  021746            1$:     JSR     PC,LINE6D       ;PRINT LINE 6D OF ERROR MESSAGE
4042  012600  004737  022022            2$:     JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
4043  012604  000207                            RTS     PC              ;RETURN
4044
4045                                    ;REPORT AN 'UNKNOWN' DATA PATTERN
4046
4047  012606  105737  001300            NOMTCH: TSTB    FRSTER          ;FIRST ERROR IN THE SECTOR ?
4048  012612  001013                            BNE     1$              ;BR IF NOT OR IF PROCESSING 'DCKER'
4049  012614  004737  020240                    JSR     PC,LINE1        ;TYPE LINE 1 OF ERROR MESSAGE
4050  012620  104414  047052                    DISPLY  ,EM43           ;'CAN'T MATCH DATA WITH PATTERN'
4051  012624  004737  020304                    JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
4052  012630  004737  020720                    JSR     PC,LINE3A       ;PRINT LINE 3A OF ERROR MESSAGE
4053  012634  004737  021366                    JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
4054  012640  000404                            BR      2$              ;CONTINUE PROCESSING ERROR
4055  012642  104414  047052            1$:     DISPLY  ,EM43           ;'CAN'T MATCH DATA WITH PATTERN'
4056  012646  104414  001165                    DISPLY  ,$CRLF          ;CR-LF
4057  012652  104414  051532            2$:     DISPLY  ,LIN9I          ;HEADER FOR DATA PRINTOUT
4058  012656  010146                            MOV     R1,-(SP)        ;ADDRESS OF WORD 1
4059  012660  004737  022302                    JSR     PC,LINOCT       ;TYPE WORD 1
4060  012664  104414  052215                    DISPLY  ,LINSP          ;SPACES
4061  012670  012146                            MOV     (R1)+,-(SP)     ;ADDRESS OF WORD 1
4062  012672  004737  022302                    JSR     PC,LINOCT       ;TYPE WORD 1
4063  012676  104414  001165                    DISPLY  ,$CRLF          ;CR-LF
4064  012702  010146                            MOV     R1,-(SP)        ;ADDRESS OF WORD 2
4065  012704  004737  022302                    JSR     PC,LINOCT       ;TYPE WORD 2
4066  012710  104414  052215                    DISPLY  ,LINSP          ;SPACES
4067  012714  012146                            MOV     (R1)+,-(SP)     ;ADDRESS OF WORD 2
4068  012716  004737  022302                    JSR     PC,LINOCT       ;TYPE WORD 2
4069  012722  104414  001165                    DISPLY  ,$CRLF          ;CR-LF
4070  012726  010146                            MOV     R1,-(SP)        ;ADDRESS OF WORD 3
4071  012730  004737  022302                    JSR     PC,LINOCT       ;TYPE WORD 3
4072  012734  104414  052215                    DISPLY  ,LINSP          ;SPACES
4073  012740  012146                            MOV     (R1)+,-(SP)     ;ADDRESS OF WORD 3
4074  012742  004737  022302                    JSR     PC,LINOCT       ;TYPE WORD 3
4075  012746  104414  001165                    DISPLY  ,$CRLF          ;CR-LF
4076  012752  010146                            MOV     R1,-(SP)        ;ADDRESS OF WORD 4
4077  012754  004737  022302                    JSR     PC,LINOCT       ;TYPE WORD 4
4078  012760  104414  052215                    DISPLY  ,LINSP          ;SPACES
4079  012764  012146                            MOV     (R1)+,-(SP)     ;ADDRESS OF WORD 4
4080  012766  004737  022302                    JSR     PC,LINOCT       ;TYPE WORD 4
4081  012772  104414  001165                    DISPLY  ,$CRLF          ;CR-LF
4082  012776  062701  000770                    ADD     #<252.*2.>,R1   ;INCREMENT BUFFER POINTER
4083  013002  005002                            CLR     R2              ;CLEAR 'WORDS TO COMPARE' COUNT IN R2
```

```
4084  013004  112737  000001  001300          MOVB    #1,FRSTER       ;SET 'NOT FIRST ERROR' INDICATOR
4085  013012  112737  177777  001301          MOVB    #-1,FRSTER+1    ;SET ERROR FOUND INDICATOR
4086  013020  013737  001414  001310          MOV     CMPLMT,LIMIT    ;RESET THE COMPARE ERROR TYPEOUT LIMIT
4087  013026  000207                          RTS     PC              ;RETURN
4088
4089                                  ;CHECK ERROR BITS IN THE RH11 & RP04/5/6 REGISTERS
4090
4091  013030  032760  060000  000234  CKERR:  BIT     #60000,$RPCS1(R0)  ;SEE IF 'TRE' OR 'MCPE' SET
4092  013036  001015                          BNE     1$              ;BR IT EITHER SET
4093  013040  032760  177400  000244          BIT     #177400,$RPCS2(R0)  ;SEE IF ERROR BITS IN CS2 SET
4094  013046  001011                          BNE     1$              ;BR IF ANY SET
4095  013050  005760  000250                  TST     $RPER1(R0)      ;ANY BITS SET IN ER1
4096  013054  001006                          BNE     1$              ;BR IF ANY SET
4097  013056  005760  000274                  TST     $RPER2(R0)      ;ANY BITS SET IN ER2 ?
4098  013062  001003                          BNE     1$              ;BR IF ANY SET
4099  013064  005760  000276                  TST     $RPER3(R0)      ;ANY BITS SET IN ER3 ?
4100  013070  001416                          BEQ     2$              ;BR IF NONE SET
4101  013072  004737  020240          1$:     JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
4102  013076  104414  047117                  DISPLY  ,EM44           ;ERROR BITS SET, BUT 'SC' OR 'TRE' NOT SET
4103  013102  004737  020304                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
4104  013106  004737  020712                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
4105  013112  004737  021366                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
4106  013116  004737  023464                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
4107  013122  004737  022022                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
4108  013126  000207                  2$:     RTS     PC              ;RETURN
4109
4110                                  ;CHECK BUS ADDRESS REGISTER & WORD COUNT REGISTER
4111
4112  013130  005760  000236          CKBUS:  TST     $RPWC(R0)       ;CHECK WORD COUNT
4113  013134  001010                          BNE     1$              ;BR IF NOT ZERO
4114  013136  016046  000020                  MOV     $WRDL(R0),-(SP) ;WORD LENGTH
4115  013142  006316                          ASL     (SP)            ;CHANGE INTO BYTE COUNT
4116  013144  066016  000006                  ADD     $BUF(R0),(SP)   ;ADD THE STARTING LOCATION
4117  013150  022660  000240                  CMP     (SP)+,$RPBA(R0) ;BUFFER ADDRESS PROPER ?
4118  013154  001416                          BEQ     2$              ;BR IF OK
4119  013156  004737  020240          1$:     JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
4120  013162  104414  046725                  DISPLY  ,EM41           ;BUS ADDRESS OR WORD COUNT INCORRECT
4121  013166  004737  020304                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
4122  013172  004737  020750                  JSR     PC,LINE3D       ;PRINT LINE 3D OF ERROR MESSAGE
4123  013176  004737  021366                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
4124  013202  004737  023464                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
4125  013206  004737  022022                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
4126  013212  000207                  2$:     RTS     PC
4127
4128                                  ;COMPARE THE BUFFER
4129
4130  013214  132760  000004  000024  CMPAR:  BITB    #BIT02,$CODE(R0)  ;SEE IF READ ORDER
4131  013222  001001                          BNE     1$              ;BR IF IT IS
4132  013224  000207                          RTS     PC              ;RETURN
4133  013226  005037  001300          1$:     CLR     FRSTER          ;CLEAR 'FIRST ERROR' INDICATOR
4134  013232  032777  000002  165700  CMPARD: BIT     #SW01,@SWR      ;IS SWITCH 1 SET?
4135  013240  001401                          BEQ     1$              ;BR IF NOT
4136  013242  000207                          RTS     PC              ;YES, DON'T COMPARE
4137  013244  005037  001306          1$:     CLR     ERCTR           ;CLEAR THE ERROR COUNTER
4138  013250  016001  000006                  MOV     $BUF(R0),R1     ;BUFFER ADDRESS
4139  013254  016037  000020  001312          MOV     $WRDL(R0),CMCNT  ;WORD COUNT TO WORKING LOCATION
```

```
4140  013262  066037  000236  001312          ADD     $RPWC(R0),CMCNT  ;CALCULATE ACTUAL WORDS TRANSFERED
4141  013270  016037  000012  001314          MOV     $CYL(R0),CMCYL   ;CYLINDER ADDRESS WORKING LOCATION
4142  013276  052737  010000  001314          BIS     #BIT12,CMCYL     ;SET FORMAT BIT
4143  013304  016037  000010  001316          MOV     $SEC(R0),CMSEC   ;SECTOR & TRACK ADDRESSES TO WORKING LOCNS
4144  013312  013737  001414  001310          MOV     CMPLMT,LIMIT     ;DISPLAY LIMIT
4145  013320  005237  001310                  INC     LIMIT            ;CONVERT PARAMETER INTO LIMIT VALUE
4146  013324  012737  177777  001276  CMSTR:  MOV     #-1,ZROIND       ;CLEAR THE 'ZERO'S' INDICATOR
4147  013332  005037  001302                  CLR     SAVER1           ;CLEAR THE R1 SAVE WORD
4148  013336  005037  001304                  CLR     SAVER5           ;CLEAR THE R5 SAVE WORD
4149  013342  023760  001312  000022          CMP     CMCNT,$SSEC(R0)  ;IS BUFFER SIZE GREATER THAN ONE SECTOR ?
4150  013350  101003                          BHI     1$               ;BR IF IT IS
4151  013352  013702  001312                  MOV     CMCNT,R2         ;LESS THAN, USE REMAINING BUFFER
4152  013356  000402                          BR      2$               ;
4153  013360  016002  000002          1$:     MOV     $SSEC(R0),R2     ;COMPARE SECTOR
4154  013364  166037  000022  001312  2$:     SUB     $SSEC(R0),CMCNT  ;DECREMENT WORD COUNT
4155  013372  126027  000024  000005          CMPB    $CODE(R0),#5     ;READ HEADER & DATA?
4156  013400  001036                          BNE     CMDAT            ;BR IF NOT
4157  013402  023721  001314          CMHED:  CMP     CMCYL,(R1)+      ;CHECK CYLINDER
4158  013406  001402                          BEQ     1$               ;BR IF COMPARE OK
4159  013410  004737  013464                  JSR     PC,5$            ;REPORT ERROR
4160  013414  023721  001316          1$:     CMP     CMSEC,(R1)+      ;COMPARE SECTOR & TRACK
4161  013420  001402                          BEQ     2$               ;BR IF EQ
4162  013422  004737  013464                  JSR     PC,5$            ;REPORT ERROR
4163  013426  005721                  2$:     TST     (R1)+            ;1ST KEY WORD ZERO?
4164  013430  001402                          BEQ     3$               ;BR IF IT IS
4165  013432  004737  013464                  JSR     PC,5$            ;REPORT ERROR
4166  013436  005721                  3$:     TST     (R1)+            ;CHECK 2ND KEY WORD
4167  013440  001402                          BEQ     4$               ;BR IF ZERO
4168  013442  004737  013464                  JSR     PC,5$            ;REPORT THE ERROR
4169  013446  162702  000004          4$:     SUB     #4,R2            ;SUBTRACT HEADER LENGTH FROM SIZE
4170  013452  003530                          BLE     CMPRX            ;BR IF FINISHED
4171  013454  022702  000004                  CMP     #4,R2            ;SEE IF AT LEAST 4 MORE WORDS TO CHECK
4172  013460  101125                          BHI     CMPRX            ;BR IF NOT
4173  013462  000405                          BR      CMDAT            ;COMPARE THE DATA PORTION
4174  013464  005237  001306          5$:     INC     ERCTR            ;INCREMENT THE ERROR COUNT
4175  013470  004737  013742                  JSR     PC,CMPRT         ;REPORT THE COMPARISON ERROR
4176  013474  000207                          RTS     PC               ;CHECK THE REST OF THE HEADER
4177  013476  004737  014264          CMDAT:  JSR     PC,MATCH         ;FIND THE PATTERN
4178  013502  000403                          BR      2$               ;FOUND A PATTERN
4179  013504  004737  012606                  JSR     PC,NOMTCH        ;RETURN HERE IF NO MATCH WITH PATTERN MADE
4180  013510  000456                          BR      8$               ;BYPASS COMPARE ROUTINE
4181  013512  011405                  2$:     MOV     (R4),R5          ;ADDRESS OF PATTERN ADDRESS IN R4
4182  013514  012703  000020                  MOV     #20,R3           ;R3 IS PATTERN POS COUNTER
4183  013520  022125                  3$:     CMP     (R1)+,(R5)+      ;COMPARE BUFFER WITH PATTERN
4184  013522  001016                          BNE     5$               ;BR IF NOT EQUAL
4185  013524  005737  001306                  TST     ERCTR            ;ERRORS DETECTED ?
4186  013530  001406                          BEQ     4$               ;BR IF NO ERRORS
4187  013532  032777  000010  165400          BIT     #SW3,@SWR        ;SWITCH 3 SET ?
4188  013540  001402                          BEQ     4$               ;BR IF NOT SET
4189  013542  004737  013742                  JSR     PC,CMPRT         ;DISPLAY THE WORD
4190  013546  005302                  4$:     DEC     R2               ;DECREMENT SIZE COUNT
4191  013550  001436                          BEQ     8$               ;BR WHEN AT END
4192  013552  005303                          DEC     R3               ;DECREMENT PATT POS COUNT
4193  013554  001361                          BNE     3$               ;BR IF NOT AT END OF PATT
4194  013556  000755                          BR      2$               ;RESTART THE PATTERN
4195  013560  005761  177776          5$:     TST     -2(R1)           ;IS MISCOMPARED CHARACTER=0
```

```
4196  013564  001410                          BEQ     6$              ;BR IF YES
4197  013566  012737  177777  001276          MOV     #-1,ZROIND      ;SET NON-ZERO MISCOMPARED INDCATOR
4198  013574  005237  001306                  INC     ERCTR           ;INCREMENT THE ERROR COUNTER
4199  013600  004737  013742                  JSR     PC,CMPRT        ;REPORT ERROR
4200  013604  000760                          BR      4$              ;CONTINUE COMPARE
4201  013606  105737  001300          6$:     TSTB    FRSTER          ;FIRST ERROR?
4202  013612  100407                          BMI     7$              ;BR IF NOT
4203  013614  005037  001276                  CLR     ZROIND          ;SET THE ZERO INDICATOR
4204  013620  010137  001302                  MOV     R1,SAVER1       ;SAVE CURRENT R1
4205  013624  010537  001304                  MOV     R5,SAVER5       ;SAVE CURRENT R5
4206  013630  000746                          BR      4$              ;CONTINUE COMPARE
4207  013632  005737  001276          7$:     TST     ZROIND          ;ANY MISCOMPARIONS NOT ZEROS ?
4208  013636  001743                          BEQ     4$              ;BR IF NONE-ALL ERRORS=ZERO
4209  013640  004737  013742                  JSR     PC,CMPRT        ;REPORT ERROR
4210  013644  000740                          BR      4$              ;CONTINUE COMPARING
4211  013646  005737  001312          8$:     TST     CMCNT           ;AT END OF BUFFER
4212  013652  003430                          BLE     CMPRX           ;BR IF AT END
4213  013654  126027  000024  000005          CMPB    $CODE(R0),#5    ;SEE IF READ HEADER & DATA
4214  013662  001220                          BNE     CMSTR           ;BR IF NOT
4215  013664  105237  001316                  INCB    CMSEC           ;INCREMENT SECTOR
4216  013670  123727  001316  000026          CMPB    CMSEC,#22.      ;SECTOR GREATER THAN MAX ?
4217  013676  103612                          BLO     CMSTR           ;BR IF NOT GREATER THAN MAX
4218  013700  105037  001316                  CLRB    CMSEC           ;CLEAR SECTOR ADDRESS
4219  013704  105237  001317                  INCB    CMTRK           ;INCREMENT TRACK
4220  013710  123727  001317  000023          CMPB    CMTRK,#19.      ;TRACK GREATER THAN MAX ?
4221  013716  103602                          BLO     CMSTR           ;BR IF NOT GREATER
4222  013720  105037  001317                  CLRB    CMTRK           ;RESET TRACK ADDRESS
4223  013724  005237  001314                  INC     CMCYL           ;INCREMENT CYLINDER ADDRESS
4224  013730  000137  013324                  JMP     CMSTR           ;CONTINUE WITH COMPARE
4225  013734  004737  014216          CMPRX:  JSR     PC,ENDCMP       ;PRINT LAST LINE IF ERRORS
4226  013740  000207                          RTS     PC
4227
4228                                  ;TYPE DATA COMPARE ERRORS
4229
4230  013742  005737  001302          CMPRT:  TST     SAVER1          ;PRINT SAVED VALUES ?
4231  013746  001010                          BNE     2$              ;BR IF NOT
4232  013750  105737  001300                  TSTB    FRSTER          ;FIRST ERROR?
4233  013754  100402                          BMI     1$              ;BR IF NOT
4234  013756  004737  014036                  JSR     PC,4$           ;PRINT INITIAL MESSAGE INFO
4235  013762  004737  014120          1$:     JSR     PC,8$           ;PRINT REMAINDER OF MESSAGE
4236  013766  000422                          BR      3$              ;EXIT
4237  013770                          2$:
4238  013770  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
4239  013772  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
4240  013774  013701  001302                  MOV     SAVER1,R1       ;DISPLAY SAVED R1
4241  014000  013705  001304                  MOV     SAVER5,R5       ;DISPLAY SAVED R5
4242  014004  004737  014036                  JSR     PC,4$           ;PRINT INITIAL MESSAGE INFO
4243  014010  004737  014120                  JSR     PC,8$           ;PRINT SAVED VALUES
4244  014014  005037  001302                  CLR     SAVER1          ;CLEAR SAVED REGISTER INDICATORS
4245  014020  005037  001304                  CLR     SAVER5          ;CLEAR THE OTHER ONE
4246  014024  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
4247  014026  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
4248  014030  004737  014120                  JSR     PC,8$           ;PRINT REMAINDER OF MESSAGE
4249  014034  000207                  3$:     RTS     PC              ;RETURN
4250  014036  105737  001300          4$:     TSTB    FRSTER          ;FIRST ERROR ?
4251  014042  100425                          BMI     7$              ;BR IF NOT
```

```
4252  014044  001013                        BNE     5$              ;BR IF FIRST ERROR AND PROCESSING 'DCK' ERROR
4253  014046  004737  020240                JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
4254  014052  104414  046771                DISPLY  .EM42           ;DATA COMPARE ERROR
4255  014056  004737  020304                JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
4256  014062  004737  020720                JSR     PC,LINE3A       ;PRINT LINE 3A OF ERROR MESSAGE
4257  014066  004737  021366                JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
4258  014072  000404                        BR      6$              ;GO TO TYPE HEADER
4259  014074  104414  051425          5$:   DISPLY  .LIN9B          ;HEADER MESSAGE OF PROCESSING 'DCK' ERROR
4260  014100  104414  001165                DISPLY  .$CRLF          ;CR-LF
4261  014104  104414  051454          6$:   DISPLY  .LIN9H          ;DISPLAY HEADER
4262  014110  012737  177777  001300        MOV     #-1,FRSTER      ;SET ERROR FLAG
4263  014116  000207                  7$:   RTS     PC              ;RETURN
4264  014120  005737  001310          8$:   TST     LIMIT           ;TYPEOUT LIMIT REACHED ?
4265  014124  001403                        BEQ     9$              ;BR IF IT HAS
4266  014126  005337  001310                DEC     LIMIT           ;DECREMENT LIMIT COUNTER
4267  014132  001005                        BNE     10$             ;BR IF NOT AT LIMIT
4268  014134  032777  000200  164776  9$:   BIT     #SW07,@SWR      ;PRINT ALL DATA COMPARE ERRORS ?
4269  014142  001001                        BNE     10$             ;BR IF YES
4270  014144  000207                        RTS     PC              ;RETURN
4271  014146  010146                 10$:   MOV     R1,-(SP)        ;BUFFER ADDRESS
4272  014150  162716  000002                SUB     #2,(SP)         ;ADJUST ADDRESS
4273  014154  004737  022302                JSR     PC,LINOCT       ;TYPE IT
4274  014160  104414  052215                DISPLY  .LINSP          ;2 SPACES
4275  014164  016546  177776                MOV     -2(R5),-(SP)    ;PUT GOOD DATA ON THE STACK
4276  014170  004737  022302                JSR     PC,LINOCT       ;TYPE IT
4277  014174  104414  052215                DISPLY  .LINSP          ;2 SPACES
4278  014200  016146  177776                MOV     -2(R1),-(SP)    ;BAD DATA
4279  014204  004737  022302                JSR     PC,LINOCT       ;TYPE IT
4280  014210  104414  001165                DISPLY  .$CRLF          ;CR-LF
4281  014214  000207                        RTS     PC              ;RETURN
4282
4283                                  ;LAST LINE OF COMPARE ERROR REPORTING
4284
4285  014216  105737  001301          ENDCMP: TSTB    FRSTER+1        ;ANY COMPARE ERRORS FOUND ?
4286  014222  001417                          BEQ     2$              ;BR IF NOT
4287  014224  005737  001306                  TST     ERCTR           ;SEE HOW MANY ERRORS
4288  014230  001410                          BEQ     1$              ;BR IF ONLY CAN'T MATCH PATTERN
4289  014232  104414  051552                  DISPLY  .LIN9E          ;'NUMBER OF ERRORS='
4290  014236  013746  001306                  MOV     ERCTR,-(SP)     ;NUMBER OF ERRORS
4291  014242  004737  022334                  JSR     PC,LINDEC       ;TYPE IT
4292  014246  104414  001165                  DISPLY  .$CRLF          ;CR-LF
4293  014252  004737  023464          1$:     JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
4294  014256  004737  022022                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
4295  014262  000207                  2$:     RTS     PC              ;RETURN
4296
4297
4298                                  ;ROUTINE TO MATCH THE DATA WITH A PATTERN
4299                                  ;CALL:
4300                                  ;       MOV     #BUFFER,R1      ;BUFFER ADDRESS
4301                                  ;       JSR     PC,MATCH
4302                                  ;       RETURN1                 ;PATTERN ADDRESS IN R4
4303                                  ;       RETURN2                 ;COULDN'T MATCH PATTERN
4304
4305  014264  010146                  MATCH:  MOV     R1,-(SP)        ;SAVE R1 ON THE STACK
4306  014266  012704  000044                  MOV     #44,R4          ;PATTERN TABLE INDEX
4307  014272  011601                  1$:     MOV     (SP),R1         ;RELOAD R1
```

```
4308  014274  162704  000002                  SUB     #2,R4           ;DECREMENT INDEX
4309  014300  016405  002762                  MOV     STNDAT(R4),R5   ;ADDRESS OF PATTERN ADDRESS
4310  014304  001411                          BEQ     3$              ;BR IF ALL PATTERNS CHECKED AND NO MATCH
4311                                                                  ;FOUND
4312  014306  012703  000004                  MOV     #4,R3           ;NUMBER OF LOCATIONS TO CHECK
4313  014312  022125                  2$:     CMP     (R1)+,(R5)+     ;COMPARE THE BUFFER AGAINST THE PATTERN
4314  014314  001366                          BNE     1$              ;BR IF NOT EQUAL, TRY NEXT PATTERN
4315  014316  005303                          DEC     R3              ;FINISHED CHECKING?
4316  014320  001374                          BNE     2$              ;BR IF NOT FINISHED
4317  014322  062704  002762                  ADD     #STNDAT,R4      ;MAKE PATTERN ADDRESS ABSOLUTE
4318  014326  000403                          BR      4$              ;EXIT
4319  014330  062766  000002  000002  3$:     ADD     #2,2(SP)        ;INCREMENT RETURN ADDRESS
4320  014336  012601                  4$:     MOV     (SP)+,R1        ;RESTORE R1
4321  014340  000207                          RTS     PC              ;RETURN
4322
4323                                  ;USE ECC TO CORRECT THE DATA ERROR
4324
4325  014342  016037  000240  001322  ECC:    MOV     $RPBA(R0),ECSEC ;ADDRESS OF LAST LOCN XFERED
4326  014350  016046  000236                  MOV     $RPWC(R0),-(SP) ;ACT WORDS XFERED (2'S COMP)
4327  014354  066016  000020                  ADD     $WROL(R0),(SP)  ;ADD WORDS REQUESTED
4328  014360  005046                          CLR     -(SP)           ;CLEAR NEXT STACK LOCN
4329  014362  016046  000022                  MOV     $SSEC(R0),-(SP) ;SECTOR SIZE
4330  014366  004737  027114                  JSR     PC,LINKDV       ;DIVIDE WORDS XFERED BY SECTOR SIZE
4331  014372  005716                          TST     (SP)            ;PARTIAL SECTOR XFERED ?
4332  014374  001413                          BEQ     1$              ;BR IF NOT
4333  014376  006316                          ASL     (SP)            ;CONVERT INTO NUMBER OF BYTES
4334  014400  161637  001322                  SUB     (SP),ECSEC      ;SUBTRACT SECTOR RESIDUE
4335  014404  126027  000024  000005          CMPB    $CODE(R0),#5    ;WAS OP READ HEAD & DATA
4336  014412  001007                          BNE     2$              ;BR IF NOT
4337  014414  062737  000010  001322          ADD     #8.,ECSEC       ;ADD HEADER SIZE (IN BYTES) BACK IN
4338  014422  000403                          BR      2$              ;GO ADJUST THE STACK POINTER
4339  014424  162737  001000  001322  1$:     SUB     #1000,ECSEC     ;SUBTRACT SECTOR DATA FIELD SIZE (IN BYTES)
4340  014432  062706  000004          2$:     ADD     #4,SP           ;ADJUST THE STACK POINTER
4341  014436  016037  000300  001320          MOV     $RPEC1(R0),ECBIT  ;ECC POSITION COUNT
4342  014444  005337  001320                  DEC     ECBIT           ;ADJUST THE POSITION COUNT
4343  014450  013737  001320  001330          MOV     ECBIT,ECWRD     ;LOAD THE WORD COUNT LOCATION
4344  014456  042737  177760  001320          BIC     #^C17,ECBIT     ;SAVE THE BIT OFFSET COUNT
4345  014464  042737  000017  001330          BIC     #17,ECWRD       ;CLEAR THE BIT OFFSET
4346  014472  006237  001330                  ASR     ECWRD           ;CHANGE TO BYTE COUNT
4347  014476  006237  001330                  ASR     ECWRD           ;CHANGE TO BYTE COUNT
4348  014502  006237  001330                  ASR     ECWRD           ;CHANGE TO BYTE COUNT
4349  014506  104414  051631                  DISPLY  ,LIN10A         ;'ERROR BURST BEGINS AT '
4350  014512  013746  001330                  MOV     ECWRD,-(SP)     ;PUT THE WORD COUNT ON THE STACK
4351  014516  006216                          ASR     (SP)            ;CONVERT TO WORD COUNT FOR MESSAGE
4352  014520  004737  030030                  JSR     PC,$S82D        ;CONVERT THE WORD COUNT
4353  014524  004737  027430                  JSR     PC,$SUPRS       ;PRINT IT
4354  014530  104414  051665                  DISPLY  ,LIN10B         ;' IN DATA FIELD OF ERROR SECTOR'
4355  014534  063737  001322  001330          ADD     ECSEC,ECWRD     ;FIND THE BEGINNING OF THE ERROR BURST
4356  014542  026037  000240  001330          CMP     $RPBA(R0),ECWRD  ;SEE IF BURST WAS IN DATA READ
4357  014550  101002                          BHI     .+6             ;BR IF IN DATA READ
4358  014552  000137  015070                  JMP     ECC2            ;NOT IN DATA READ - REPORT IT
4359  014556  016037  000302  001324          MOV     $RPEC2(R0),ECMSK0  ;GET THE ERROR MASK
4360  014564  005037  001326                  CLR     ECMSK1          ;CLEAR THE UPPER MASK WORD
4361  014570  005737  001320          3$:     TST     ECBIT           ;BIT OFFSET EQUAL ZERO
4362  014574  001407                          BEQ     4$              ;BR IF IT IS
4363  014576  005337  001320                  DEC     ECBIT           ;DECREMENT THE BIT OFFSET COUNT
```

```
4364  014602  006337  001324                  ASL     ECMSK0          ;SHIFT THE ERROR MASK
4365  014606  006137  001326                  ROL     ECMSK1          ;SHIFT THE LOWER INTO THE UPPER
4366  014612  000766                          BR      3$              ;CONTINUE THE SHIFT
4367  014614  017737  164510  001334  4$:     MOV     @ECWRD,ECBAD0   ;SAVE THE INCORRECT WORD
4368  014622  005037  001336                  CLR     ECWRD1          ;CLEAR SECOND INCORRECT WORD ADDRESS
4369  014626  013746  001324                  MOV     ECMSK0,-(SP)    ;PUT LOWER MASK ON STACK
4370  014632  047716  164472                  BIC     @ECWRD,(SP)     ;CLEAR ERRONEOUS ONE BITS FROM MASK
4371  014636  043777  001324  164464          BIC     ECMSK0,@ECWRD   ;CLEAR ERRONEOUS ONE BITS FROM BAD WORD
4372  014644  052677  164460                  BIS     (SP)+,@ECWRD    ;SET DROPPED BITS
4373  014650  005737  001326                  TST     ECMSK1          ;DOES BURST GO INTO NEXT WORD ?
4374  014654  001431                          BEQ     ECC1            ;BR IF BURST ONLY IN ONE WORD
4375  014656  013737  001330  001336          MOV     ECWRD,ECWRD1    ;DUPLICATE ADDRESS
4376  014664  062737  000002  001336          ADD     #2,ECWRD1       ;INCREMENT ERROR ADDRESS
4377  014672  026037  000240  001336          CMP     $RPBA(R0),ECWRD1  ;IS NEXT WORD IN THE BUFFER
4378  014700  101003                          BHI     5$              ;BR IF IT IS
4379  014702  005037  001336                  CLR     ECWRD1          ;CLEAR 2ND WORD ADDRESS
4380  014706  000414                          BR      ECC1            ;PRINT WORD CORRECTED
4381  014710  017737  164422  001342  5$:     MOV     @ECWRD1,ECBAD1  ;SAVE THE SECOND BAD WORD
4382  014716  013746  001326                  MOV     ECMSK1,-(SP)    ;PUT THE UPPER MASK ON THE STACK
4383  014722  047716  164410                  BIC     @ECWRD1,(SP)    ;CLEAR ERRONEOUS ONE BITS FROM UPPER MASK
4384  014726  043777  001326  164402          BIC     ECMSK1,@ECWRD1  ;CLEAR ERRONEOUS ONE BITS FROM DATA WORD
4385  014734  052677  164376                  BIS     (SP)+,@ECWRD1   ;SET DROPPED BITS
4386  014740  104414  052033          ECC1:   DISPLY  ,LIN10H         ;HEADER
4387  014744  013746  001330                  MOV     ECWRD,-(SP)     ;PUT ECWRD ON THE STACK
4388  014750  004737  022302                  JSR     PC,LINOCT       ;TYPE ECWRD
4389  014754  104414  052215                  DISPLY  ,LINSP          ;SPACES
4390  014760  013746  001334                  MOV     ECBAD0,-(SP)    ;PUT ECBAD0 ON THE STACK
4391  014764  004737  022302                  JSR     PC,LINOCT       ;TYPE ECBAD0
4392  014770  104414  052215                  DISPLY  ,LINSP          ;SPACES
4393  014774  017746  164330                  MOV     @ECWRD,-(SP)    ;PUT @ECWRD ON THE STACK
4394  015000  004737  022302                  JSR     PC,LINOCT       ;TYPE @ECWRD
4395  015004  104414  052215                  DISPLY  ,LINSP          ;SPACES
4396  015010  005737  001336                  TST     ECWRD1          ;PRINT THE NEXT WORD ?
4397  015014  001427                          BEQ     ECCX            ;BR IF NOT
4398  015016  104414  001165                  DISPLY  ,$CRLF          ;CR-LF
4399  015022  013746  001336                  MOV     ECWRD1,-(SP)    ;PUT ECWRD1 ON THE STACK
4400  015026  004737  022302                  JSR     PC,LINOCT       ;TYPE ECWRD1
4401  015032  104414  052215                  DISPLY  ,LINSP          ;SPACES
4402  015036  013746  001342                  MOV     ECBAD1,-(SP)    ;PUT ECBAD1 ON THE STACK
4403  015042  004737  022302                  JSR     PC,LINOCT       ;TYPE ECBAD1
4404  015046  104414  052215                  DISPLY  ,LINSP          ;SPACES
4405  015052  017746  164260                  MOV     @ECWRD1,-(SP)   ;PUT @ECWRD1 ON THE STACK
4406  015056  004737  022302                  JSR     PC,LINOCT       ;TYPE @ECWRD1
4407  015062  104414  052215                  DISPLY  ,LINSP          ;SPACES
4408  015066  000402                          BR      ECCX            ;EXIT
4409  015070  104414  051726          ECC2:   DISPLY  ,LIN10C         ;ERROR BURST WAS NOT TRANSFERED TO MEMORY
4410  015074  104414  001165          ECCX:   DISPLY  ,$CRLF          ;CR-LF
4411  015100  000207                          RTS     PC              ;RETURN
4412
4413                                  ;ROUTINE TO DISPLAY THE SECTOR WHICH GAVE THE HARD ERROR
4414
4415  015102  032777  000010  164030  PRTBAD: BIT     #SW3,@SWR       ;PRINT THE BAD SECTOR ?
4416  015110  001460                          BEQ     6$              ;BR IF NOT
4417  015112  016001  000240                  MOV     $RPBA(R0),R1    ;PUT THE END ADDRESS INTO R1
4418  015116  016046  000020                  MOV     $WRDL(R0),-(SP) ;FIND THE BEGINNING OF THE SECTOR
4419  015122  066016  000236                  ADD     $RPWC(R0),(SP)  ;SUBTRACT THE WORDS NOT TRANSFERED
```

```
4420  015126  005046                          CLR     -(SP)           ;MAKE THE UPPER DIVIDEND 0
4421  015130  016046  000022                  MOV     $SSEC(R0),-(SP) ;DIVDE THE WORDS TRANSFERED BY THE SECTOR SIZE
4422  015134  004737  027114                  JSR     PC,LINKDV       ;DIVIDE
4423  015140  005716                          TST     (SP)            ;REMAINDER = 0 ?
4424  015142  001403                          BEQ     1$              ;BR IF IT IS - COMPLETE SECTOR TRANSFERED
4425  015144  006316                          ASL     (SP)            ;CONVERT THE RESIDUAL SECTOR SIZE INTO BYTE COUNT
4426  015146  161601                          SUB     (SP),R1         ;SUBTRACT IT FROM THE END ADDRESS
4427  015150  000410                          BR      2$              ;FINISH THE SIZING
4428  015152  162701  001000          1$:     SUB     #1000,R1        ;SUBTRACT FULL SECTOR SIZE FROM END ADDR
4429  015156  126027  000024  000005          CMPB    $CODE(R0),#5    ;WAS OPERATION READ HEADER & DATA ?
4430  015164  001002                          BNE     2$              ;BR IF NOT
4431  015166  162701  000010                  SUB     #10,R1          ;SUBTRACK HEADER SIZE FROM ADDR
4432  015172  062706  000004          2$:     ADD     #4,SP           ;RESTORE THE STACK POINTER
4433  015176  104414  052120                  DISPLY  ,LIN11H         ;PRINT THE HEADER
4434  015202  012702  000007          3$:     MOV     #7,R2           ;R2 CONTAINS THE WORDS/LINE COUNT
4435  015206  010146                          MOV     R1,-(SP)        ;PUT THE ADDRESS ON THE STACK
4436  015210  004737  022302                  JSR     PC,LINOCT       ;TYPE THE ADDRESS
4437  015214  020160  000240          4$:     CMP     R1,$RPBA(R0)    ;PRINTED ALL THE SECTOR ?
4438  015220  001412                          BEQ     5$              ;BR IF ALL PRINTED
4439  015222  104414  052215                  DISPLY  ,LINSP          ;SPACES
4440  015226  012146                          MOV     (R1)+,-(SP)     ;PUT THE DATA ON THE STACK
4441  015230  004737  022302                  JSR     PC,LINOCT       ;TYPE THE DATA
4442  015234  005302                          DEC     R2              ;DECREMENT THE HORIZONTAL COUNT
4443  015236  001366                          BNE     4$              ;BR IF NOT AT THE END OF THE LINE
4444  015240  104414  001165                  DISPLY  ,$CRLF          ;CR-LF
4445  015244  000756                          BR      3$              ;RESTORE THE WORDS/LINE COUNT
4446  015246  104414  001165          5$:     DISPLY  ,$CRLF          ;PRINT WHAT REMAINS IN THE BUFFER
4447  015252  000207                  6$:     RTS     PC              ;RETURN
4448
4449                                  ;ROUTINE TO DO AN RTC - DRIVE SELECTED IN R0
4450                                  ;CALL:
4451                                  ;       MOV     #DPB,R0         ;DPB ADDRESS
4452                                  ;       JSR     PC,RTNCTR
4453                                  ;       RETURN
4454
4455  015254  111037  044770          RTNCTR: MOVB    (R0),GENDPB     ;MOVE THE DRIVE # TO THE GENERAL DPB
4456  015260  112737  000117  044772          MOVB    #RTC,GENDPB+$COMND  ;COMMAND CODE
4457  015266  004037  034160          1$:     JSR     R0,RP04         ;DRIVER ENTRANCE
4458  015272  044770                          GENDPB                  ;DPB ADDRESS FOR ORDER
4459  015274  000774                          BR      1$              ;DRIVER DIDN'T ACCEPT ORDER
4460  015276  000207                          RTS     PC              ;RETURN
4461
4462                                  ;ROUTINE TO DO A RECALIBRATE - DRIVE SELECTED IN R0
4463                                  ;CALL:
4464                                  ;       MOV     #DPB,R0         ;DPB ADDRESS
4465                                  ;       JSR     PC,RECALT
4466                                  ;       RETURN
4467                                  ;
4468                                  ;OR
4469                                  ;
4470                                  ;       MOV     #DPB,R0         ;DPB ADDRESS
4471                                  ;       MOVB    #DRIVE,GENDPB   ;DRIVE ADDRESS
4472                                  ;       JSR     PC,RECALTO
4473                                  ;       RETURN
4474
4475  015300  111037  044770          RECALT: MOVB    (R0),GENDPB     ;MOVE THE DRIVE # TO THE GENERAL DPB
```

```
4476  015304  112737  000107  044772  RECAL0: MOVB    #RECAL,GENDPB+$COMND  ;RELCALIBRATE COMMAND
4477  015312  004037  034160          1$:     JSR     R0,RP04               ;DRIVER ENTRANCE
4478  015316  044770                          GENDPB                        ;DPB ADDRESS FOR ORDER
4479  015320  000774                          BR      1$                    ;DRIVER DIDN'T ACCEPT THE ORDER
4480  015322  005737  045006          2$:     TST     GENDPB+$TATUS                 ;SEE IF FINISHED
4481  015326  001775                          BEQ     2$                    ;BR IF NOT FINISHED
4482  015330  000207                          RTS     PC                    ;RETURN
4483
4484                                  ;OFFSET THE DRIVE IN R0 (OFFSET CODE PRELOADED INTO 'RPOF')
4485                                  ;CALL:
4486                                  ;       MOVB    #OFFSET,GENDPB+$FMT  ;OFFSET CODE
4487                                  ;       MOV     #DPB,R0              ;DPB ADDRESS
4488                                  ;       JSR     PC,OFFST
4489                                  ;       RETURN
4490
4491  015332  111037  044770          OFFST:  MOVB    (R0),GENDPB           ;DRIVE # TO GENERAL DPB
4492  015336  112737  000115  044772          MOVB    #OFFSET,GENDPB+$COMND ;COMMAND
4493  015344  004037  034160          1$:     JSR     R0,RP04               ;DRIVER ENTRANCE
4494  015350  044770                          GENDPB                        ;DPB ADDRESS FOR ORDER
4495  015352  000774                          BR      1$                    ;DRIVER DIDN'T ACCEPT ORDER
4496  015354  000207                          RTS     PC
4497
4498                                  ;UTILITY READ HEADER ROUTINE
4499                                  ;CALL:
4500                                  ;       MOV     #DPB,R0              ;DPB ADDRESS
4501                                  ;       MOV     #SECTOR,-(SP)        ;SECTOR ADDRESS
4502                                  ;       MOV     #TRACK,-(SP)         ;TRACK ADDRESS
4503                                  ;       JSR     PC,READDR
4504                                  ;       RETURN
4505
4506  015356  116637  000002  045001  READHD: MOVB    2(SP),GENDPB+$TRK ;TRACK ADDRESS
4507  015364  116637  000004  045000          MOVB    4(SP),GENDPB+$SEC ;SECTOR ADDRESS
4508  015372  111037  044770                  MOVB    (R0),GENDPB       ;DRIVE NUMBER
4509  015376  016037  000272  045002          MOV     $RPCC(R0),GENDPB+$CYL ;CYLINDER ADDRESS
4510  015404  112737  000173  044772          MOVB    #RDHD,GENDPB+$COMND ;COMMAND
4511  015412  004037  034160          1$:     JSR     R0,RP04               ;DRIVER ENTRANCE
4512  015416  044770                          GENDPB                        ;DPB ADDRESS FOR ORDER
4513  015420  000774                          BR      1$                    ;DRIVER DIDN'T ACCEPT COMMAND
4514  015422  005737  045006          2$:     TST     GENDPB+$TATUS         ;FINISHED?
4515  015426  001775                          BEQ     2$                    ;BR IF NOT
4516  015430  012666  000002                  MOV     (SP)+,2(SP)           ;ADJUST STACK FOR RETURN
4517  015434  005726                          TST     (SP)+
4518  015436  000207                          RTS     PC                    ;RETURN
4519
4520                                  ;RETRY THE PRESENT OPERATION
4521                                  ;CALL:
4522                                  ;       MOV     #COUNT,RETRY         ;RETRY COUNT
4523                                  ;       JSR     PC,$RETRY
4524                                  ;       RETURN1                      ;RETRY UNSUCESSFUL
4525                                  ;       RETURN2                      ;SUCESSFUL RETRY
4526                                  ;                                    ;NOTE: IF A DIFFERENT ERROR OCCURS DURING
4527                                  ;                                    ;RETRY, THE ROUTINE EXITS TO 'ERPRC1'
4528
4529  015440  004737  016430          $RETRY: JSR     PC,GODRIV             ;RE-START ORDER
4530  015444  005760  000016          1$:     TST     $TATUS(R0)            ;ORDER FINISHED?
4531  015450  001775                          BEQ     1$                    ;BR IF NOT
```

```
4532  015452  100405                          BMI     2$                      ;BR IF ERROR
4533  015454  105237  001253                  INCB    RETRY+1                 ;INCREMENT RETRY COUNT
4534  015460  062716  000002                  ADD     #2,(SP)                 ;INCREMENT RETURN
4535  015464  000425                          BR      5$                      ;GO TO EXIT
4536  015466  032760  000200  000016  2$:     BIT     #BIT7,$STATUS(R0)       ;DID ORDER TERMINATE NORMALLY ?
4537  015474  001430                          BEQ     7$                      ;BR IF NOT
4538  015476  005737  001250                  TST     MASK                    ;IS ERROR MASK 0 ?
4539  015502  001004                          BNE     3$                      ;BR IF NOT
4540  015504  005760  000250                  TST     $RPER1(R0)              ;MAKE SURE THAT THE DRIVE ERROR REG IS CLEAR
4541  015510  001014                          BNE     5$                      ;BR IF NOT
4542  015512  000404                          BR      4$                      ;CONTINUE RETRY
4543  015514  033760  001250  000250  3$:     BIT     MASK,$RPER1(R0)         ;SAME ERROR?
4544  015522  001407                          BEQ     5$                      ;BR IF NOT
4545  015524  105237  001253                4$:     INCB    RETRY+1                 ;INCREMENT RETRY COUNT
4546  015530  123737  001252  001253              CMPB    RETRY,RETRY+1           ;DONE ?
4547  015536  001340                          BNE     $RETRY                  ;BR IF NOT DONE
4548  015540  000207                        5$:     RTS     PC                      ;RETURN
4549  015542  004737  022270                6$:     JSR     PC,LINE8                ;REPORT DIFFERENT ERROR
4550  015546  004737  022022                        JSR     PC,LINE7                ;PRINT LINE 7
4551  015552  005726                          TST     (SP)+                   ;ADJUST STACK POINTER FOR DIRECT RETURN
4552  015554  000207                          RTS     PC                      ;RETURN
4553  015556  104414  051370                7$:     DISPLY  ,LIN8M                  ;'DIFFERENT ERROR DURING RETRY'
4554  015562  000137  006704                        JMP     ERPRC1                  ;REPORT THE ERROR
4555
4556                                        ;ROUTINE TO UPDATE THE PERFORMANCE SUMMARY STATISTICS
4557                                        ;CALL:
4558                                        ;       MOV     #DPB,R0         ;DPB ADDRESS
4559                                        ;       JSR     PC,STATIS
4560                                        ;       RETURN
4561
4562  015566  032760  000300  000016  STATIS: BIT     #BIT07!BIT06,$STATUS(R0) ;CHECK FOR DATA TERMINATION
4563  015574  001454                          BEQ     3$                      ;BR IF NOT DATA TERMINATION
4564  015576  016037  000240  015730          MOV     $RPBA(R0),FACTOR        ;STORE THE FINAL BUFFER ADDRESS
4565  015604  166037  000006  015730          SUB     $BUF(R0),FACTOR         ;SUBTRACT THE INITIAL ADDRESS
4566  015612  001434                          BEQ     2$                      ;BR IF NO DATA TRANSFER
4567  015614  006237  015730                  ASR     FACTOR                  ;CONVERT TO A WORD COUNT
4568  015620  063760  015730  000046          ADD     FACTOR,$TRANS(R0)       ;UPDATE WORD COUNT
4569  015626  005560  000050                  ADC     $TRANS+2(R0)            ;ADD ANY CARRY
4570  015632  132760  000002  000024          BITB    #BIT01,$CODE(R0)        ;SEE IF ORDER READ OR WRITE
4571  015640  001021                          BNE     2$                      ;BRANCH IF ORDER WRITE
4572  015642  005737  001424                  TST     AUTOCK                  ;AUTO WRITE CHECKS BEING PERFORMED
4573  015646  001411                          BEQ     1$                      ;BR IF NOT
4574  015650  126027  000024  000001          CMPB    $CODE(R0),#1            ;PRESENT OPERATION AN AUTOMATIC WRITE CHECK ?
4575  015656  101005                          BHI     1$                      ;BR IF NOT
4576  015660  066060  000020  000046          ADD     $WRDL(R0),$TRANS(R0)    ;ADD WORDS WRITTEN
4577  015666  005560  000050                  ADC     $TRANS+2(R0)            ;ADD A CARRY
4578  015672  063760  015730  000052  1$:     ADD     FACTOR,$READ(R0)        ;UPDATE THE READ WORD COUNT
4579  015700  005560  000054                  ADC     $READ+2(R0)             ;ADD ANY CARRY
4580  015704  026060  000012  000270  2$:     CMP     $CYL(R0),$RPCA(R0)      ;DID MID-TRANSFER SEEK OCCUR
4581  015712  001405                          BEQ     3$                      ;BR IF NOT
4582  015714  062760  000001  000042          ADD     #1,$POSIT(R0)           ;INCREMENT SEEK COUNT
4583  015722  005560  000044                  ADC     $POSIT+2(R0)            ;ADD CARRY TO UPPER WORD
4584  015726  000207                        3$:     RTS     PC
4585
4586  015730  000000                        FACTOR: .WORD   0                       ;USED FOR WORDS TRANSFERED
4587
```

```
                                        ;ROUTINE TO GET A BUFFER
                                        ;CALL:
                                        ;       MOV     #DPB,R0         ;DPB 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

015732  010146                  GETBUF: MOV     R1,-(SP)        ;SAVE R1
015734  010246                          MOV     R2,-(SP)        ;SAVE R2
015736  010346                          MOV     R3,-(SP)        ;SAVE R3
015740  013702  001616                  MOV     BUFTBL,R2       ;NUMBER OF SEPARATE BUFFERS
015744  001444                          BEQ     6$              ;BR IF NONE AVAILABLE
015746  012701  001620                  MOV     #BUFTBL+2,R1    ;FIRST ADDRESS OF ALLOCATION TABLE
015752  026061  000020  000002  1$:     CMP     $WRDL(R0),2(R1) ;SEE IF THERE IS A BLOCK LARGE ENOUGH
015760  101405                          BLOS    3$              ;BRANCH IF IT IS
015762  005302                          DEC     R2              ;DECREMENT TABLE COUNT
015764  001434                          BEQ     6$              ;BR IF THROUGH TABLE
015766  062701  000004                  ADD     #4,R1           ;INCREMENT TABLE POINTER
015772  000767                          BR      1$              ;CONTINUE LOOKING
015774  011166  000010          3$:     MOV     (R1),10(SP)     ;BUFFER ADDRESS TO STACK
016000  166061  000020  000002          SUB     $WRDL(R0),2(R1) ;ADJUST BUFFER SIZE
016006  001407                          BEQ     4$              ;BR IF DIFFERENCE IS ZERO
016010  006360  000020                  ASL     $WRDL(R0)       ;CONVERT # WORDS TO BYTES
016014  066011  000020                  ADD     $WRDL(R0),(R1)  ;MAKE NEW STARTING ADDRESS
016020  006260  000020                  ASR     $WRDL(R0)       ;RETURN # BYTES TO WORDS
016024  000414                          BR      6$              ;RETURN
016026  005337  001616          4$:     DEC     BUFTBL          ;DECREMENT ENTRIES COUNT
016032  001411                          BEQ     6$              ;BR IF ALLOCATION TABLE EMPTY
016034  005302                          DEC     R2              ;DECREMENT TABLE COUNT
016036  001407                          BEQ     6$              ;BR IF ITEM WERE LAST ENTRY
016040  010103                          MOV     R1,R3           ;MOVE TABLE POINTER
016042  062703  000004                  ADD     #4,R3           ;POINT TO NEXT ENTRY
016046  012321                  5$:     MOV     (R3)+,(R1)+     ;MOVE ITEMS
016050  012321                          MOV     (R3)+,(R1)+
016052  005302                          DEC     R2              ;DECREMENT TABLE COUNT
016054  001374                          BNE     5$              ;CONTINUE IF NOT AT END OF TABLE
016056  012603                  6$:     MOV     (SP)+,R3        ;RESTORE R3
016060  012602                          MOV     (SP)+,R2        ;RESTORE R2
016062  012601                          MOV     (SP)+,R1        ;RESTORE R1
016064  000207                          RTS     PC              ;RETURN


                                        ;ROUTINE TO PUT BUFFER BACK IN TABLE
                                        ;CALL:
                                        ;       MOV     #DPB,R0         ;DPB ADDRESS
                                        ;       JSR     PC,RELBUF
                                        ;       RETURN

016066  010146                  RELBUF: MOV     R1,-(SP)        ;SAVE R1
016070  012701  001620                  MOV     #BUFTBL+2,R1    ;BEGINNING OF TABLE
016074  013702  001616                  MOV     BUFTBL,R2       ;ENTRY COUNT
016100  001424                          BEQ     2$              ;BR IF EMPTY TABLE
016102  016003  000020                  MOV     $WRDL(R0),R3    ;TRIAL ADDRESS
016106  006303                          ASL     R3              ;CHANGE TO BYTE COUNT
016110  066003  000006                  ADD     $BUF(R0),R3     ;ADDRESS OF HIGHER ADJACENT BLOCK
```

```
4644  016114  021103                    1$:   CMP   (R1),R3          ;UPPER ADJACENT BLOCK
4645  016116  001424                          BEQ   4$               ;BR IF YES
4646  016120  062701  000004                 ADD   #4,R1            ;INCREMENT POINTER
4647  016124  005302                          DEC   R2               ;DECREMENT ENTRY COUNT
4648  016126  001372                          BNE   1$               ;CONTINUE SEARCHING
4649  016130  016011  000006                 MOV   $BUF(R0),(R1)    ;PUT THE BUFFER BLOCK INTO THE TABLE
4650  016134  016061  000020  000002         MOV   $WRDL(R0),2(R1)  ;BLOCK SIZE
4651  016142  005237  001616                 INC   BUFTBL           ;INCREMENT ENTRY COUNT
4652  016146  005202                          INC   R2               ;INCREMENT R2 FOR USE LATER
4653  016150  000414                          BR    5$               ;SEE IF A LOWER ADJACENT BLOCK IS IN THE TABLE
4654  016152  016021  000006           2$:   MOV   $BUF(R0),(R1)+    ;BLOCK ADDRESS TO TABLE
4655  016156  016021  000020                 MOV   $WRDL(R0),(R1)+  ;SIZE TO TABLE
4656  016162  005237  001616                 INC   BUFTBL           ;INCREMENT ENTRY COUNT
4657  016166  000443                          BR    10$              ;EXIT
4658  016170  016011  000006           4$:   MOV   $BUF(R0),(R1)    ;RELEASED BUFFER IS LOWER ADJACENT
4659  016174  066061  000020  000002         ADD   $WRDL(R0),2(R1)  ;INCREMENTED SIZE
4660  016202  010246           5$:   MOV   R2,-(SP)           ;SAVE R2
4661  016204  013702  001616                 MOV   BUFTBL,R2        ;ENTRY COUNT
4662  016210  012705  001620                 MOV   #BUFTBL+2,R5     ;BEGINNING OF TABLE
4663  016214  016504  000002           6$:   MOV   2(R5),R4         ;BLOCK SIZE (IN WORDS)
4664  016220  006304                          ASL   R4               ;CHANGE TO BYTE COUNT
4665  016222  061504                          ADD   (R5),R4          ;ADD BLOCK BEGINNING ADDRESS
4666  016224  020411                          CMP   R4,(R1)          ;R1 STILL POINTS TO INSERTED ENTRY
4667  016226  001406                          BEQ   8$               ;LOWER ADJACENT IN TABLE
4668  016230  062705  000004                 ADD   #4,R5            ;INCREMENT POINTER
4669  016234  005302                          DEC   R2               ;DECREMENT ENTRY COUNT
4670  016236  001366                          BNE   6$               ;CONTINUE LOOKING
4671  016240  005726                          TST   (SP)+            ;RESTORE STACK POINTER
4672  016242  000415                          BR    10$              ;END
4673  016244  012602           8$:   MOV   (SP)+,R2           ;RESTORE R2
4674  016246  066165  000002  000002         ADD   2(R1),2(R5)      ;INCREMENT LOWER BLOCK LENGTH
4675  016254  005337  001616                 DEC   BUFTBL           ;DECREMENT ENTRY COUNT
4676  016260  010105                          MOV   R1,R5            ;GET READY TO COMPRESS
4677  016262  062705  000004                 ADD   #4,R5            ;INCREMENT TO NEXT ENTRY
4678  016266  012521           9$:   MOV   (R5)+,(R1)+        ;COMPRESS TABLE
4679  016270  012521                          MOV   (R5)+,(R1)+      ;MOVE SIZE FIELD DOWN
4680  016272  005302                          DEC   R2               ;DECREMENT ENTRY COUNT
4681  016274  001374                          BNE   9$               ;BR IF NOT FINISHED
4682  016276  012601           10$:  MOV   (SP)+,R1           ;RESTORE R1
4683  016300  000207                          RTS   PC               ;RETURN
4684
4685
4686                            ;FILL THE ASSIGNED BUFFER (IF WRITE OR WRITE CHECK ORDER)
4687                            ;CALL:
4688                            ;       MOV     #DPB,R0             ;DPB ADDRESS
4689                            ;       MOV     #BUFADR,$BUF(R0)    ;LOAD BUFFER ADDRESS INTO THE DPB
4690                            ;       MOVB    #PATTERN,$PATTC(R0)  ;PATTERN CODE
4691                            ;       JSR     PC,FILBUF
4692                            ;       RETURN
4693
4694  016302  104412           FILBUF: SAVREG                      ;SAVE THE REGISTERS
4695  016304  132760  000004  000024         BITB    #BIT02,$CODE(R0)  ;SEE IF READ ORDER
4696  016312  001044                          BNE     4$              ;BR IF READ
4697  016314  016001  000006           1$:   MOV     $BUF(R0),R1     ;BUFFER ADDRESS
4698  016320  016002  000020                 MOV     $WRDL(R0),R2    ;POSITIVE WORD COUNT
4699  016324  132760  000001  000024         BITB    #BIT00,$CODE(R0)  ;SEE IF WRITE HEADER TYPE ORDER
```

```
4700  016332  001413                        BEQ     2$                ;BR IF NOT
4701  016334  016011  000012                MOV     $CYL(R0),(R1)     ;CYLINDER ADDRESS
4702  016340  052721  010000                BIS     #BIT12,(R1)+      ;SET FMT22 BIT
4703  016344  016021  000010                MOV     $SEC(R0),(R1)+    ;MOVE SECTOR & TRACK
4704  016350  005021                        CLR     (R1)+             ;CLEAR FIRST KEY WORD
4705  016352  005021                        CLR     (R1)+             ;CLEAR THE SECOND
4706  016354  162702  000004                SUB     #4,R2             ;ADJUST THE WORD COUNT
4707  016360  003421                        BLE     4$                ;BR IF END OF PATTERN
4708  016362  005004              2$:       CLR     R4                ;CLEAR R4
4709  016364  116004  000030                MOVB    $PATTC(R0),R4     ;RELATIVE PATTERN ADDRESS
4710  016370  016405  002762                MOV     STNDAT(R4),R5     ;PATTERN ADDRESS
4711  016374  012703  000020                MOV     #20,R3            ;PATTERN COUNT
4712  016400  012521              3$:       MOV     (R5)+,(R1)+       ;MOVE THE PATTERN INTO THE BUFFER
4713  016402  005302                        DEC     R2                ;DECREMENT THE WORD COUNT
4714  016404  001407                        BEQ     4$                ;BR IF DONE (WORD COUNT = 0)
4715  016406  005303                        DEC     R3                ;DECREMENT THE PATTERN COUNT
4716  016410  001373                        BNE     3$                ;BR IF MORE PATTERN
4717  016412  012703  000020                MOV     #20,R3            ;RESTORE PATTERN COUNT
4718  016416  016405  002762                MOV     STNDAT(R4),R5     ;RESTORE THE ADDRESS
4719  016422  000766                        BR      3$                ;CONTINUE DISTRIBUTING THE PATTERN
4720  016424  104413              4$:       RESREG                    ;RESTORE THE REGISTERS
4721  016426  000207                        RTS     PC                ;RETURN
4722
4723                                        ;START THE ORDER FOR THE DPB IN R0
4724                                        ;CALL:
4725                                        ;       MOV     #DPB,R0           ;DPB ADDRESS
4726                                        ;       JSR     PC,GODRIV
4727                                        ;       RETURN
4728
4729  016430  010046              GODRIV:   MOV     R0,-(SP)          ;SAVE R0
4730  016432  010037  016442                MOV     R0,2$             ;CURRENT DPB ADDRESS
4731  016436  004037  034160      1$:       JSR     R0,RP04           ;CALL THE DRIVE HANDLER
4732  016442  000000              2$:       .WORD   0                 ;DRIVE BLOCK ADDRESS GOES HERE
4733  016444  000000                        HALT                      ;DRIVER REJECTED REQUEST
4734  016446  012600                        MOV     (SP)+,R0          ;RESTORE R0
4735  016450  062760  000001  000036        ADD     #1,$OPERC(R0)     ;INCREMENT THE OPERATION COUNT
4736  016456  005560  000040                ADC     $OPERC+2(R0)
4737  016462  026060  000034  000012        CMP     $PREVA+2(R0),$CYL(R0)  ;DID ORDER REQUIRE A CYLINDER CHANGE
4738  016470  001405                        BEQ     3$                ;BR IF NOT
4739  016472  062760  000001  000042        ADD     #1,$POSIT(R0)     ;INCREMENT SEEK COUNT
4740  016500  005560  000044                ADC     $POSIT+2(R0)      ;ADD ANY CARRY
4741  016504  000207              3$:       RTS     PC
4742
4743                                        ;GENERATE PARAMETERS FOR THE OPERATION
4744                                        ;CALL:
4745                                        ;       MOV     #DPB,R0           ;DPB ADDRESS
4746                                        ;       JSR     PC,SELPAR
4747                                        ;       RETURN
4748
4749  016506  004737  032424      SELPAR:   JSR     PC,$RAND          ;CYCLE THE RANDOM NUMBER GENERATOR
4750  016512  032777  000001  162420        BIT     #SW0,@SWR         ;SEE IF SW0 SET
4751  016520  001012                        BNE     2$                ;BR IF SET - READ ONLY
4752  016522  012705  000010      1$:       MOV     #10,R5            ;READ/WRITE SELECTION DIVISOR
4753  016526  004737  027066                JSR     PC,GETREM         ;GET SELECTION VALUE
4754  016532  020537  001422                CMP     R5,RATIO          ;DETERMINE IF READ OR WRITE
4755  016536  103003                        BHIS    2$                ;BR IF READ
```

```
4756  016540  004737  017230            JSR     PC,RANWRT       ;SELECT A WRITE ORDER
4757  016544  000406                    BR      3$              ;CONTINUE WITH THE SELECTION
4758  016546  013705  032524    2$:     MOV     $LONUM,R5       ;SELECT READ OPERATION CODE
4759  016552  042705  177776            BIC     #^C1,R5         ;MASK OUT ALL BUT BIT 0
4760  016556  062705  000004            ADD     #4,R5           ;TABLE OFFSET FOR READ CODE
4761  016562  110560  000074    3$:     MOVB    R5,$NCODE(R0)   ;ORDER SELECTION CODE TO CONTROL BLOCK
4762
4763                                    ;GENERATE A RANDOM SECTOR ADDRESS BETWEEN VALUES 'MINSEC' & 'MAXSEC'
4764
4765  016566  016005  000116    RANSEC: MOV     MAXSEC(R0),R5   ;GET MAXIMUM SECTOR ADDRESS
4766  016572  026005  000120            CMP     MINSEC(R0),R5   ;'MINSEC' AND 'MAXSEC' THE SAME ?
4767  016576  001417                    BEQ     2$              ;BR IF THEY ARE
4768  016600  166005  000120            SUB     MINSEC(R0),R5   ;SUBTRACT MINIMUM SECTOR ADDRESS
4769  016604  100002                    BPL     1$              ;BR IF MAX LARGER THAN MIN
4770  016606  062705  000026            ADD     #22.,R5         ;CORRECT THE NUMBER
4771  016612  005205    1$:     INC     R5                      ;INCREMENT DIFFERENCE TO USE AS DIVISOR
4772  016614  004737  027066            JSR     PC,GETREM       ;GET THE RANDOM AUGMENT
4773  016620  066005  000120            ADD     MINSEC(R0),R5   ;NEW ADDRESS
4774  016624  020527  000025            CMP     R5,#21.         ;IS VALUE TOO LARGE ?
4775  016630  101402                    BLOS    2$              ;BR IF NOT
4776  016632  162705  000026            SUB     #22.,R5         ;CORRECT VALUE
4777  016636  110560  000076    2$:     MOVB    R5,$NSEC(R0)    ;STORE SECTOR ADDRESS IN DPB
4778
4779                                    ;GENERATE A RANDOM TRACK ADDRESS BETWEEN VALUES 'MINTRK' & 'MAXTRK'
4780
4781  016642  016005  000112    RANTRK: MOV     MAXTRK(R0),R5   ;GET MAXIMUM TRACK ADDRESS
4782  016646  026005  000114            CMP     MINTRK(R0),R5   ;'MINTRK' AND 'MAXTRK' THE SAME ?
4783  016652  001417                    BEQ     2$              ;BR IF THEY ARE
4784  016654  166005  000114            SUB     MINTRK(R0),R5   ;SUBTRACT MINIMUM TRACK ADDRESS
4785  016660  100002                    BPL     1$              ;BR IF MAX LARGER THAN MIN
4786  016662  062705  000023            ADD     #19.,R5         ;CORRECT THE NUMBER
4787  016666  005205    1$:     INC     R5                      ;INCREMENT DIFFERENCE TO USE AS DIVISOR
4788  016670  004737  027066            JSR     PC,GETREM       ;GET THE RANDOM AUGMENT
4789  016674  066005  000114            ADD     MINTRK(R0),R5   ;NEW TRACK ADDRESS
4790  016700  020527  000022            CMP     R5,#18.         ;IS VALUE TOO LARGE ?
4791  016704  101402                    BLOS    2$              ;BR IF NOT
4792  016706  162705  000023            SUB     #19.,R5         ;CORRECT VALUE
4793  016712  110560  000077    2$:     MOVB    R5,$NTRK(R0)    ;STORE TRACK ADDRESS IN DPB
4794
4795                                    ;GENERATE A RANDOM CYLINDER ADDRESS BETWEEN VALUES 'MINCYL' & 'MAXCYL'
4796
4797  016716  012737  000633  001350            MOV     #411.,CYLIMT    ;ASSUME AN RP04/5
4798  016724  032760  000002  000262            BIT     #BIT01,$RPDT(R0)  ;SEE IF RP06
4799  016732  001403                            BEQ     RANCYL          ;BR IF NOT
4800  016734  012737  001457  001350            MOV     #815.,CYLIMT    ;CHANGE CYLINDER LIMIT
4801  016742  016005  000106    RANCYL: MOV     MAXCYL(R0),R5   ;GET MAXIMUM CYLINDER ADDRESS
4802  016746  026005  000110            CMP     MINCYL(R0),R5   ;'MINCYL' AND 'MAXCYL' THE SAME ?
4803  016752  001417                    BEQ     2$              ;BR IF THEY ARE
4804  016754  166005  000110            SUB     MINCYL(R0),R5   ;SUBTRACT MINIMUM CYLINDER ADDRESS
4805  016760  100002                    BPL     1$              ;BR IF MAX LARGER THAN MIN
4806  016762  063705  001350            ADD     CYLIMT,R5       ;CORRECT THE NUMBER
4807  016766  005205    1$:     INC     R5                      ;INCREMENT DIFFERENCE TO USE AS DIVISOR
4808  016770  004737  027066            JSR     PC,GETREM       ;GET THE RANDOM AUGMENT
4809  016774  066005  000110            ADD     MINCYL(R0),R5   ;NEW CYLINDER ADDRESS
4810  017000  023705  001350            CMP     CYLIMT,R5       ;IS VALUE TOO LARGE ?
4811  017004  101002                    BHI     2$              ;BR IF NOT
```

```
4812  017006  163705  001350              SUB     CYLIMT,R5          ;CORRECT VALUE
4813  017012  010560  000100      2$:     MOV     R5,$NCYL(R0)       ;STORE CYLINDER ADDRESS IN DPB
4814  017016  122760  000003  000074      CMPB    #3,$NCODE(R0)      ;WRITE HEADER & DATA ?
4815  017024  001013                      BNE     RANSIZ             ;BR IF NOT
4816  017026  012760  000404  000102      MOV     #260.,$NWRDL(R0)   ;CHANGE WORD LENGTH TO 260 FOR WRTHD ORDER
4817  017034  023727  001404  000404      CMP     MAXDL,#260.        ;CAN A FULL SECTOR BE WRITTEN ?
4818  017042  103062                      BHIS    RANPAT             ;BR IF IT CAN
4819  017044  013760  001404  000102      MOV     MAXDL,$NWRDL(R0)   ;CHANGE TRANSFER SIZE
4820  017052  000456                      BR      RANPAT             ;CONTINUE WITH THE SELECTION
4821
4822                              ;GENERATE A RANDOM BUFFER LENGTH BETWEEN 4 & THE VALUE IN 'MAXDL'
4823
4824  017054  013705  001404      RANSIZ: MOV     MAXDL,R5           ;GET BUFFER SIZE
4825  017060  005737  001420              TST     WCSEL              ;SELECT A RANDOM WORD COUNT ?
4826  017064  001010                      BNE     1$                 ;BR IF NOT
4827  017066  005205                      INC     R5                 ;INCREMENT THE MAXIMUM SIZE
4828  017070  004737  027066              JSR     PC,GETREM          ;DIVIDE BY MAX VALUE
4829  017074  005705                      TST     R5                 ;IS THE REMAINDER 0 ?
4830  017076  001003                      BNE     1$                 ;NOT 0, CONTINUE
4831  017100  004737  032424              JSR     PC,$RAND           ;CYCLE THE RANDOM NUMBER GENERATOR
4832  017104  000763                      BR      RANSIZ             ;TRY AGAIN
4833  017106  010560  000102      1$:     MOV     R5,$NWRDL(R0)      ;WORD LENGTH TO CONTROL BLOCK
4834  017112  010546                      MOV     R5,-(SP)           ;NEW WORD LENGTH ON STACK FOR CHECK
4835  017114  005046                      CLR     -(SP)              ;MAKE UPPER DIVIDEND ZERO
4836  017116  012746  000400              MOV     #256.,-(SP)        ;SECTOR SIZE IS THE DIVISOR
4837  017122  132760  000001  000074      BITB    #1,$NCODE(R0)      ;SEE IF NEXT ORDER IS A HEADER ORDER
4838  017130  001402                      BEQ     2$                 ;BR IF NOT
4839  017132  062716  000004              ADD     #4,(SP)            ;ADD HEADER SIZE TO SECTOR SIZE
4840  017136  004737  027114      2$:     JSR     PC,LINKDV          ;DIVIDE BUFFER SIZE BY SECTOR SIZE
4841  017142  012616                      MOV     (SP)+,(SP)         ;MOV REMAINDER UP THE STACK
4842  017144  021627  000004              CMP     (SP),#4 ;SEE IF REMAINDER LESS THAN 4
4843  017150  103012                      BHIS    4$                 ;BR IF NOT
4844  017152  005737  001420              TST     WCSEL              ;SELECTING RANDOM TRANSFER SIZES ?
4845  017156  001403                      BEQ     3$                 ;BR IF YES
4846  017160  161660  000102              SUB     (SP),$NWRDL(R0)    ;ADJUST WORD LENGTH DOWNWARD
4847  017164  000404                      BR      4$                 ;CONTINUE
4848  017166  005726              3$:     TST     (SP)+              ;CORRECT THE STACK POINTER
4849  017170  004737  032424              JSR     PC,$RAND           ;CYCLE THE RANDOM NUMBER GENERATOR
4850  017174  000727                      BR      RANSIZ             ;TRY AGAIN
4851  017176  005726              4$:     TST     (SP)+              ;CORRECT THE STACK POINTER
4852  017200  122760  000002  000074      CMPB    #2,$NCODE(R0)      ;SEE IF WRITE DATA
4853  017206  001004                      BNE     RANXIT             ;BR IF NOT WRITE DATA
4854
4855                              ;GET A RANDOM PATTERN NUMBER
4856
4857  017210  004737  017334      RANPAT: JSR     PC,GETPAT          ;GET PATTERN CODE
4858  017214  110560  000075              MOVB    R5,$NPATC(R0)      ;MOVE PATTERN CODE TO CONTROL BLOCK
4859  017220  012760  177777  000104      MOV     #-1,$NEXT(R0)  ;SET PARAMETERS SELECTED INDICATOR
4860  017226  000207              RANXIT: RTS     PC                 ;RETURN
4861
4862                              ;ROUTINE TO SELECT A WRITE (OR WRITE CHECK) OPERATION
4863
4864  017230  012705  000004      RANWRT: MOV     #4,R5              ;WRITE OPERATION SELECTION DIVISOR
4865  017234  004737  027066              JSR     PC,GETREM          ;GET SELECTION CODE
4866  017240  005737  001424              TST     AUTOCK             ;ARE WRITE CHECK ORDERS TO BE SELECTED
4867                                                                 ;RANDOMLY ?
```

```
 4868  017244  001403                          BEQ     1$              ;BR IF THEY ARE
 4869  017246  152705  000002                  BISB    #2,R5           ;SET CODE TO EXCLUDE WRITE CHECK ORDERS
 4870  017252  000420                          BR      3$              ;COMPLETE SELECTION
 4871  017254  020527  000001          1$:     CMP     R5,#1           ;WRITE CHECK SELECTED ?
 4872  017260  101015                          BHI     3$              ;BR IF NOT
 4873  017262  132760  000002  000024          BITB    #2,$CODE(R0)    ;PREVIOUS WRITE OPERATION ?
 4874  017270  001407                          BEQ     2$              ;BR IF PREVIOUS WAS READ OR WRITE CHECK
 4875  017272  116060  000024  000074          MOVB    $CODE(R0),$NCODE(R0)  ;MOVE CODE TO 'NEXT CODE'
 4876  017300  142760  000002  000074          BICB    #2,$NCODE(R0)   ;CHANGE WRITE TO WRITE CHECK
 4877  017306  000411                          BR      5$              ;EXIT
 4878  017310  052705  000002          2$:     BIS     #2,R5           ;CHANGE WRITE CHECK TO WRITE
 4879  017314  005737  001416          3$:     TST     FORMAT          ;WRITE HEADER ORDERS ALLOWED ?
 4880  017320  001002                          BNE     4$              ;BR IF THEY ARE
 4881  017322  042705  000001                  BIC     #1,R5           ;ALTER POSSIBLE WRITE HEADER
 4882  017326  110560  000074          4$:     MOVB    R5,$NCODE(R0)   ;SETUP 'NEXT' CODE
 4883  017332  000207                  5$:     RTS     PC              ;RETURN
 4884
 4885                                          ;ROUTINE TO SELECT A PATTERN
 4886
 4887  017334  012705  000020          GETPAT: MOV     #20,R5          ;SELECT PATTERN
 4888  017340  004737  027066                  JSR     PC,GETREM       ;GET CODE
 4889  017344  005705                          TST     R5              ;WAS PATTERN ZERO SELECTED ?
 4890  017346  001003                          BNE     1$              ;BR IF NOT ZERO
 4891  017350  004737  032424                  JSR     PC,$RAND        ;CYCLE THE RANDOM NUMBER GENERATOR
 4892  017354  000767                          BR      GETPAT          ;TRY AGAIN
 4893  017356  006305                  1$:     ASL     R5              ;MAKE CODE INTO TABLE INDEX
 4894  017360  000207                          RTS     PC
 4895
 4896                                          ;ROUTINE TO GET THE PREVIOUSLY SELECTED PARAMETER VALUES
 4897                                          ;CALL:
 4898                                          ;       MOV     #DPB,R0         ;DPB ADDRESS
 4899                                          ;       JSR     PC,SELPAR       ;SELECT THE PARAMETERS
 4900                                          ;       JSR     PC,GETPAR
 4901                                          ;       RETURN
 4902
 4903  017362  010546                  GETPAR: MOV     R5,-(SP)        ;SAVE R5
 4904  017364  116060  000234  000027          MOVB    $RPCS1(R0),$PREV0(R0)  ;SAVE CURRENT PARAMETERS
 4905  017372  032760  000006  000074          BIT     #6,$NCODE(R0)   ;SEE IF NEXT OPERATION IS READ OR WRITE
 4906  017400  001007                          BNE     1$              ;BR IF EITHER
 4907  017402  016060  000012  000034          MOV     $CYL(R0),$PREVA+2(R0)  ;SAVE STARTING CYLINDER
 4908  017410  016060  000010  000032          MOV     $SEC(R0),$PREVA(R0)  ;SAVE STARTING SECTOR AND TRACK
 4909  017416  000411                          BR      2$
 4910  017420  004737  022354          1$:     JSR     PC,READDR       ;GET THE DECREMENTED SECTOR AND TRACK ADDRESSES
 4911  017424  112660  000033                  MOVB    (SP)+,$PREVA+1(R0)  ;TRACK ADDRESS
 4912  017430  112660  000032                  MOVB    (SP)+,$PREVA(R0)  ;SECTOR ADDRESS
 4913  017434  016060  000272  000034          MOV     $RPCC(R0),$PREVA+2(R0)  ;CURRENT CYLINDER
 4914  017442  032777  000100  161470  2$:     BIT     #SW06,@SWR      ;SWITCH 6 SET ?
 4915  017450  001043                          BNE     3$              ;BR IF SET
 4916  017452  116060  000074  000024          MOVB    $NCODE(R0),$CODE(R0)  ;LOGICAL CODE FOR OPERATION
 4917  017460  116005  000074                  MOVB    $NCODE(R0),R5   ;LOAD R5 FOR USE AS TABLE INDEX
 4918  017464  116560  001760  000002          MOVB    COMTBL(R5),$COMND(R0)  ;RP04 COMMAND CODE
 4919  017472  116060  000075  000030          MOVB    $NPATC(R0),$PATTC(R0)  ;PATTERN CODE
 4920  017500  016060  000076  000010          MOV     $NSEC(R0),$SEC(R0)  ;TRACK AND SECTOR ADDRESSES
 4921  017506  016060  000100  000012          MOV     $NCYL(R0),$CYL(R0)  ;CYLINDER ADDRESS
 4922  017514  016060  000102  000020          MOV     $NWRDL(R0),$WRDL(R0)  ;BUFFER SIZE
 4923  017522  016060  000102  000004          MOV     $NWRDL(R0),$WRDM(R0)  ;WORD COUNT FOR THE RH11
```

```
4924  017530  005460  000004                       NEG     $WRDM(R0)          ;COMPLEMENT IT
4925  017534  012760  000400  000022               MOV     #256.,$SSEC(R0)    ;INITIAL VALUE OF SECTOR SIZE
4926  017542  032760  000001  000024               BIT     #1,$CODE(R0)       ;HEADER OPERATION ?
4927  017550  001403                               BEQ     3$                 ;BR IF NOT
4928  017552  062760  000004  000022               ADD     #4,$SSEC(R0)       ;ADD HEADER SIZE
4929  017560  005060  000104              3$:      CLR     $NEXT(R0)          ;RESET 'PARAMETERS LOADED' INDICATOR
4930  017564  012605                               MOV     (SP)+,R5           ;RESTORE R5
4931  017566  000207                               RTS     PC                 ;RETURN
4932
4933                                     ;ROUTINE TO COMPRESS A LIST
4934                                     ;CALL:
4935                                     ;       MOV     #ADDRS,R1          ;COMPRESS LIST STARTING AT THIS ADDRESS
4936                                     ;       JSR     PC,CMPRES
4937                                     ;       RETURN
4938
4939  017570  016111  000002            CMPRES: MOV     2(R1),(R1)         ;COMPRESS THE TABLE IN R1
4940  017574  001403                               BEQ     1$                 ;BR WHEN ZERO FOUND
4941  017576  062701  000002                       ADD     #2,R1              ;INCREMENT R1
4942  017602  000772                               BR      CMPRES             ;CONTINUE COMPRESSING TABLE
4943  017604  000207                      1$:      RTS     PC                 ;RETURN
4944
4945                                     ;ROUTINE TO SETUP PARAMETERS FOR A SEQUENTIAL READ OR WRITE OF THE DISK
4946                                     ;CALL:
4947                                     ;       MOV     #DPB,R0            ;DPB ADDRESS
4948                                     ;       MOV     #-1,$PACK(R0)      ;'WRITE PACK' FLAG
4949                                     ;  OR
4950                                     ;       MOV     #1,$PACK(R0)       ;'READ PACK' FLAG
4951                                     ;       JSR     PC,WRTPK
4952                                     ;       RETURN
4953
4954  017606  004737  032424            WRTPK:  JSR     PC,$RAND           ;CYCLE THE RANDOM NUMBER GENERATOR
4955  017612  005760  000040                       TST     $OPERC+2(R0)       ;SEE IF FIRST OPERATION
4956  017616  001007                               BNE     WRTPK1             ;BR IF UPPER WORD OF COUNTER NOT ZERO
4957  017620  005760  000036                       TST     $OPERC(R0)         ;LOWER WORD ZERO ?
4958  017624  001004                               BNE     WRTPK1             ;BR IF NOT 1ST OPERATION
4959  017626  105760  000026                       TSTB    $PACK(R0)          ;SEE WHICH - 'R' OR 'W'
4960  017632  100503                               BMI     WRTPK3             ;BR IF 'W'
4961  017634  000470                               BR      WRTPK2             ;'R' OPERATION
4962  017636  116060  000234  000027  WRTPK1:  MOVB    $RPCS1(R0),$PREV0(R0)  ;SAVE CURRENT PARAMETERS
4963  017644  004737  022354                       JSR     PC,READDR          ;GET THE DECREMENTED SECTOR AND TRACK ADDRESSES
4964  017650  112660  000033                       MOVB    (SP)+,$PREVA+1(R0) ;TRACK ADDRESS
4965  017654  112660  000032                       MOVB    (SP)+,$PREVA(R0)   ;SECTOR ADDRESS
4966  017660  016060  000272  000034               MOV     $RPCC(R0),$PREVA+2(R0)  ;CURRENT CYLINDER
4967  017666  016060  000242  000010               MOV     $RPDA(R0),$SEC(R0)  ;NEW SECTOR & TRACK ADDRESS
4968  017674  016060  000270  000012               MOV     $RPCA(R0),$CYL(R0)  ;NEW CYLINDER ADDRESS
4969  017702  026060  000012  000106               CMP     $CYL(R0),MAXCYL(R0)  ;SEE IF AT END
4970  017710  103427                               BLO     2$                 ;BR IF LESS THAN 'MAXCYL'
4971  017712  101004                               BHI     1$                 ;BR IF GREATER THAN 'MAXCYL'
4972  017714  126060  000011  000112               CMPB    $TRK(R0),MAXTRK(R0)  ;SEE IF AT MAX TRACK
4973  017722  101422                               BLOS    2$                 ;BR IF NOT GREATER
4974  017724  116060  000114  000011  1$:      MOVB    MINTRK(R0),$TRK(R0)  ;RESET TRACK ADDRESS
4975  017732  116060  000120  000010               MOVB    MINSEC(R0),$SEC(R0)  ;RESET SECTOR ADDRESS
4976  017740  016060  000110  000012               MOV     MINCYL(R0),$CYL(R0)  ;RESET CYLINDER ADDRESS
4977  017746  004737  026666                       JSR     PC,EOP2            ;DROP THE DRIVE (NORMAL TERMINATION)
4978  017752  032777  000020  161160               BIT     #SW04,@SWR         ;IS SWITCH 4 SET ?
4979  017760  001003                               BNE     2$                 ;BR IF SET
```

```
4980  017762  005726                          TST     (SP)+           ;INCREMENT THE STACK POINTER
4981  017764  000137  005434                  JMP     MAIN            ;RETURN DIRECTLY TO 'MAIN'
4982  017770  013760  001404  000020  2$:     MOV     MAXDL,$WRDL(R0)  ;BUFFER SIZE IS MAXIMUM
4983  017776  013760  001404  000004          MOV     MAXDL,$WROM(R0)  ;WORD COUNT
4984  020004  005460  000004                  NEG     $WROM(R0)       ;CHANGE WORD COUNT TO 2'S COMPLEMENT
4985  020010  105760  000026                  TSTB    $PACK(R0)       ;READ OR WRITE ?
4986  020014  100412                          BMI     WRTPK3          ;BR IF WRITE
4987  020016  012760  000404  000022  WRTPK2: MOV     #260.,$SSEC(R0)  ;SECTOR SIZE FOR READ
4988  020024  112760  000005  000024          MOVB    #5,$CODE(R0)    ;CODE FOR READ HEADER & DATA
4989  020032  112760  000173  000002          MOVB    #RDHD,$COMND(R0)  ;DRIVE CODE FOR OPERATION
4990  020040  000415                          BR      WRTPK4          ;SET UP FOR EXIT
4991  020042  012760  000400  000022  WRTPK3: MOV     #256.,$SSEC(R0) ;SECTOR SIZE
4992  020050  112760  000002  000024          MOVB    #2,$CODE(R0)    ;CODE FOR WRTDAT
4993  020056  112760  000161  000002          MOVB    #WRTDAT,$COMND(R0)  ;OP CODE
4994  020064  004737  017334                  JSR     PC,GETPAT       ;GET PATTERN CODE
4995  020070  110560  000030                  MOVB    R5,$PATTC(R0)   ;PATTERN CODE
4996  020074  005060  000104          WRTPK4: CLR     $NEXT(R0)       ;CLEAR 'PARAMETER SELECTED' INDICATOR
4997  020100  000207                          RTS     PC              ;RETURN
4998
4999                                  ;ROUTINE TO DETERMINE OF ERROR IS AT A LOCATION ON THE PACK DEFINED
5000                                  ;       IN THE BAD TRACK/SECTOR TABLE FOR THE DRIVE.
5001                                  ;CALL:
5002                                  ;       JSR     PC,$POTCK
5003                                  ;       RETURN1                 ;ERROR AT AN ADDRESS IN TABLE
5004                                  ;       RETURN2                 ;NO TABLE ENTRY FOR ERROR ADDRESS OR
5005                                  ;                               ;PARAMETER 'NOTPRT' IS 0
5006
5007  020102                          SPOTCK:
5008  020102  010146                          MOV     R1,-(SP         ;;PUSH R1 ON STACK
5009  020104  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
5010  020106  012701  000124                  MOV     #$BDSEC,R1      ;INCREMENT FOR BAD SECTOR TABLE
5011  020112  060001                          ADD     R0,R1           ;ADD THE BLOCK'S STARTING ADDRESS
5012  020114  012702  000020                  MOV     #16.,R2         ;BAD SECTOR TABLE SIZE COUNT
5013  020120  021160  000272          1$:     CMP     (R1),$RPCC(R0)  ;IS CYLINDER IN THE TABLE ?
5014  020124  001022                          BNE     4$              ;BR IF NOT
5015  020126  105761  000003                  TSTB    3(R1)           ;TRACK ENTRY ?
5016  020132  100426                          BMI     5$              ;BR IF NOT
5017  020134  004737  022354                  JSR     PC,READDR       ;DECREMENT THE SECTOR/TRACK ADDRESS
5018  020140  122661  000003                  CMPB    (SP)+,3(R1)     ;COMPARE THE TRACK ADDRESS
5019  020144  001011                          BNE     3$              ;BR IF IT IS NOT EQUAL
5020  020146  105761  000002                  TSTB    2(R1)           ;IS A SECTOR ADDRESS IN THE TABLE ?
5021  020152  100002                          BPL     2$              ;BR IF ONE IS
5022  020154  005726                          TST     (SP)+           ;INCREMENT THE STACK POINTER
5023  020156  000414                          BR      5$              ;DISPLAY THE MESSAGE
5024  020160  122661  000002          2$:     CMPB    (SP)+,2(R1)     ;COMPARE THE SECTOR ADDRESS
5025  020164  001002                          BNE     4$              ;BR IF NOT EQUAL
5026  020166  000410                          BR      5$              ;CHECK 'NOTPRT'
5027  020170  005726                  3$:     TST     (SP)+           ;INCREMENT THE STACK POINTER
5028  020172  062701  000004          4$:     ADD     #4,R1           ;GO TO THE NEXT LOCATION IN THE TABLE
5029  020176  005711                          TST     (R1)            ;PAST THE TABLE ENTRIES ?
5030  020200  100411                          BMI     6$              ;BR IF PAST
5031  020202  005302                          DEC     R2              ;DECREMENT THE MAXIMUM ENTRY COUNT
5032  020204  001345                          BNE     1$              ;BR IF MORE TO CHECK
5033  020206  000406                          BR      6$              ;END, EXIT
5034  020210  005737  001426          5$:     TST     NOTPRT          ;PRINT THE ERROR ANYWAY ?
5035  020214  001006                          BNE     7$              ;BR IF NOT
```

```
5036  020216  012737  177777  001264          MOV     #-1,BADSEC      ;SET THE INDICATOR FOR THE IDENTIFICATION LINE
5037  020224  062766  000002  000004  6$:     ADD     #2,4(SP)        ;INCREMENT THE RETURN
5038  020232                          7$:
5039  020232  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
5040  020234  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
5041  020236  000207                          RTS     PC              ;RETURN
5042
5043                                  ;;*****************************************************************
5044
5045                                  .SBTTL  ERROR MESSAGE GENERATION ROUTINES
5046
5047                                  ;;*****************************************************************
5048
5049                                  ;PRINT LINE 1 OF ERROR MESSAGE:
5050                                  ;'HH:MM:SS'
5051
5052  020240  032777  002000  160672  LINE1:  BIT     #SW10,@SWR      ;SWITCH 10 SET ?
5053  020246  001402                          BEQ     1$              ;BR IF NOT
5054  020250  104401  001160                  TYPE    ,$BELL          ;RING THE BELL
5055  020254  032777  020000  160656  1$:     BIT     #SW13,@SWR      ;INHIBIT TYPEOUT ?
5056  020262  001403                          BEQ     2$              ;BR IF NOT
5057  020264  104414  001165                  DISPLY  ,$CRLF          ;CR-LF
5058  020270  000404                          BR      3$              ;EXIT
5059  020272  004737  023510          2$:     JSR     PC,$TIME        ;TYPE THE TIME
5060  020276  104414  052216                  DISPLY  ,LINSPO         ;SPACES
5061  020302  000207                  3$:     RTS     PC              ;RETURN & TYPE DESCRIPTION
5062
5063                                  ;PRINT LINE 2 OF ERROR MESSAGE
5064                                  ;'PRESENT ORDER = XXXX  PREVIOUS ORDER = XXXX'
5065                                  ;'* ERROR AT BAD TRACK/SECTOR'
5066                                  ;'DRV RPCS1   RPCS2   RPDS1   RPER1   RPER2   RPER3   RPEC1   RPEC2'
5067                                  ;'RPWC    RPBA    RPDA    RPAS    RPLA    RPDB    RPMR   RPDT'
5068                                  ;'RPSN    RPOF    RPCA    RPCC    STATUS'
5069                                  ;'BUS ADDRESS OR WORD COUNT NOT CONSISTENT'
5070                                  ;'RPBA = XXXXXX   RPWC = XXXXXX'
5071                                  ;'BUFFER ADR = XXXXXX   SIZE = XXXX   ACTUAL NMBR WRDS XFRD = XXX'
5072
5073  020304                          LINE2:
5074  020304  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
5075  020306  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
5076  020310  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
5077  020312  104414  001165                  DISPLY  ,$CRLF          ;CR-LF
5078  020316  005037  020444                  CLR     4$              ;CLEAR MESSAGE ADDRESS STORAGE
5079  020322  005004                          CLR     R4              ;WORKING REGISTER
5080  020324  012737  050302  020444          MOV     #LIN2C,4$       ;ADDRESS OF 'PRESENT ORDER = ' MSG
5081  020332  116004  000234                  MOVB    $RPCS1(R0),R4   ;GET THE OPCODE
5082  020336  042704  177701                  BIC     #^C76,R4        ;SAVE ONLY SIGNIFICANT BITS
5083  020342  004737  020400                  JSR     PC,1$           ;TYPE THE FIRST MNEMONIC
5084  020346  005737  020450                  TST     5$              ;SEE IF MNEMONIC ENTRY FOUND
5085  020352  001440                          BEQ     LINE2A          ;BR IF NOT
5086  020354  012737  050323  020444          MOV     #LIN2P,4$       ;ADDRESS OF 'PREVIOUS ORDER = ' MSG
5087  020362  116004  000027                  MOVB    $PREVO(R0),R4   ;PREVIOUS OPERATION CODE
5088  020366  042704  177701                  BIC     #^C76,R4        ;SAVE ONLY SIGNIFICANT BITS
5089  020372  004737  020400                  JSR     PC,1$           ;TYPE THE PREVIOUS MNEMONIC
5090  020376  000426                          BR      LINE2A          ;CONTINUE
5091  020400  005005                  1$:     CLR     R5              ;CLEAR THE TABLE INDEX
```

```
5092  020402  126504  001766          2$:     CMPB    OPTBL(R5),R4    ;LOOK FOR THE OPCODE
5093  020406  001405                          BEQ     3$              ;BR WHEN OPCODE COUNT EQUALS OPCODE
5094  020410  105765  001766                  TSTB    OPTBL(R5)       ;LOOK FOR END OF TABLE
5095  020414  100402                          BMI     3$              ;BR IF END
5096  020416  005205                          INC     R5              ;INCREMENT THE POINTER
5097  020420  000770                          BR      2$              ;CONTINUE - NOT END OF TABLE
5098  020422  006305                  3$:     ASL     R5              ;SHIFT INDEX
5099  020424  006305                          ASL     R5              ;SHIFT THE INDEX
5100  020426  006305                          ASL     R5              ;SHIFT THE INDEX
5101  020430  012737  002010  020450          MOV     #MNTBL,5$       ;ADDRESS OF ASCII TEXT TABLE
5102  020436  060537  020450                  ADD     R5,5$           ;ADD THE INDEX
5103  020442  104414                          DISPLY                  ;TYPE IT
5104  020444  000000                  4$:     .WORD   0               ;ADDRESS OF 'PRESENT' OR 'PREVIOUS' MESSAGE
5105  020446  104414                          DISPLY                  ;TYPE THE OPERATION MNEMONIC
5106  020450  000000                  5$:     .WORD   0               ;ADDRESS OF MESSAGE
5107  020452  000207                          RTS     PC              ;RETURN TO MAIN ROUTINE
5108  020454  005737  001264          LINE2A: TST     BADSEC          ;PRINT THE BAD SECTOR LINE ?
5109  020460  001404                          BEQ     LINE2B          ;BR IF NOT
5110  020462  104414  001165                  DISPLY  ,$CRLF          ;CR-LF
5111  020466  104414  050347                  DISPLY  ,LIN2S          ;ERROR ADDRESS DEFINED AS BAD AREA
5112  020472  104414  001165          LINE2B: DISPLY  ,$CRLF          ;CR-LF
5113  020476  104414  047676                  DISPLY  ,DH14           ;STANDARD RP04/5/6 REGISTER HEADER
5114  020502  104414  052216                  DISPLY  ,LINSP0         ;TYPE A SPACE
5115  020506  013746  001246                  MOV     UNIT,-(SP)      ;PUT THE DRIVE NUMBER ON THE STACK
5116  020512  004737  022334                  JSR     PC,LINOEC       ;TYPE DRIVE NUMBER
5117  020516  104414  052215                  DISPLY  ,LINSP          ;SPACES
5118  020522  012705  050222                  MOV     #DT14,R5        ;REGISTER INDEXES
5119  020526  004737  020656                  JSR     PC,3$           ;PRINT THE REGISTERS
5120  020532  032777  000040  160400          BIT     #SW05,@SWR      ;PRINT THE OPTIONAL REGISTERS ?
5121  020540  001014                          BNE     1$              ;BR IF NOT
5122  020542  104414  050002                  DISPLY  ,DH15
5123  020546  012705  050244                  MOV     #DT15,R5        ;SECOND DATA LINE
5124  020552  004737  020656                  JSR     PC,3$           ;PRINT THEM
5125  020556  104414  050101                  DISPLY  ,DH16
5126  020562  012705  050266                  MOV     #DT16,R5        ;THIRD DATA LINE
5127  020566  004737  020656                  JSR     PC,3$           ;PRINT THE REGISTERS
5128  020572  032760  000100  000016  1$:     BIT     #BIT6,$STATUS(R0)  ;DATA ERROR ?
5129  020600  001422                          BEQ     2$              ;BR IF NOT
5130  020602  016046  000020                  MOV     $WRDL(R0),-(SP) ;TRANSFER SIZE
5131  020606  066016  000236                  ADD     $RPWC(R0),(SP)  ;ADD REMAINING WORD COUNT
5132  020612  006316                          ASL     (SP)            ;CONVERT TO AN BYTE INCREMENT
5133  020614  066016  000006                  ADD     $BUF(R0),(SP)   ;BUFFER STARTING ADDRESS
5134  020620  022660  000240                  CMP     (SP)+,$RPBA(R0) ;CORRECT BUFFER ADDRESS ?
5135  020624  001410                          BEQ     2$              ;BR IF YES
5136  020626  104414  047271                  DISPLY  ,EM46           ;'BUS ADDRESS AND WORD COUNT ARE NOT CONSISTENT'
5137  020632  104414  001165                  DISPLY  ,$CRLF          ;CR-LF
5138  020636  004737  020750                  JSR     PC,LINE3D       ;PRINT LINE 3D OF ERROR MESSAGE
5139  020642  004737  021366                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
5140  020646                          2$:
5141  020646  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
5142  020650  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
5143  020652  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
5144  020654  000207                          RTS     PC              ;RETURN TO ERROR PROCESSING ROUTINE
5145  020656  012546                  3$:     MOV     (R5)+,-(SP)     ;PUT THE REGISTER INDEX ON THE STACK
5146  020660  060016                          ADD     R0,(SP)         ;ADD DRIVE'S TABLE ADDRESS
5147  020662  017646  000000                  MOV     @(SP),-(SP)     ;VALUE
```

```
5148  020666  004737  022302                  JSR     PC,LINOCT       ;TYPE II
5149  020672  005726                          TST     (SP)+           ;CORRECT THE STACK POINTER
5150  020674  104414  052215                  DISPLY  ,LINSP          ;PRINT 2 SPACES
5151  020700  005715                          TST     (R5)            ;AT END OF LINE ?
5152  020702  001365                          BNE     3$              ;BR IF NOT
5153  020704  104414  001165          4$:     DISPLY  ,$CRLF          ;CR-LF
5154  020710  000207                          RTS     PC              ;RETURN
5155
5156                                  ;PRINT LINE 3 OF ERROR MESSAGE
5157                                  ;'ERROR AT CCC TT SS   PREVIOUS ADR = CCC TT SS'
5158
5159  020712  104414  050403          LINE3:  DISPLY  ,LINM3          ;LINE 3 ENTRANCE
5160  020716  000517                          BR      LIN3.1          ;FINISH PRINTOUT
5161
5162                                  ;PRINT LINE 3A OF ERROR MESSAGE
5163                                  ;'START CYL = CCC   END CYL = CCC'
5164
5165  020720  104414  050421          LINE3A: DISPLY  ,LINN3          ;LINE 3A ENTRANCE
5166  020724  000514                          BR      LIN3.1          ;FINISH ERROR LINE
5167
5168                                  ;PRINT LINE 3B OF ERROR MESSAGE
5169                                  ;'START CYL = CCC   END CYL = CCC   ACTUAL CYL = CCC'
5170
5171  020726  004737  021270          LINE3B: JSR     PC,LIN3.3       ;LINE 3B ENTRANCE
5172  020732  104414  001165                  DISPLY  ,$CRLF
5173  020736  000207                          RTS     PC
5174
5175                                  ;PRINT LINE 3C OF ERROR MESSAGE
5176                                  ;'START CYL = CCC   END CYL = CCC   ACTUAL CYL = CCC   TRK = TT'
5177
5178  020740  004737  021270          LINE3C: JSR     PC,LIN3.3       ;LINE 3C ENTRANCE
5179  020744  000137  021322                  JMP     LIN3.4          ;FINISH MESSAGE
5180
5181                                  ;PRINT LINE 3D OF ERROR MESSAGE
5182                                  ;'RPBA = XXXXXX   RPWC = XXXXXX'
5183
5184  020750  032777  000040  160162  LINE3D: BIT     #SWO5,@SWR      ;SWITCH 5 SET ?
5185  020756  001416                          BEQ     1$              ;BR IF IT IS
5186  020760  104414  050572                  DISPLY  ,LIN83          ;'RPBA = '
5187  020764  016046  000240                  MOV     $RPBA(R0),-(SP) ;BUFFER ADDR REG CONTENTS
5188  020770  004737  022302                  JSR     PC,LINOCT       ;CONVERT TO OCTAL AND TYPE IT
5189  020774  104414  050602                  DISPLY  ,LINW3          ;'   RPWC = '
5190  021000  016046  000236                  MOV     $RPWC(R0),-(SP) ;WORD COUNT REGISTER CONTENTS
5191  021004  004737  022302                  JSR     PC,LINOCT       ;CONVERT TO OCTAL AND TYPE IT
5192  021010  104414  001165                  DISPLY  ,$CRLF
5193  021014  000207                  1$:     RTS     PC
5194
5195                                  ;PRINT LINE 3E OF ERROR MESSAGE
5196                                  ;'START CYL = CCC   START TRK = TT   START SEC = SS'
5197
5198  021016  104414  050466          LINE3E: DISPLY  ,LINS3          ;'START CYL = '
5199  021022  016046  000012                  MOV     $CYL(R0),-(SP)  ;MOVE CYL TO STACK
5200  021026  004737  022334                  JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
5201  021032  104414  052215                  DISPLY  ,LINSP          ;SPACES
5202  021036  104414  050614                  DISPLY  ,LINT3          ;'START TRK = '
5203  021042  005046                          CLR     -(SP)           ;CLEAR STACK
```

```
5204  021044  116016  000011                  MOVB    $TRK(R0),(SP)   ;TRACK TO STACK
5205  021050  004737  022334                  JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
5206  021054  104414  052215                  DISPLY  ,LINSP          ;SPACES
5207  021060  104414  050631                  DISPLY  ,LINSS3         ;'START SEC = '
5208  021064  005046                          CLR     -(SP)           ;CLEAR STACK
5209  021066  116016  000010                  MOVB    $SEC(R0),(SP)   ;SECTOR ADDR TO STACK
5210  021072  004737  022334                  JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
5211  021076  104414  001165                  DISPLY  ,$CRLF
5212  021102  000207                          RTS     PC
5213
5214                                  ;PRINT LINE 3F OF ERROR MESSAGE
5215                                  ;'RPDA = XXXXXX   RPCA = XXXXXX'
5216
5217  021104  032777  000040  160026  LINE3F: BIT     #SW5,@SWR       ;SWITCH 5 SET ?
5218  021112  001420                          BEQ     1$              ;BR IF NOT
5219  021114  104414  050562                  DISPLY  ,LINDA3         ;'RPDA = '
5220  021120  016046  000242                  MOV     $RPDA(R0),-(SP) ;PUT SECTOR/TRACK ADDRESS ON THE STACK
5221  021124  004737  022302                  JSR     PC,LINOCT       ;TYPE IT
5222  021130  104414  052215                  DISPLY  ,LINSP          ;SPACES
5223  021134  104414  050551                  DISPLY  ,LINCA3         ;' RPCA = '
5224  021140  016046  000270                  MOV     $RPCA(R0),-(SP) ;PUT DESIRED CYLINDER ADDRESS ON THE STACK
5225  021144  004737  022302                  JSR     PC,LINOCT       ;TYPE IT
5226  021150  104414  001165                  DISPLY  ,$CRLF
5227  021154  000207                  1$:     RTS     PC
5228
5229                                  ;'CCC TT SS   PREV ADR = CCC TT SS'
5230
5231  021156  016046  000272          LIN3.1: MOV     $RPCC(R0),-(SP) ;PUT CYLINDER ADDR ON STACK
5232  021162  004737  022334                  JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
5233  021166  104414  050416                  DISPLY  ,T              ;PRINT ' T'
5234  021172  004737  022354                  JSR     PC,READDR       ;DECREMENT TRACK AND SECTOR ADDRESSES
5235  021176  004737  022334                  JSR     PC,LINDEC       ;TYPE TRACK IN DECIMAL
5236  021202  104414  050442                  DISPLY  ,S              ;PRINT ' S'
5237  021206  004737  022334                  JSR     PC,LINDEC       ;TYPE SECTOR ADDRESS
5238  021212  104414  050445                  DISPLY  ,LINP3          ;PRINT 'PREV ADDR'
5239  021216  016046  000034                  MOV     $PREVA+2(R0),-(SP) ;PREVIOUS CYLINDER
5240  021222  004737  022334                  JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
5241  021226  104414  050416                  DISPLY  ,T              ;PRINT ' T'
5242  021232  005046                          CLR     -(SP)           ;MAKE ROOM ON THE STACK
5243  021234  116016  000033                  MOVB    $PREVA+1(R0),(SP)  ;PREVIOUS TRACK ADDRESS
5244  021240  004737  022334                  JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
5245  021244  104414  050442                  DISPLY  ,S              ;PRINT ' S'
5246  021250  005046                          CLR     -(SP)           ;MAKE ROOM ON THE STACK
5247  021252  116016  000032                  MOVB    $PREVA(R0),(SP)  ;PREVIOUS SECTOR DDRESS
5248  021256  004737  022334                  JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
5249  021262  104414  001165                  DISPLY  ,$CRLF
5250  021266  000207                          RTS     PC
5251
5252                                  ;'START CYL = CCC   END CYL = CCC'
5253
5254  021270  104414  050466          LIN3.3: DISPLY  ,LINS3          ;LINE '38 & 3C' ENTRANCE
5255  021274  016046  000034                  MOV     $PREVA+2(R0),-(SP) ;PREVIOUS CYLINDER
5256  021300  004737  022334                  JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
5257  021304  104414  050503                  DISPLY  ,LINEN3         ;PRINT 'END CYL'
5258  021310  016046  000272                  MOV     $RPCC(R0),-(SP) ;PRESENT CYLINDER
5259  021314  004737  022334                  JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
```

```
5260  021320  000207                        RTS     PC
5261
5262                                  ;'ACTUAL CYL = CCC   TRK = TT'
5263
5264  021322  104414  050520          LIN3.4: DISPLY  ,LINA3          ;PRINT 'ACTUAL'
5265  021326  013746  054516                  MOV     CYLDER,-(SP)    ;ACTUAL CYLINDER
5266  021332  042716  010000                  BIC     #BIT12,(SP)     ;CLEAR THE FORMAT BIT
5267  021336  004737  022334                  JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
5268  021342  104414  050540                  DISPLY  ,LIN3           ;PRINT TRACK
5269  021346  005046                          CLR     -(SP)           ;CLEAR STACK WORD
5270  021350  116016  000243                  MOVB    $RPDA+1(R0),(SP) ;PUT TRACK ON STACK
5271  021354  004737  022334                  JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
5272  021360  104414  001165                  DISPLY  ,$CRLF
5273  021364  000207                          RTS     PC
5274
5275                                  ;PRINT LINE 4 OF ERROR MESSAGE
5276                                  ;'BUFFER ADR = XXXXXX   SIZE = XXXX   ACTUAL NMBR WRDS XFRD = XXX'
5277
5278  021366  032760  000100  000016 LINE4:  BIT     #BIT06,$STATUS(R0)  ;DATA ERROR ?
5279  021374  001427                          BEQ     1$              ;BR IF NOT
5280  021376  104414  050646                  DISPLY  ,LIN44          ;'PRINT BUFFER'
5281  021402  016046  000006                  MOV     $BUF(R0),-(SP)  ;BUFFER ADDR ON STACK
5282  021406  004737  022302                  JSR     PC,LINOCT       ;CONVERT TO OCTAL & PRINT
5283  021412  104414  050665                  DISPLY  ,LIN54          ;PRINT 'SIZE'
5284  021416  016046  000020                  MOV     $WRDL(R0),-(SP) ;BUFFER SIZE
5285  021422  004737  022334                  JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
5286  021426  104414  050677                  DISPLY  ,LINX4          ;'ACTUAL NMBR WRDS XFRD = '
5287  021432  016046  000240                  MOV     $RPBA(R0),-(SP) ;VALUE IN BUFFER ADDR REGISTER
5288  021436  166016  000006                  SUB     $BUF(R0),(SP)   ;SUBTRACT STARTING ADDRESS
5289  021442  006216                          ASR     (SP)            ;CONVERT INTO A WORD COUNT
5290  021444  004737  022334                  JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
5291  021450  104414  001165                  DISPLY  ,$CRLF          ;CR-LF
5292  021454  000207                  1$:     RTS     PC              ;RETURN
5293
5294                                  ;PRINT LINE 5 OF ERROR MESSAGE
5295                                  ;'GOOD DATA = XXXXXX   BAD DATA = XXXXXX   SECT POS = XXX'
5296
5297  021456  104414  050732          LINE5:  DISPLY  ,LIN05          ;PRINT 'GOOD DATA'
5298  021462  162760  000002  000240          SUB     #2,$RPBA(R0)    ;BACK THE ADDRESS UP
5299  021470  017046  000240                  MOV     @$RPBA(R0),-(SP)  ;'GOOD' DATA - AT THE BUFFER LOCATION
5300  021474  004737  022302                  JSR     PC,LINOCT       ;TYPE IT
5301  021500  104414  050747                  DISPLY  ,LIN55          ;PRINT 'BAD DATA'
5302  021504  016046  000256                  MOV     $RPDB(R0),-(SP) ;BAD DATA FROM BUFFER
5303  021510  004737  022302                  JSR     PC,LINOCT       ;TYPE IT
5304  021514  016046  000236                  MOV     $RPWC(R0),-(SP) ;WORD LENGTH ON STACK
5305  021520  066016  000020                  ADD     $WRDL(R0),(SP)  ;MAKE INTO A POSITIVE NUMBER
5306  021524  005046                          CLR     -(SP)           ;UPPER DIVIDEND TO ZERO
5307  021526  016046  000022                  MOV     $SSEC(R0),-(SP)  ;SECTOR SIZE ON THE STACK
5308  021532  004737  027114                  JSR     PC,LINKDV       ;DIVIDE WORDS XFERED BY SECTOR SIZE
5309  021536  012616                          MOV     (SP)+,(SP)      ;MOVE REMAINDER UP THE STACK
5310  021540  104414  050765                  DISPLY  ,LINP5          ;PRINT 'SECT POS'
5311  021544  004737  022334                  JSR     PC,LINDEC       ;TYPE THE POSITION
5312  021550  104414  001165                  DISPLY  ,$CRLF
5313  021554  000207                          RTS     PC
5314
5315                                  ;PRINT LINE 5A OF THE ERROR MESSAGE
```

```
5316                                    ;'HEADER FROM ERROR SECTOR  XXXXXX  XXXXXX  XXXXXX  XXXXXX'
5317
5318  021556  104414  051003  LINE5A: DISPLY  ,LINS5          ;'HEADER CONTENTS OF ERROR SECTOR'
5319  021562  013746  054516          MOV     CYLDER,-(SP)    ;HEADER POSITION
5320  021566  004737  022302          JSR     PC,LINOCT       ;TYPE IT
5321  021572  104414  052215          DISPLY  ,LINSP          ;SPACES
5322  021576  013746  054520          MOV     CYLDER+2,-(SP)  ;HEADER POSITION +2
5323  021602  004737  022302          JSR     PC,LINOCT       ;TYPE IT
5324  021606  104414  052215          DISPLY  ,LINSP          ;SPACES
5325  021612  013746  054522          MOV     CYLDER+4,-(SP)  ;HEADER POSITION +4
5326  021616  004737  022302          JSR     PC,LINOCT       ;TYPE IT
5327  021622  104414  052215          DISPLY  ,LINSP          ;SPACES
5328  021626  013746  054524          MOV     CYLDER+6,-(SP)  ;HEADER POSITION +6
5329  021632  004737  022302          JSR     PC,LINOCT       ;TYPE IT
5330  021636  104414  052215          DISPLY  ,LINSP          ;SPACES
5331  021642  104414  001165          DISPLY  ,$CRLF
5332  021646  000207                  RTS     PC
5333
5334                                    ;PRINT LINE 5B OF ERROR MESSAGE
5335                                    ;'RPEC1 = XXXXXX   RPEC2 = XXXXXX'
5336
5337  021650  104414  051037  LINE5B: DISPLY  ,LINE5          ;'RPEC1 = '
5338  021654  016046  000300          MOV     $RPEC1(R0),-(SP)  ;PUT REGISTER CONTENTS ON THE STACK
5339  021660  004737  022302          JSR     PC,LINOCT       ;TYPE IT
5340  021664  104414  052215          DISPLY  ,LINSP          ;SPACES
5341  021670  104414  051050          DISPLY  ,LINE0S         ;' RPEC2 = '
5342  021674  016046  000302          MOV     $RPEC2(R0),-(SP)  ;PUT REGISTER CONTENTS ON THE STACK
5343  021700  004737  022302          JSR     PC,LINOCT       ;TYPE IT
5344  021704  104414  001165          DISPLY  ,$CRLF
5345  021710  000207                  RTS     PC              ;RETURN
5346
5347                                    ;PRINT LINE 6 OF ERROR MESSAGE
5348                                    ;'SECTOR IS ECC CORRECTABLE'
5349
5350  021712  104414  051062  LINE6:  DISPLY  ,LIN86          ;ECC CORRECTABLE
5351  021716  104414  001165          DISPLY  ,$CRLF
5352  021722  000207                  RTS     PC
5353
5354                                    ;PRINT LINE 6A OF THE ERROR MESSAGE
5355                                    ;'SECTOR READ CORRECTLY AT OFFSET N'
5356
5357  021724  104414  051115  LINE6A: DISPLY  ,LINC6          ;PRINT 'READ CORRECTLY AT OFFSET N'
5358  021730  000411                  BR      LIN6.1          ;TYPE THE REST OF THE LINE
5359
5360                                    ;PRINT LINE 6B OF THE ERROR MESSAGE
5361                                    ;'SECTOR IS ECC CORRECTABLE AT OFFSET N'
5362
5363  021732  104414  051062  LINE6B: DISPLY  ,LIN86          ;PRINT 'SECTOR IS ECC CORRECTABLE '
5364  021736  000406                  BR      LIN6.1
5365
5366                                    ;PRINT LINE 6C OF THE ERROR MESSAGE
5367                                    ;'CORRECTED ON NTH RETRY'
5368
5369  021740  104414  051144  LINE6C: DISPLY  ,LING6          ;'CORRECTED ON NTH RETRY'
5370  021744  000414                  BR      LIN6.2          ;TYPE THE REST OF THE LINE
5371
```

```
                                ;PRINT LINE 6D OF THE ERROR MESSAGE
                                ;'UNCORRECTABLE AFTER N RETRIES'
                                
021746  104414  051173  LINE6D: DISPLY  ,LIN06          ;'UNCORRECTABLE AFTER N RETRIES'
021752  000411                 BR      LIN6.2          ;FINISH
                                
                                ;TYPE THE OFFSET VALUE IN MICRO-INCHES
                                
021754  006301         LIN6.1: ASL     R1              ;DOUBLE THE OFFSET TABLE INDEX
021756  016137  002240  021766         MOV     OFMTBL(R1),1$   ;ADDRESS OF OFFSET POSITION MESSAGE
021764  104414                 DISPLY
021766  000000         1$:     .WORD   0               ;OFFSET VALUE
021770  104414  001165         DISPLY  ,$CRLF
021774  000207                 RTS     PC
                                
                                ;RETRY COUNT TYPEOUT
                                
021776  005046         LIN6.2: CLR     -(SP)           ;CLEAR STACK
022000  113716  001253         MOVB    RETRY+1,(SP)    ;RETRY COUNT
022004  004737  022334         JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
022010  104414  051162         DISPLY  ,LIN6           ;'RETRY'
022014  104414  001165         DISPLY  ,$CRLF
022020  000207                 RTS     PC
                                
                                ;PRINT LINE 7 OF THE ERROR MESSAGE
                                ;'ORDERS:XXXXX   TOTAL ERRORS:XXX   WRDS XFRD:XXXXXXX   WRDS READ:XXXXXXX'
                                
022022  104414  051246  LINE7:  DISPLY  ,LIN70           ;PRINT ORDER COUNT
022026  012746  000036         MOV     #$OPERC,-(SP)   ;TO STACK
022032  060016                 ADD     R0,(SP)         ;ADD THE BASE ADDRESS
022034  004737  032622         JSR     PC,$DB2D        ;CONVERT IT
022040  004737  027430         JSR     PC,$SUPRS       ;PRINT IT
022044  104414  051325         DISPLY  ,LIN7T          ;TOTAL ERRORS
022050  016046  000056         MOV     $TOTAL(R0),-(SP)        ;TO STACK
022054  004737  022334         JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
022060  104414  051337         DISPLY  ,LIN7X          ;PRINT 'WRDS XFR'
022064  012746  000046         MOV     #$TRANS,-(SP)   ;ADDRESS OF LOW WORD ON STACK
022070  060016                 ADD     R0,(SP)
022072  004737  032622         JSR     PC,$DB2D        ;CONVERT
022076  004737  027430         JSR     PC,$SUPRS       ;PRINT
022102  104414  051353         DISPLY  ,LIN7R          ;'BITS READ'
022106  012746  000052         MOV     #$READ,-(SP)    ;LOW WORD ADDRESS
022112  060016                 ADD     R0,(SP)
022114  004737  032622         JSR     PC,$DB2D        ;CONVERT
022120  004737  027430         JSR     PC,$SUPRS       ;PRINT
022124  104414  001165         DISPLY  ,$CRLF
022130  104414  001165         DISPLY  ,$CRLF
022134  032777  100000  156776         BIT     #SW15,@SWR      ;SEE IF 'HALT ON ERROR' - SWITCH 15
022142  001401                 BEQ     1$              ;BR IF NOT
022144  000000                 HALT                    ;SWITCH 15 HALT
022146  000207         1$:     RTS     PC
                                
                                ;PRINT LINE 7A OF ERROR MESSAGE
                                ;'ORDERS:XXXXX   TOTAL SEEKS=XXXXX   TOTAL MISPOS ERR = XXX   TOTAL SKI,OCYL ERR = XXX'
                                
022150  104414  051246  LINE7A: DISPLY  ,LIN70          ;'ORDERS = '
```

```
5428  022154  012746  000036          MOV     #$OPERC,-(SP)   ;ORDER COUNT INCREMENT
5429  022160  060016                  ADD     R0,(SP)         ;ADD BASE ADDRESS
5430  022162  004737  032622          JSR     PC,$DB2D        ;CONVERT THE COUNT
5431  022166  004737  027430          JSR     PC,$SUPRS       ;PRINT IT
5432  022172  104414  051256          DISPLY  ,LIN7P          ;'TOTAL SEEKS = '
5433  022176  012746  000042          MOV     #$POSIT,-(SP)   ;TOTAL SEEKS
5434  022202  060016                  ADD     R0,(SP)         ;DEVICE TABLE ADDRESS
5435  022204  004737  032622          JSR     PC,$DB2D        ;CONVERT THE SEEK COUNT
5436  022210  004737  027430          JSR     PC,$SUPRS       ;PRINT IT
5437  022214  104414  051220          DISPLY  ,LIN7M          ;'  TOTAL MISPOS ERR = '
5438  022220  016046  000066          MOV     $MISPO(R0),-(SP) ;TOTAL ERRORS
5439  022224  004737  022334          JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
5440  022230  104414  051276          DISPLY  ,LIN7S          ;'  TOTAL SKI.OCYL ERR = '
5441  022234  016046  000064          MOV     $SKI(R0),-(SP)  ;CONVERT & PRINT IT
5442  022240  004737  022334          JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
5443  022244  104414  001165          DISPLY  ,$CRLF
5444  022250  104414  001165          DISPLY  ,$CRLF
5445  022254  032777  100000  156656  BIT     #SW15,@SWR      ;SEE IF HALT ON ERROR - SWITCH 15 SET
5446  022262  001401                  BEQ     1$              ;BR IF NOT
5447  022264  000000                  HALT                    ;SWITCH 15 HALT
5448  022266  000207          1$:     RTS     PC
5449
5450                                  ;PRINT LINE 8 OF THE ERROR MESSAGE
5451                                  ;'DIFFERENT ERROR DURING RETRY'
5452
5453  022270  104414  051370  LINE8:  DISPLY  ,LIN8M
5454  022274  004737  020304          JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
5455  022300  000207                  RTS     PC
5456
5457                                  ;OCTAL TYPEOUT ROUTINE
5458                                  ;CALL:
5459                                  ;       MOV     NUM,-(SP)       ;PUT THE NUMBER ON THE STACK
5460                                  ;       JSR     PC,LINOCT
5461                                  ;       RETURN
5462
5463  022302  016646  000002  LINOCT: MOV     2(SP),-(SP)     ;PUT NUMBER IN PROPER LOCATION ON STACK
5464  022306  004737  030060          JSR     PC,$SB2O        ;CONVERT THE NUMBER TO OCTAL
5465  022312  012637  022326          MOV     (SP)+,1$        ;GET THE ADDRESS OF THE ASCII STRING
5466  022316  062737  000005  022326  ADD     #5.,1$          ;ADDRESS THE LAST 6 ASCII DIGITS
5467  022324  104414                  DISPLY                  ;TYPE IT
5468  022326  000000          1$:     .WORD   0               ;ADDRESS
5469  022330  012616                  MOV     (SP)+,(SP)      ;CORRECT THE STACK
5470  022332  000207                  RTS     PC              ;RETURN
5471
5472                                  ;ROUTINE TO CONVERT THE INPUT NUMBER TO DECIMAL AND TYPE IT WITH
5473                                  ;LEADING ZERO SUPRESSION
5474                                  ;CALL:
5475                                  ;       MOV     NUM,-(SP)       ;PUT THE NUMBER ON THE STACK
5476                                  ;       JSR     PC,LINDEC
5477                                  ;       RETURN
5478
5479  022334  016646  000002  LINDEC: MOV     2(SP),-(SP)     ;SET UP STACK FOR CONVERT
5480  022340  004737  030030          JSR     PC,$SB2D        ;CONVERT IT TO DECIMAL
5481  022344  004737  027430          JSR     PC,$SUPRS       ;TYPE IT (WITH LEADING ZEROS SUPRESSED)
5482  022350  012616                  MOV     (SP)+,(SP)      ;RESTORE STACK POINTER
5483  022352  000207                  RTS     PC
```

```
5484
5485                                    ;;******************************************************************
5486
5487                                    .SBTTL  GENERAL SUPPORT SUBROUTINES
5488
5489                                    ;;******************************************************************
5490
5491                                    ;DECREMENT THE SECTOR-TRACK ADDRESS
5492                                    ;CALL:
5493                                    ;       MOV     #DPB,R0         ;DPB ADDRESS
5494                                    ;       JSR     PC,READDR
5495                                    ;       RETURN
5496                                    ;       (SP) CONTAINS THE TRACK ADDRESS
5497                                    ;       2(SP) CONTAINS THE SECTOR ADDRESS
5498
5499  022354  162706  000004          READDR: SUB     #4,SP           ;DECREMENT THE STACK POINTER
5500  022360  016616  000004                  MOV     4(SP),(SP)      ;MOVE THE RETURN ADDR DOWN THE STACK
5501  022364  005066  000004                  CLR     4(SP)           ;CLEAR STACK FOR SECTOR
5502  022370  005066  000002                  CLR     2(SP)           ;CLEAR STACK FOR TRACK
5503  022374  116066  000242  000004          MOVB    $RPDA(R0),4(SP)  ;INCREMENTED SECTOR ON STACK
5504  022402  005366  000004                  DEC     4(SP)           ;DECREMENT THE SECTOR ADDRESS
5505  022406  100015                          BPL     1$              ;BR IF SECTOR GREATER THAN 0
5506  022410  012766  000025  000004          MOV     #21.,4(SP)      ;JAM SECTOR ADDRESS TO 21(10)
5507  022416  116066  000243  000002          MOVB    $RPDA+1(R0),2(SP)  ;TRACK ADDRESS
5508  022424  005366  000002                  DEC     2(SP)           ;DECREMENT TRACK ADDRESS
5509  022430  100007                          BPL     2$              ;BR IF IT DIDN'T GO NEG
5510  022432  012766  000022  000002          MOV     #18.,2(SP)      ;RESET TRACK TO 18(10)
5511  022440  000403                          BR      2$
5512  022442  116066  000243  000002  1$:     MOVB    $RPDA+1(R0),2(SP)  ;TRACK ADDRESS
5513  022450  000207                  2$:     RTS     PC              ;RETURN
5514
5515                                    ;ROUTINE TO CHECK FOR KW11-L OR KW11-P CLOCKS
5516
5517  022452  012737  177777  001210  CKCLK:  MOV     #-1,CLKFLG      ;CLEAR CLOCK AVAILABILITY FLAG
5518  022460  012737  177777  001206          MOV     #-1,PCLOCK      ;CLEAR KW11-P CLOCK AVAILABILITY FLAG
5519  022466  012737  022546  000004          MOV     #CKCLK1,ERRVEC  ;SET UP VECTOR FOR CLOCK CHECK
5520  022474  005037  000006                  CLR     @#ERRVEC+2      ;NEW PSW
5521  022500  005777  156470                  TST     @$LKCSR         ;CHECK FOR KW11-P
5522  022504  005037  001210                  CLR     CLKFLG          ;SET CLOCK AVAILABILITY FLAG
5523  022510  005037  001206                  CLR     PCLOCK          ;SET KW11-P CLOCK FLAG
5524  022514  013701  001200                  MOV     $LPVEC,R1       ;KW11-P VECTOR ADDRESS
5525  022520  012721  023606                  MOV     #CLOCK,(R1)+    ;SET UP KW11-P VECTOR
5526  022524  012711  000300                  MOV     #300,(R1)       ;PSW - PRI 6
5527  022530  012777  174575  156440          MOV     #-1667.,@$LKCSB ;LOAD COUNTER BUFFER WITH 16.67
5528  022536  012777  000131  156430          MOV     #131,@$LKCSR    ;SET CLOCK - CNT UP, 10US, CONT INT
5529  022544  000437                          BR      CKCLK3
5530  022546  062706  000004          CKCLK1: ADD     #4,SP           ;RESTORE THE STACK POINTER
5531  022552  012737  022614  000004          MOV     #CKCLK2,@#ERRVEC  ;CHANGE ERROR VECTOR TO CHECK FOR KW11-L
5532  022560  005777  156416                  TST     @$LKS           ;LOOK FOR KW11-L
5533  022564  005037  001210                  CLR     CLKFLG          ;SET CLOCK FLAG
5534  022570  013701  001204                  MOV     $LLVEC,R1       ;KW11-L VECTOR ADDRESS
5535  022574  012721  023606                  MOV     #CLOCK,(R1)+    ;SET UP KW11-L VECTOR
5536  022600  012711  000300                  MOV     #300,(R1)       ;PSW - PRI 6
5537  022604  012777  000100  156370          MOV     #100,@$LKS      ;SET KW11-L INTERRUPT
5538  022612  000414                          BR      CKCLK3
5539  022614  062706  000004          CKCLK2: ADD     #4,SP           ;RESTORE THE STACK POINTER
```

```
5540  022620  104401  052773                  TYPE    ,NEDCLK         ;'P OR L CLOCK MUST BE ON SYSTEM'
5541  022624  005737  000042                  TST     42              ;UNDER MONITOR CONTROL ?
5542  022630  001402                          BEQ     1$              ;BR IF NOT
5543  022632  000137  005404                  JMP     $GET42          ;ABORT PROGRAM
5544  022636  000000                  1$:     HALT                    ;HALT
5545  022640  000137  004106                  JMP     START           ;TRY AGAIN
5546  022644  012737  000006  000004  CKCLK3: MOV     #6,@#ERRVEC     ;RESTORE THE ERROR VECTOR
5547  022652  000207                          RTS     PC
5548
5549                                  ;ROUTINE TO DISPLAY STATISTICS FOR ALL DRIVES ASSIGNED
5550                                  ;CALL:
5551                                  ;       JSR     PC,STATPR
5552                                  ;       RETURN
5553
5554  022654  010046                  STATPR: MOV     R0,-(SP)        ;SAVE R0
5555  022656  010446                          MOV     R4,-(SP)        ;SAVE R4
5556  022660  005737  001462                  TST     ASNLST          ;ANY DRIVES ASSIGNED ?
5557  022664  001421                          BEQ     3$              ;BR IF NOT
5558  022666  004737  022764                  JSR     PC,SHDTYP       ;TYPE THE HEADING
5559  022672  005004                          CLR     R4              ;CLEAR THE DRIVE INDEX
5560  022674  006304                  .$:     ASL     R4              ;CHANGE TO INDEX WORDS
5561  022676  016400  001740                  MOV     BLKADR(R4),R0   ;GET THE DRIVE'S BLOCK ADDRESS
5562  022702  006204                          ASR     R4              ;RESTORE R4
5563  022704  136437  033360  001462          BITB    ATABIT(R4),ASNLST ;IS THIS DRIVE ASSIGNED ?
5564  022712  001402                          BEQ     2$              ;BR IF NOT
5565  022714  004737  023006                  JSR     PC,SDETAL       ;TYPE THE PERFORMANCE SUMMARY
5566  022720  005204                  2$:     INC     R4              ;INCREMENT THE INDEX
5567  022722  020427  000010                  CMP     R4,#8.          ;FINISHED ?
5568  022726  001362                          BNE     1$              ;BR IF NOT
5569  022730  012604                  3$:     MOV     (SP)+,R4        ;RESTORE R4
5570  022732  012600                          MOV     (SP)+,R0        ;RESTORE R0
5571  022734  000207                          RTS     PC              ;RETURN
5572
5573                                  ;ROUTINE TO TYPE STATISTICS FOR AN INDIVIDUAL DRIVE
5574                                  ;CALL:
5575                                  ;       MOV     #DPB,R0         ;DPB ADDRESS
5576                                  ;       JSR     PC,TYPEST
5577                                  ;       RETURN
5578
5579  022736  010046                  TYPEST: MOV     R0,-(SP)        ;SAVE R0
5580  022740  010446                          MOV     R4,-(SP)        ;SAVE R4
5581  022742  004737  022764                  JSR     PC,SHDTYP       ;TYPE THE HEADING
5582  022746  005004                          CLR     R4              ;CLEAR R4 FOR DRIVE NUMBER
5583  022750  111004                          MOVB    (R0),R4         ;DRIVE NUMBER
5584  022752  004737  023006                  JSR     PC,SDETAL       ;TYPE THE STATISTICS
5585  022756  012604                          MOV     (SP)+,R4        ;RESTORE R4
5586  022760  012600                          MOV     (SP)+,R0        ;RESTORE R0
5587  022762  000207                          RTS     PC              ;RETURN
5588
5589                                  ;TYPE THE HEADER FOR THE DRIVE PERFORMANCE SUMMARY TYPEOUT
5590                                  ;CALL:
5591                                  ;       JSR     PC,SHDTYP
5592                                  ;       RETURN
5593
5594  022764  004737  023510  SHDTYP: JSR     PC,$TIME        ;TYPE THE TIME OF DAY
5595  022770  004537  027470          JSR     R5,TYPRI4       ;TYPE AT PRIORITY 4
```

```
5596  022774  001165                  $CRLF                   ;CR-LF
5597  022776  004537  027470          JSR     R5,TYPRI4       ;TYPE THE HEADER
5598  023002  052443                  STATHD                  ;HEADER
5599  023004  000207                  RTS     PC              ;RETURN
5600
5601                                  ;TYPE THE PERFORMANCE SUMMARY DATE LINE
5602                                  ;CALL:
5603                                  ;       MOV     #DRIVE,R4       ;DRIVE NUMBER
5604                                  ;       MOV     #DPB,R0         ;DPB ADDRESS
5605                                  ;       RETURN
5606
5607  023006  010246                  SDETAL: MOV     R2,-(SP)        ;SAVE R2
5608  023010  010002                  MOV     R0,R2           ;DPB ADDRESS
5609  023012  062702  000036          ADD     #$OPERC,R2      ;FIRST STATISTICAL FIELD
5610  023016  010446                  MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
5611                                                          ;;TYPE DRIVE NUMBER
5612  023020  104403                  TYPOS                   ;;GO TYPE--OCTAL ASCII
5613  023022     002                  .BYTE   2               ;;TYPE 2 DIGIT(S)
5614  023023     000                  .BYTE   0               ;;SUPPRESS LEADING ZEROS
5615  023024  104401  052215          TYPE    ,LINSP          ;SPACES
5616  023030  016046  000070          MOV     $PASSC(R0),-(SP)  ;PUT THE PASS COUNT ON THE STACK
5617  023034  004737  030030          JSR     PC,$B2D         ;CONVERT IT
5618  023040  004537  027340          JSR     R5,REPLZ        ;TYPE IT
5619  023044  000003                  .WORD   3               ;TYPE 3 DIGITS
5620  023046  104401  052215          TYPE    ,LINSP          ;SPACES
5621  023052  010246                  MOV     R2,-(SP)        ;PUT $OPERC ON THE STACK
5622  023054  004737  032622          JSR     PC,$DB2D        ;CONVERT IT
5623  023060  004537  027340          JSR     R5,REPLZ        ;TYPE $OPERC
5624  023064  000006                  .WORD   6               ;TYPE 6 DIGITS
5625  023066  104401  052215          TYPE    ,LINSP          ;SPACES
5626  023072  062702  000004          ADD     #4,R2           ;INCREMENT R2
5627  023076  010246                  MOV     R2,-(SP)        ;PUT $POSIT ON THE STACK
5628  023100  004737  032622          JSR     PC,$DB2D        ;CONVERT IT
5629  023104  004537  027340          JSR     R5,REPLZ        ;TYPE $POSIT
5630  023110  000006                  .WORD   6               ;TYPE 6 DIGITS
5631  023112  104401  052215          TYPE    ,LINSP          ;SPACES
5632  023116  062702  000004          ADD     #4,R2           ;INCREMENT R2
5633  023122  010246                  MOV     R2,-(SP)        ;PUT $TRANS ON THE STACK
5634  023124  004737  032622          JSR     PC,$DB2D        ;CONVERT $TRANS
5635  023130  004537  027340          JSR     R5,REPLZ        ;TYPE IT
5636  023134  000012                  .WORD   10.             ;TYPE 10 DIGITS
5637  023136  104401  052215          TYPE    ,LINSP          ;SPACES
5638  023142  062702  000004          ADD     #4,R2           ;INCREMENT R2
5639  023146  010246                  MOV     R2,-(SP)        ;PUT $READ ON THE STACK
5640  023150  004737  032622          JSR     PC,$DB2D        ;CONVERT $READ
5641  023154  004537  027340          JSR     R5,REPLZ        ;TYPE IT
5642  023160  000012                  .WORD   10.             ;TYPE 10 DIGITS
5643  023162  104401  052216          TYPE    ,LINSPO         ;1 SPACE
5644  023166  062702  000004          ADD     #4,R2           ;INCREMENT R2
5645  023172  062702  000002          ADD     #2,R2           ;INCREMENT R2 AGAIN
5646  023176  012246                  MOV     (R2)+,-(SP)     ;PUT $SOFT ON THE STACK
5647  023200  004737  030030          JSR     PC,$B2D         ;CONVERT $SOFT
5648  023204  004537  027340          JSR     R5,REPLZ        ;TYPEOUT $SOFT
5649  023210  000004                  .WORD   4               ;TYPE 4 DIGITS
5650  023212  104401  052216          TYPE    ,LINSPO         ;SPACES
5651  023216  012246                  MOV     (R2)+ -(SP)     ;PUT $HARD ON THE STACK
```

```
5652  023220  004737  030030            JSR     PC,$SB2D        ;CONVERT $HARD
5653  023224  004537  027340            JSR     R5,REPLZ        ;TYPEOUT $HARD
5654  023230  000004                    .WORD   4               ;TYPE 4 DIGITS
5655  023232  104401  052216            TYPE    ,LINSPO         ;SPACES
5656  023236  012246                    MOV     (R2)+,-(SP)     ;PUT $SKI ON THE STACK
5657  023240  004737  030030            JSR     PC,$SB2D        ;CONVERT $SKI
5658  023244  004537  027340            JSR     R5,REPLZ        ;TYPEOUT $SKI
5659  023250  000004                    .WORD   4               ;TYPE 4 DIGITS
5660  023252  104401  052216            TYPE    ,LINSPO         ;SPACES
5661  023256  012246                    MOV     (R2)+,-(SP)     ;PUT $MISPO ON THE STACK
5662  023260  004737  030030            JSR     PC,$SB2D        ;CONVERT $MISPO
5663  023264  004537  027340            JSR     R5,REPLZ        ;TYPEOUT $MISPO
5664  023270  000004                    .WORD   4               ;TYPE 4 DIGITS
5665  023272  104401  052216            TYPE    ,LINSPO         ;SPACES
5666  023276  016046  000056            MOV     $TOTAL(R0),-(SP)  ;CALCULATE NUMBER OF OTHER ERRORS
5667  023302  166016  000060            SUB     $SOFT(R0),(SP)  ;SUBTRACT $SOFT FROM $TOTAL
5668  023306  166016  000062            SUB     $HARD(R0),(SP)  ;SUBTRACT $HARD FROM $TOTAL
5669  023312  166016  000064            SUB     $SKI(R0),(SP)   ;SUBTRACT $SKI FROM $TOTAL
5670  023316  166016  000066            SUB     $MISPO(R0),(SP) ;SUBTRACT $MISPO FROM $TOTAL
5671  023322  004737  030030            JSR     PC,$SB2D        ;CONVERT 'OTHER' COUNT
5672  023326  004537  027340            JSR     R5,REPLZ        ;TYPE IT
5673  023332  000004                    .WORD   4               ;TYPE 4 DIGITS
5674  023334  104401  001165            TYPE    ,$CRLF
5675  023340  012602                    MOV     (SP)+,R2        ;RESTORE R2
5676  023342  000207                    RTS     PC
5677
5678
5679
5680                                    ;ROUTINE TO INCREMENT $SOFT
5681                                    ;
5682                                    ;NOTE:  $SOFT WILL NOT BE INCREMENTED BEYOND 9999 (10)
5683
5684  023344  005737  001264    INCSOF: TST     BADSEC          ;SEE IF BAD TRK/SEC INDICATOR SET
5685  023350  001006                    BNE     1$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
5686  023352  026027  000060  023417    CMP     $SOFT(R0),#9999.  ;IS $SOFT ALREADY AT MAXIMUM ?
5687  023360  103002                    BHIS    1$              ;BR IF IT IS
5688  023362  005260  000060                    INC     $SOFT(R0)       ;INCREMENT $SOFT
5689  023366  000207            1$:     RTS     PC              ;RETURN
5690
5691
5692                                    ;ROUTINE TO INCREMENT $HARD
5693                                    ;
5694                                    ;NOTE:  $HARD WILL NOT BE INCREMENTED BEYOND 9999 (10)
5695
5696  023370  005737  001264    INCHRD: TST     BADSEC          ;SEE IF BAD TRK/SEC INDICATOR SET
5697  023374  001006                    BNE     1$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
5698  023376  026027  000062  023417    CMP     $HARD(R0),#9999.  ;IS $HARD ALREADY AT MAXIMUM ?
5699  023404  103002                    BHIS    1$              ;BR IF IT IS
5700  023406  005260  000062                    INC     $HARD(R0)       ;INCREMENT $HARD
5701  023412  000207            1$:     RTS     PC              ;RETURN
5702
5703
5704                                    ;ROUTINE TO INCREMENT $SKI
5705                                    ;
5706                                    ;NOTE:  $SKI WILL NOT BE INCREMENTED BEYOND 9999 (10)
5707
```

```
5708  023414  005737  001264          INCSKI: TST     BADSEC          ;SEE IF BAD TRK/SEC INDICATOR SET
5709  023420  001006                          BNE     1$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
5710  023422  026027  000064  023417          CMP     $SKI(R0),#9999. ;IS $SKI ALREADY AT MAXIMUM ?
5711  023430  103002                          BHIS    1$              ;BR IF IT IS
5712  023432  005260  000064                  INC     $SKI(R0)                ;INCREMENT $SKI
5713  023436  000207                  1$:     RTS     PC              ;RETURN
5714
5715
5716                                  ;ROUTINE TO INCREMENT $MISPO
5717                                  ;
5718                                  ;NOTE:  $MISPO WILL NOT BE INCREMENTED BEYOND 9999 (10)
5719
5720  023440  005737  001264          INCMIS: TST     BADSEC          ;SEE IF BAD TRK/SEC INDICATOR SET
5721  023444  001006                          BNE     1$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
5722  023446  026027  000066  023417          CMP     $MISPO(R0),#9999.  ;IS $MISPO ALREADY AT MAXIMUM ?
5723  023454  103002                          BHIS    1$              ;BR IF IT IS
5724  023456  005260  000066                  INC     $MISPO(R0)              ;INCREMENT $MISPO
5725  023462  000207                  1$:     RTS     PC              ;RETURN
5726
5727
5728                                  ;ROUTINE TO INCREMENT $TOTAL
5729                                  ;
5730                                  ;NOTE:  $TOTAL WILL NOT BE INCREMENTED BEYOND 9999 (10)
5731
5732  023464  005737  001264          INCTOT: TST     BADSEC          ;SEE IF BAD TRK/SEC INDICATOR SET
5733  023470  001006                          BNE     1$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
5734  023472  026027  000056  023417          CMP     $TOTAL(R0),#9999.  ;IS $TOTAL ALREADY AT MAXIMUM ?
5735  023500  103002                          BHIS    1$              ;BR IF IT IS
5736  023502  005260  000056                  INC     $TOTAL(R0)              ;INCREMENT $TOTAL
5737  023506  000207                  1$:     RTS     PC              ;RETURN
5738
5739
5740                                  ;ROUTINE TO TYPE THE TIME
5741
5742  023510  005737  001210          $TIME:  TST     CLKFLG          ;CLOCK ON THE SYSTEM ?
5743  023514  001033                          BNE     '$              ;BR IF NOT
5744  023516  104401  001165                  TYPE    ,$CRLF          ;CR-LF
5745  023522  013746  001266                  MOV     HOUR,-(SP)      ;PUT 'HOURS' ON THE STACK
5746  023526  004737  030030                  JSR     PC,$SB2D        ;CONVERT TO DECIMAL
5747  023532  004537  027340                  JSR     R5,REPLZ        ;TYPE IT
5748  023536  000002                          .WORD   2               ;TYPE 2 DIGITS
5749  023540  104401  053247                  TYPE    ,COLON          ;':'
5750  023544  013746  001270                  MOV     MINUTE,-(SP)    ;PUT 'MINUTES' ON THE STACK
5751  023550  004737  030030                  JSR     PC,$SB2D        ;CONVERT TO DECIMAL
5752  023554  004537  027340                  JSR     R5,REPLZ        ;TYPE IT
5753  023560  000002                          .WORD   2               ;TYPE 2 DIGITS
5754  023562  104401  053247                  TYPE    ,COLON          ;':'
5755  023566  013746  001272                  MOV     SECOND,-(SP)    ;PUT SECONDS ON THE STACK
5756  023572  004737  030030                  JSR     PC,$SB2D        ;CONVERT TO DECIMAL
5757  023576  004537  027340                  JSR     R5,REPLZ        ;TYPE IT
5758  023602  000002                          .WORD   2               ;TYPE 2 DIGITS
5759  023604  000207                  1$:     RTS     PC
5760
5761                                  ;CLOCK HANDLER ROUTINE
5762
5763  023606  005337  001274          CLOCK:  DEC     SIXTEE          ;INCREMENT THE 1/60 SECOND COUNTER
```

```
5764  023612  001035                          BNE     1$              ;BR IF A SECOND NOT COUNTED
5765  023614  013737  001212  001274          MOV     HZ,SIXTEE       ;RESTORE THE VALUE
5766  023622  005237  001272                  INC     SECOND          ;COUNT THE SECOND
5767  023626  022737  000074  001272          CMP     #60.,SECOND     ;AT MAXIMUM ?
5768  023634  001024                          BNE     1$              ;BR IF NOT
5769  023636  005037  001272                  CLR     SECOND          ;CLEAR THE SECOND'S COUNTER
5770  023642  005237  001412                  INC     INTRVL+2        ;COUNT THE PERFORMANCE SUMMARY INTERVAL
5771  023646  005237  001270                  INC     MINUTE          ;COUNT THE MINUTE
5772  023652  022737  000074  001270          CMP     #60.,MINUTE     ;AT MAXIMUM ?
5773  023660  001012                          BNE     1$              ;BR IF NOT
5774  023662  005037  001270                  CLR     MINUTE          ;CLEAR THE MINUTE'S COUNTER
5775  023666  005237  001266                  INC     HOUR            ;COUNT THE HOURS
5776  023672  022737  001747  001266          CMP     #999.,HOUR      ;AT MAXIMUM
5777  023700  103002                          BHIS    1$              ;BR IF NOT
5778  023702  005037  001266                  CLR     HOUR            ;CLEAR THE HOURS
5779  023706  012746  000021          1$:     MOV     #17.,-(SP)      ;17 MS ON THE STACK
5780  023712  004737  037720                  JSR     PC,RPTMR        ;DRIVER TIMER ROUTINE
5781  023716  005737  001410                  TST     INTRVL          ;DISPLAY THE PERFORMANCE SUMMARY ?
5782  023722  001411                          BEQ     2$              ;BR IF NOT
5783  023724  023737  001410  001412          CMP     INTRVL,INTRVL+2 ;DISPLAY INTERVAL FINISHED ?
5784  023732  001005                          BNE     2$              ;BR IF NOT
5785  023734  012737  177777  001214          MOV     #-1,STATIN      ;SET PERFORMANCE SUMMARY DISPLAY FLAG
5786  023742  005037  001412                  CLR     INTRVL+2        ;CLEAR THE PERFORMANCE INTERVAL COUNTER
5787  023746  000002                  2$:     RTI                     ;RETURN
5788
5789                                  ;COMMAND DECODE ROUTINE
5790                                  ;CALL:
5791                                  ;       MOV     #-1,CFLAG       ;'CFLAG' IS NORMALLY SET BY THE TTY SERVICE
5792                                  ;                               ;ROUTINE IN INTERRUPT MODE
5793                                  ;       JSR     PC,KSR
5794                                  ;       RETURN1                 ;SYSTEM BUSY RETURN
5795                                  ;       RETURN2                 ;RETURN AFTER KEYBOARD SERVICED
5796
5797  023750  005737  001440          KSR:    TST     ORDERQ          ;ANY OPERATIONS ACTIVE ?
5798  023754  001003                          BNE     1$              ;BR IF SOME ARE
5799  023756  005037  024002                  CLR     3$              ;CLEAR THE LOOP COUNTER
5800  023762  000410                          BR      KSR1            ;PROCESS THE KEYBOARD REQUEST
5801  023764  062737  000001  024002  1$:     ADD     #1,3$           ;COUNT THE TIMES THROUGH THE LOOP
5802  023772  001002                          BNE     2$              ;BR IF NOT ENOUGH
5803  023774  104401  053360                  TYPE    ,BUSY           ;'SYSTEM BUSY...'
5804  024000  000207                  2$:     RTS     PC              ;PROCESS ANY COMPLETED DRIVES
5805  024002  000000                  3$:     .WORD   0               ;LOOP COUNTER
5806  024004  104412                  KSR1:   SAVREG                  ;SAVE THE REGISTERS
5807  024006  012737  000200  177776          MOV     #PR4,PS         ;SET PRIORITY TO 4
5808  024014  005037  001262                  CLR     CFLAG           ;CLEAR THE 'CONTROL C' FLAG
5809  024020  004737  023510                  JSR     PC,$TIME        ;TYPE THE TIME
5810  024024  005777  155116                  TST     @$TKB           ;CLEAR ANY GARBAGE IN THE TTY BUFFER
5811  024030  104401  053073                  TYPE    ,ENTCOM         ;'ENTER COMMANDS'
5812  024034  104411                          RDLIN                   ;READ THE KEYBOARD
5813  024036  012605                          MOV     (SP)+,R5        ;GET ADDRESS OF INPUT STRING
5814  024040  005737  001262                  TST     CFLAG           ;CHECK THE CONTROL C FLAG
5815  024044  001065                          BNE     7$              ;EXIT IF 'CONTROL C' ENTERED
5816  024046  005205                          INC     R5              ;POINT TO SECOND CHARACTER
5817  024050  122715  000101                  CMPB    #'A,(R5)        ;EQ TO AN 'A'
5818  024054  001410                          BEQ     1$              ;BR IF IT IS
5819  024056  121527  000067                  CMPB    (R5),#'7        ;DRIVE NUMBER GREATER THAN AN ASCII 7 ?
```

```
5820  024062  101054                          BHI     6$              ;BR IF IT IS
5821  024064  121527  000060                  CMPB    (R5),#'0        ;DRIVE NUMBER LESS THAN AN ASCII 0 ?
5822  024070  103451                          BLO     6$              ;BR IF IT IS
5823  024072  142715  177770                  BICB    #^C7,(R5)       ;LEAVE ONLY LOWER 3 BITS IF CHAR NOT 'A'
5824  024076  122765  000124  177777  1$:     CMPB    #'T,-1(R5)      ;EQ TO 'T'
5825  024104  001003                          BNE     2$              ;BR IF NOT EQ
5826  024106  004737  024612                  JSR     PC,NEWASN       ;ASSIGN DRIVE FOR TEST
5827  024112  000442                          BR      7$              ;EXIT
5828  024114  122765  000104  177777  2$:     CMPB    #'D,-1(R5)      ;EQ TO 'D' ?
5829  024122  001003                          BNE     3$              ;BR IF NOT EQ
5830  024124  004737  024622                  JSR     PC,DEASGN       ;DEASSIGN DRIVE
5831  024130  000433                          BR      7$              ;EXIT
5832  024132  122765  000123  177777  3$:     CMPB    #'S,-1(R5)      ;EQ TO 'S'
5833  024140  001003                          BNE     4$              ;BR IF NOT EQ
5834  024142  004737  024730                  JSR     PC,SCMND        ;TYPE STATISTICS
5835  024146  000424                          BR      7$              ;EXIT
5836  024150  122765  000127  177777  4$:     CMPB    #'W,-1(R5)      ;EQ TO 'W'
5837  024156  001007                          BNE     5$              ;BR IF NOT EQ
5838  024160  032777  000001  154752          BIT     #SWD,@SWR       ;IS SWITCH 0 SET ?
5839  024166  001012                          BNE     6$              ;BR IF SET, CAN'T DO 'W' COMMAND
5840  024170  004737  025200                  JSR     PC,DATAPK       ;WRITE A DATA PACK
5841  024174  000411                          BR      7$              ;EXIT
5842  024176  122765  000122  177777  5$:     CMPB    #'R,-1(R5)      ;EQ TO 'R' ?
5843  024204  001003                          BNE     6$              ;BR IF NOT EQ
5844  024206  004737  025212                  JSR     PC,REDAPK       ;READ A DATA PACK
5845  024212  000402                          BR      7$              ;EXIT
5846  024214  104401  053051          6$:     TYPE    ,INVLD          ;TYPE 'INVALID COMMAND' MESSAGE
5847  024220  104413                  7$:     RESREG                  ;RESTORE R0 - R5
5848  024222  062716  000002                  ADD     #2,(SP)         ;INCREMENT THE RETURN ADDRESS
5849  024226  005777  154714                  TST     @#TKB           ;CLEAR THE TTY BUFFER
5850  024232  052777  000100  154704          BIS     #BIT06,@#TKS    ;SET TTY INTERRUPT ENABLE
5851  024240  005037  177776                  CLR     PS              ;SET PRIORITY BACK TO ZERO
5852  024244  000207                          RTS     PC              ;RETURN
5853
5854                          ;ROUTINE TO PROCESS THE ASSIGN REQUEST ('T', 'R', OR 'W' COMMANDS)
5855
5856  024246  122715  000101          ASSIGN: CMPB    #'A,(R5)        ;ASSIGN ALL DRIVES?
5857  024252  001430                          BEQ     ASGN2           ;BR IF ALL DRIVES
5858  024254  111504                  ASGN1:  MOVB    (R5),R4         ;PUT DRIVE # IN R4
5859  024256  012737  052271  026476          MOV     #UNTASN,ASNMSG  ;'DRIVE ASSIGNED' MESSAGE ADDRESS
5860  024264  012703  000001                  MOV     #1,R3           ;RELOAD R3 FOR 1 UNIT
5861  024270  136437  033360  001462          BITB    ATABIT(R4),ASNLST  ;DRIVE ALREADY ASSIGNED ?
5862  024276  001013                          BNE     2$              ;BR IF IT IS
5863  024300  005704                          TST     R4              ;TRYING TO ASSIGN DRIVE 0 ?
5864  024302  001007                          BNE     1$              ;BR IF NOT
5865  024304  012737  052355  026476          MOV     #NOTAVL,ASNMSG  ;'NOT AVAILABLE' MESSAGE ADDRESS
5866  024312  122737  000011  000041          CMPB    #11,41          ;SEE IF LOADED FROM AN RP0/4/5/6
5867  024320  001402                          BEQ     2$              ;BR IF RP04/5/6 IS THE LOAD DEVICE
5868  024322  004737  024406          1$:     JSR     PC,ASGN3        ;SEE IF DRIVE ON THE SYSTEM
5869  024326  000137  026456          2$:     JMP     ASNERR          ;RETURN HERE IF DRIVE NOT AVAIL
5870  024332  000207                          RTS     PC              ;EXIT
5871  024334  122737  000011  000041  ASGN2:  CMPB    #11,41          ;LOADED FROM AN RP04/5/6 ?
5872  024342  001005                          BNE     1$              ;BR IF NOT
5873  024344  012704  000001                  MOV     #1,R4           ;START WITH DRIVE 1
5874  024350  012703  000007                  MOV     #7.,R3          ;SETUP FOR ONLY 7 DRIVES
5875  024354  000403                          BR      2$              ;CONTINUE
```

```
5876  024356  005004                    1$:     CLR     R4                      ;START WITH DRIVE 0
5877  024360  012703  000010                    MOV     #8.,R3                  ;DRIVE COUNT FOR 8 DRIVES
5878  024364  004737  024406            2$:     JSR     PC,ASGN3                ;ASSIGN ALL UNASSIGNED, AVAIL DRIVES
5879  024370  000137  024376            3$:     JMP     4$                      ;DRIVE NOT ON SYSTEM
5880  024374  000207                            RTS     PC                      ;RETURN
5881  024376  012746  024370            4$:     MOV     #3$,-(SP)               ;PUT RETURN ADDRESS ON THE STACK
5882  024402  000137  024524                    JMP     ASGN4                   ;LOOK FOR MORE DRIVES
5883  024406  136437  033360  001462  ASGN3:  BITB    ATABIT(R4),ASNLST       ;DRIVE ALREADY ASSIGNED ?
5884  024414  001043                            BNE     ASGN4                   ;BR IF IT IS
5885  024416  005737  033356            1$:     TST     DTUW                    ;DATA TRANSFER UNDER WAY ?
5886  024422  100375                            BPL     1$                      ;BR IF IT IS
5887  024424  110437  044770                    MOVB    R4,GENDPB               ;DRIVE NUMBER
5888  024430  004737  015304                    JSR     PC,RECALO               ;RECALIBRATE DRIVE
5889  024434  105764  033244                    TSTB    DRVSTA(R4)              ;DRIVE AVAILABLE?
5890  024440  001444                            BEQ     ASGN7                   ;BR IF DRIVE OFFLINE OR NONEXISTENT
5891  024442  100437                            BMI     ASGN6                   ;BR IF DRIVE UNSAFE
5892  024444  006304                            ASL     R4                      ;MAKE R4 INTO WORD INDEX
5893  024446  016464  001740  001506            MOV     BLKADR(R4),NEWUNT(R4)   ;DPB ADDRESS
5894  024454  016400  001740                    MOV     BLKADR(R4),R0           ;PUT BLOCK'S ADDR INTO R0
5895  024460  004737  025224                    JSR     PC,CLRDPB               ;CLEAR BLOCK FOR DRIVE JUST ASSIGNED
5896  024464  004737  025420                    JSR     PC,DRVPRM               ;GET THE DRIVE'S ADDRESS LIMITS
5897  024470  004737  025702                    JSR     PC,GETID                ;GET DRIVE I.D.
5898  024474  004737  026012                    JSR     PC,GETADR               ;GET BAD SECTOR ADDRESSES
5899  024500  012760  000001  000070            MOV     #1,$PASSC(R0)           ;PRESET PASS COUNT TO 1
5900  024506  005737  001216                    TST     PACK                    ;WRITE DATA PACK ?
5901  024512  001403                            BEQ     2$                      ;BR IF NOT
5902  024514  113760  001216  000026            MOVB    PACK,$PACK(R0)          ;SET READ/WRITE DATA PACK INDICATOR
5903  024522  006204                    2$:     ASR     R4                      ;RESTORE DRIVE ADDRESS
5904  024524  005303                    ASGN4:  DEC     R3                      ;DECREMENT DRIVE COUNT
5905  024526  001402                            BEQ     ASGN5                   ;BR IF FINISHED
5906  024530  005204                            INC     R4                      ;INCREMENT DRIVE NUMBER
5907  024532  000725                            BR      ASGN3                   ;CONTINUE
5908  024534  062716  000004            ASGN5:  ADD     #4,(SP)                 ;INCREMENT RETURN
5909  024540  000207                            RTS     PC                      ;RETURN
5910  024542  012737  052374  026476  ASGN6:  MOV     #NOTSAF,ASNMSG          ;'UNSAFE' MESSAGE ADDRESS
5911  024550  000207                            RTS     PC                      ;RETURN
5912  024552  105764  033254            ASGN7:  TSTB    DRVTYP(R4)              ;DRIVE PRESENT?
5913  024556  001405                            BEQ     1$                      ;BR IF NOT
5914  024560  100010                            BPL     2$                      ;BR IF DRIVE OFFLINE
5915  024562  012737  052317  026476            MOV     #NOTRP,ASNMSG           ;ADDRESS OF 'NOT RP04/5/6' MSG
5916  024570  000407                            BR      3$                      ;EXIT
5917  024572  012737  052340  026476  1$:     MOV     #NOTPRS,ASNMSG          ;ADDRESS OF 'NOT PRESENT' MSG
5918  024600  000403                            BR      3$                      ;EXIT
5919  024602  012737  052226  026476  2$:     MOV     #UNTOFF,ASNMSG          ;ADDRESS OF 'DRIVE OFFLINE' MESSAGE
5920  024610  000207                    3$:     RTS     PC                      ;ERROR RETURN
5921
5922                                    ;'T' COMMAND (ROUTINE TO ASSIGN A DRIVE)
5923
5924  024612  005037  001216            NEWASN: CLR     PACK                    ;CLEAR 'W' COMMAND INDICATOR
5925  024616  000137  024246                    JMP     ASSIGN                  ;GO TO THE ASSIGN ROUTINE
5926
5927                                    ;'D' COMMAND (ROUTINE TO DEASSIGN A DRIVE)
5928
5929  024622  005004                    DEASGN: CLR     R4
5930  024624  122715  000101                    CMPB    #'A,(R5)                ;DEASSIGN ALL DRIVES ?
5931  024630  001434                            BEQ     5$                      ;BR IF YES
```

```
5932  024632  012703  000001                    MOV     #BIT00,R3       ;SET R3 FOR ONE UNIT
5933  024636  111504                            MOVB    (R5),R4         ;GET DRIVE NUMBER
5934  024640  136437  033360  001462  1$:       BITB    ATABIT(R4),ASNLST  ;DRIVE ASSIGNED ?
5935  024646  001414                            BEQ     3$              ;BR IF NOT
5936  024650  146437  033360  001462            BICB    ATABIT(R4),ASNLST  ;DELETE THE DRIVE FROM THE ASSIGNED LIST
5937  024656  006304                            ASL     R4              ;MAKE ADDR INTO A WORD INDEX
5938  024660  016464  001740  001464            MOV     BLKADR(R4),DUNIT(R4)  ;PUT ADDRESS IN DEASSIGN LIST
5939  024666  006204                            ASR     R4
5940  024670  005303                    2$:     DEC     R3              ;ANY MORE DRIVES ?
5941  024672  001412                            BEQ     4$              ;BR IF NOT
5942  024674  005204                            INC     R4
5943  024676  000760                            BR      1$
5944  024700  122715  000101            3$:     CMPB    #'A,(R5)        ;DEASSIGN ALL DRIVES ?
5945  024704  001771                            BEQ     2$              ;BR IF YES
5946  024706  012737  052247  026476            MOV     #UNTNOT,ASNMSG  ;ADDR OF 'NOT ASSIGNED' MESSAGE
5947  024714  004737  026456                    JSR     PC,ASNERR       ;REPORT IT
5948  024720  000207                    4$:     RTS     PC
5949  024722  012703  000010            5$:     MOV     #8.,R3          ;SET UNIT COUNT TO 8
5950  024726  000744                            BR      1$
5951
5952                                    ;'S' COMMAND (ROUTINE TO TYPE DRIVE PERFORMANCE SUMMARY)
5953
5954  024730  005004                    SCMND:  CLR     R4
5955  024732  122715  000101                    CMPB    #'A,(R5)        ;ALL STATISTICS ?
5956  024736  001421                            BEQ     2$              ;BR IF YES
5957  024740  111504                            MOVB    (R5),R4         ;GET DRIVE NUMBER
5958  024742  136437  033360  001462            BITB    ATABIT(R4),ASNLST  ;SEE IF DRIVE ASSIGNED
5959  024750  001406                            BEQ     1$              ;BR IF NOT
5960  024752  006304                            ASL     R4              ;MAKE DRIVE ADDR INTO WORD INDEX
5961  024754  016400  001740                    MOV     BLKADR(R4),R0   ;ADDR OF BLOCK
5962  024760  004737  022736                    JSR     PC,TYPEST       ;TYPE DRIVE STATISTICS
5963  024764  000504                            BR      9$              ;EXIT
5964  024766  012737  052247  026476  1$:       MOV     #UNTNOT,ASNMSG  ;ADDR OF 'NOT ASSIGNED' MSG
5965  024774  004737  026456                    JSR     PC,ASNERR       ;TYPE ERROR MESSAGE
5966  025000  000476                            BR      9$              ;EXIT
5967  025002  012703  000010            2$:     MOV     #8.,R3          ;DRIVE COUNT
5968  025006  136437  033360  001462  3$:       BITB    ATABIT(R4),ASNLST  ;SEE IF DRIVE ASSIGNED
5969  025014  001004                            BNE     4$              ;BR IF YES
5970  025016  005204                            INC     R4              ;INCREMENT DRIVE ADDRESS
5971  025020  005303                            DEC     R3              ;DECREMENT COUNTER
5972  025022  001371                            BNE     3$              ;MORE TO CHECK
5973  025024  000464                            BR      9$              ;NONE ASSINGED, RETURN
5974  025026  004737  022654            4$:     JSR     PC,STATPR       ;TYPE ALL STATISTICS
5975  025032  105737  001220                    TSTB    DATE            ;SEE IF 'DATE' ENTERED
5976  025036  001404                            BEQ     11$             ;BR IF NOT
5977  025040  104401  053251                    TYPE    ,DATEIS         ;'DATE: '
5978  025044  104401  001220                    TYPE    ,DATE           ;THE OPERATOR ENTERED DATE
5979  025050  105737  001232            11$:    TSTB    OPERID          ;SEE IF OPERATOR I.D. ENTERED
5980  025054  001404                            BEQ     12$             ;BR IF NOT
5981  025056  104401  053262                    TYPE    ,IDIS           ;'OPERATOR I.D.: '
5982  025062  104401  001232                    TYPE    ,OPERID         ;THE OPERATOR I.D.
5983  025066  104401  053304            12$:    TYPE    ,HEDLIN         ;HEADER LINE
5984  025072  012737  042154  025146            MOV     #DRIVED+$DRVID,6$  ;DRIVE I.D. FIELD ADDRESS
5985  025100  005004                            CLR     R4              ;DRIVE ADDRESS
5986  025102  012703  000010                    MOV     #8.,R3          ;COUNTER
5987  025106  136437  033360  001462  5$:       BITB    ATABIT(R4),ASNLST  ;SEE IF DRIVE ASSIGNED
```

```
5988  025114  001417                          BEQ     7$              ;BR IF NOT ASSIGNED
5989  025116  010446                          MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
5990                                                                  ;;TYPE DRIVE NUMBER
5991  025120  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
5992  025122     002                          .BYTE   2               ;;TYPE 2 DIGIT(S)
5993  025123     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
5994  025124  104401  052213                  TYPE    ,LIN4SP         ;4 SPACES
5995  025130  105777  000012                  TSTB    @6$             ;SEE IF DRIVE I.D. ENTERED
5996  025134  001003                          BNE     10$             ;BR IF DRIVE I.D. PRESENT
5997  025136  104401  053327                  TYPE    ,NONE           ;TYPE 'NONE'
5998  025142  000404                          BR      7$              ;CONTINUE
5999  025144  104401                  10$:    TYPE                    ;TYPE THE DRIVE I.D.
6000  025146  000000                  6$:     .WORD   0               ;ADDRESS OF DRIVE I.D. FIELD HERE
6001  025150  104401  001165                  TYPE    ,$CRLF          ;CR-LF
6002  025154  005303                  7$:     DEC     R3              ;DECREMENT THE COUNTER
6003  025156  001405                          BEQ     8$              ;BR IF AT END
6004  025160  062737  000304  025146          ADD     #$RPEC2+2,6$    ;INCREMENT THE MESSAGE FIELD ADDRESS
6005  025166  005204                          INC     R4              ;INCREMENT DRIVE ADDRESS
6006  025170  000746                          BR      5$              ;CONTINUE
6007  025172  104401  001165          8$:     TYPE    ,$CRLF          ;CR-LF
6008  025176  000207                  9$:     RTS     PC
6009
6010                                  ;'W' COMMAND (ROUTINE TO WRITE A DATA PACK)
6011
6012  025200  012737  177777  001216  DATAPK: MOV     #-1,PACK        ;SET THE 'W' COMMAND INDICATOR
6013  025206  000137  024246                  JMP     ASSIGN          ;ASSIGN REQUESTED DRIVE
6014
6015
6016                                  ;'R' COMMAND (ROUTINE TO READ A DATA PACK)
6017
6018  025212  012737  000001  001216  REDAPK: MOV     #1,PACK         ;SET THE 'READ' INDICATOR
6019  025220  000137  024246                  JMP     ASSIGN          ;ASSIGN THE REQUESTED DRIVE
6020
6021                                  ;ROUTINE TO CLEAR THE DPB FOR THE ASSIGNED DRIVE
6022                                  ;CALL:
6023                                  ;       MOV     #DPB,R0         ;DPB ADDRESS
6024                                  ;       JSR     PC,CLRDPB
6025                                  ;       RETURN
6026
6027  025224                          CLRDPB:
6028  025224  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
6029  025226  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
6030  025230  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
6031  025232  010004                          MOV     R0,R4           ;GET THE DPB ADDRESS
6032  025234  062704  000002                  ADD     #2,R4           ;ADDRESS OF FIRST LOCN TO BE CLEARED
6033  025240  012703  000005                  MOV     #5,R3           ;NUMBER OF LOCNS TO BE CLEARED
6034  025244  005024                  1$:     CLR     (R4)+           ;CLEAR THE LOCATION
6035  025246  005303                          DEC     R3              ;DECREMENT THE COUNTER
6036  025250  001375                          BNE     1$              ;BR IF NOT FINISHED
6037  025252  062704  000002                  ADD     #2,R4           ;MOVE THE ADDRESS PAST THE 'REG' ADDR
6038  025256  012703  000070                  MOV     #$NEXT-$REG,R3  ;NUMBER OF LOCNS TO BE CLEARED
6039  025262  005024                  2$:     CLR     (R4)+           ;CLEAR
6040  025264  162703  000002                  SUB     #2,R3           ;DECREMENT THE LOCN COUNTER
6041  025270  001374                          BNE     2$              ;BR IF NOT FINISHED
6042  025272  062704  000014                  ADD     #12.,R4         ;MOVE PAST ADDRESS LIMITS
6043  025276  012703  000162                  MOV     #$RPEC2-MINSEC,R3  ;NUMBER OF LOCNS TO BE CLEARED
```

```
6044  025302  005024                    3$:     CLR     (R4)+           ;CLEAR A LOCATION
6045  025304  162703  000002                    SUB     #2,R3           ;DECREMENT THE COUNTER
6046  025310  001374                            BNE     3$              ;BR IF NOT DONE
6047  025312  113760  001434  000024            MOVB    BEGCOD,$CODE(R0)  ;INITIAL COMMAND CODE
6048  025320  013701  001434                    MOV     BEGCOD,R1       ;GET THE ACTUAL OP CODE
6049  025324  116160  001760  000002            MOVB    COMTBL(R1),$COMND(R0)  ;OPERATION CODE
6050  025332  113760  001432  000030            MOVB    BEGPAT,$PATTC(R0)  ;PATTERN CODE
6051  025340  106360  000030                    ASLB    $PATTC(R0)      ;CONVERT CODE TO A TABLE INDEX
6052  025344  013760  001436  000020            MOV     BEGSIZ,$WROL(R0)  ;BEGINNING RECORD SIZE
6053  025352  013760  001436  000004            MOV     BEGSIZ,$WRDM(R0)  ;VALUE FOR DATA TRANSFER
6054  025360  005460  000004                    NEG     $WROM(R0)       ;MAKE IT INTO 2'S COMPLEMENT
6055  025364  012760  000400  000022            MOV     #256.,$SSEC(R0)  ;INITIAL VALUE OF SECTOR SIZE
6056  025372  132760  000001  000024            BITB    #1,$CODE(R0)    ;HEADER ORDER ?
6057  025400  001403                            BEQ     4$              ;BR IF NOT
6058  025402  062760  000004  000022            ADD     #4,$SSEC(R0)    ;ADD HEADER SIZE TO SECTOR SIZE
6059  025410                              4$:
6060  025410  012605                            MOV     (SP)+,R5        ;;POP STACK INTO R5
6061  025412  012604                            MOV     (SP)+,R4        ;;POP STACK INTO R4
6062  025414  012603                            MOV     (SP)+,R3        ;;POP STACK INTO R3
6063  025416  000207                            RTS     PC              ;RETURN
6064
6065                                    ;ROUTINE TO GET ADDRESS LIMITS FROM THE OPERATOR
6066
6067  025420  010346                    DRVPRM: MOV     R3,-(SP)        ;SAVE R3
6068  025422  010446                            MOV     R4,-(SP)        ;SAVE R4
6069  025424  005737  000042                    TST     42              ;RUNNING UNDER MONITOR CONTROL
6070  025430  001035                            BNE     3$              ;BR IF YES
6071  025432  104401  053145                    TYPE    ,ENTLMT         ;'ENTER ADDRESSES'
6072  025436  006204                            ASR     R4              ;CONVERT INDEX TO DRIVE NUMBER
6073  025440  010446                            MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
6074                                            ;;TYPE DRIVE NUMBER
6075  025442  104403                            TYPOS                   ;;GO TYPE--OCTAL ASCII
6076  025444     002                            .BYTE   2               ;;TYPE 2 DIGIT(S)
6077  025445     000                            .BYTE   0               ;;SUPPRESS LEADING ZEROS
6078  025446  104401  053636                    TYPE    ,SLASH          ;'/'
6079  025452  012737  052424  025516            MOV     #RP04B,2$       ;ADDRESS OF 'RP04' MESSAGE
6080  025460  132764  000001  033254            BITB    #BIT00,DRVTYP(R4)  ;RP04 ?
6081  025466  001012                            BNE     1$              ;BR IF YES
6082  025470  012737  052431  025516            MOV     #RP05,2$        ;ADDRESS OF 'RP05' MESSAGE
6083  025476  132764  000002  033254            BITB    #BIT01,DRVTYP(R4)  ;RP05 ?
6084  025504  001003                            BNE     1$              ;BR IF YES
6085  025506  012737  052436  025516            MOV     #RP06,2$        ;ADDRESS OF 'RP06' MESSAGE
6086  025514  104401                      1$:   TYPE                    ;TYPE THE MESSAGE WHICH FOLLOWS
6087  025516  000000                      2$:   .WORD   0               ;MESSAGE ADDRESS
6088  025520  104401  001165                    TYPE    ,$CRLF          ;CR-LF
6089  025524  012737  000632  001350    3$:     MOV     #410.,CYLIMT    ;ASSUME AN RP04/5
6090  025532  132764  000004  033254            BITB    #BIT02,DRVTYP(R4)  ;SEE IF RP06
6091  025540  001403                            BEQ     4$              ;BR IF NOT
6092  025542  012737  001456  001350            MOV     #814.,CYLIMT    ;CHANGE LIMIT TO 814
6093  025550  062760  177777  000122    4$:     ADD     #-1,$FIRST(R0)  ;SEE IF FIRST TIME STARTED
6094  025556  103417                            BCS     5$              ;BR IF NOT
6095  025560  013760  001350  000106            MOV     CYLIMT,MAXCYL(R0)  ;LOAD MAXIMUM CYLINDER
6096  025566  005060  000110                    CLR     MINCYL(R0)      ;CLEAR MINIMUM CYLINDER
6097  025572  013760  001346  000112            MOV     TRKLMT,MAXTRK(R0)  ;LOAD MAXIMUM TRACK
6098  025600  005060  000114                    CLR     MINTRK(R0)      ;CLEAR MINIMUM TRACK
6099  025604  013760  001344  000116            MOV     SECLMT,MAXSEC(R0)  ;LOAD MAXIMUM SECTOR
```

```
6100  025612  005060  000120            CLR     MINSEC(R0)          ;CLEAR MINIMUM SECTOR
6101  025616  006304                5$: ASL     R4                  ;SETUP TO ADDRESS WORDS
6102  025620  016403  054016            MOV     TABLE(R4),R3        ;PARAMETER TABLE ADDRESS
6103  025624  013763  001350  000002    MOV     CYLIMT,2(R3)        ;LOAD CYLINDER LIMIT FOR LAST CYLINDER
6104  025632  013763  001350  000010    MOV     CYLIMT,10(R3)       ;LOAD CYLINDER LIMIT FOR STARTING CYLINDER
6105  025640  005737  000042            TST     42                  ;UNDER MONITOR CONTROL ?
6106  025644  001002                    BNE     6$                  ;BR IF YES
6107  025646  004737  026332            JSR     PC,PARENT           ;GET THE DRIVE'S PARAMETERS
6108  025652  116060  000120  000010 6$: MOVB    MINSEC(R0),$SEC(R0)  ;INITIAL SECTOR VALUE
6109  025660  116060  000114  000011    MOVB    MINTRK(R0),$TRK(R0)  ;INITIAL TRACK VALUE
6110  025666  016060  000110  000012    MOV     MINCYL(R0),$CYL(R0)  ;INITIAL CYLINDER VALUE
6111  025674  012604                    MOV     (SP)+,R4            ;RESTORE R4
6112  025676  012603                    MOV     (SP)+,R3            ;RESTORE R3
6113  025700  000207                    RTS     PC                  ;RETURN
6114
6115                                ;ROUTINE TO GET THE DRIVE I.D. FROM THE OPERATOR
6116
6117  025702  010546                GETID: MOV    R5,-(SP)            ;SAVE R5
6118  025704  005737  000042            TST     42                  ;UNDER MONITOR CONTROL ?
6119  025710  001036                    BNE     2$                  ;BR IF NOT
6120  025712  005037  001262        1$: CLR     CFLAG               ;CLEAR THE 'CONTROL C' FLAG
6121  025716  104401  053120            TYPE    ,ENTDRV             ;'ENTER DRV I.D.: '
6122  025722  005046                    CLR     -(SP)               ;CLEAR THE STACK
6123  025724  111016                    MOVB    (R0),(SP)           ;PUT THE DRIVE NUMBER ON THE STACK
6124  025726  104403                    TYPOS                       ;TYPE THE DRIVE NUMBER
6125  025730     002                    .BYTE   2                   ;TYPE 2 DIGITS
6126  025731     000                    .BYTE   0                   ;SUPRESS LEADING ZEROS
6127  025732  104401  001165            TYPE    ,$CRLF              ;CR-LF
6128  025736  104411                    RDLIN                       ;READ THE ENTRY
6129  025740  012605                    MOV     (SP)+,R5            ;GET THE ENTRY ADDRESS
6130  025742  005737  001262            TST     CFLAG               ;'CONTROL C' ENTERED ?
6131  025746  001361                    BNE     1$                  ;BR IF IT WAS
6132  025750  121527  000056            CMPB    (R5),#'.            ;PERIOD ENTERED ?
6133  025754  001414                    BEQ     2$                  ;BR IF YES
6134  025756  112560  000224            MOVB    (R5)+,$DRVID(R0)    ;STORE THE I.D.
6135  025762  112560  000225            MOVB    (R5)+,$DRVID+1(R0)  ;STORE THE I.D.
6136  025766  112560  000226            MOVB    (R5)+,$DRVID+2(R0)  ;STORE THE I.D.
6137  025772  112560  000227            MOVB    (R5)+,$DRVID+3(R0)  ;STORE THE I.D.
6138  025776  112560  000230            MOVB    (R5)+,$DRVID+4(R0)  ;STORE THE I.D.
6139  026002  112560  000231            MOVB    (R5)+,$DRVID+5(R0)  ;STORE THE I.D.
6140  026006  012605                2$: MOV     (SP)+,R5            ;RESTORE R5
6141  026010  000207                    RTS     PC                  ;RETURN
6142
6143                                ;ROUTINE TO GET THE ADDRESSES OF ANY BAD SECTORS (UP TO A MAX OF 16)
6144
6145  026012                        GETADR:
6146  026012  010146                    MOV     R1,-(SP)            ;;PUSH R1 ON STACK
6147  026014  010246                    MOV     R2,-(SP)            ;;PUSH R2 ON STACK
6148  026016  010346                    MOV     R3,-(SP)            ;;PUSH R3 ON STACK
6149  026020  010446                    MOV     R4,-(SP)            ;;PUSH R4 ON STACK
6150  026022  005737  000042            TST     42                  ;UNDER MONITOR CONTROL ?
6151  026026  001012                    BNE     15$                 ;BR IF YES
6152  026030  005037  001262        14$: CLR    CFLAG               ;CLEAR 'CONTROL C' FLAG
6153  026034  104401  053206            TYPE    ,ENTADR             ;ENTER SECTOR ADDRESSES
6154  026040  005046                    CLR     -(SP)               ;CLEAR THE STACK
6155  026042  111016                    MOVB    (R0),(SP)           ;PUT THE DRIVE NUMBER ON THE STACK
```

```
6156  026044  104403                    TYPOS                   ;TYPE THE DRIVE NUMBER
6157  026046      002                   .BYTE   2               ;TYPE 2 DIGITS
6158  026047      000                   .BYTE   0               ;SUPRESS LEADING ZEROS
6159  026050  104401  001165            TYPE    ,$CRLF          ;CR-LF
6160  026054  012703  000040  15$:      MOV     #32.,R3         ;NUMBER OF LOCATIONS IN THE TABLE TO PRESET
6161  026060  012704  000124            MOV     #$BOSEC,R4      ;TABLE INCREMENT
6162  026064  060004                    ADD     R0,R4           ;BLOCK STARTING ADDRESS
6163  026066  012724  177777  1$:       MOV     #-1,(R4)+       ;SET LOCATION TO 1'S
6164  026072  005303                    DEC     R3              ;DECREMENT TABLE SIZE COUNT
6165  026074  001374                    BNE     1$              ;BR IF NOT FINISHED WITH TABLE
6166  026076  005737  000042            TST     42              ;UNDER MONITOR CO TROL ?
6167  026102  001106                    BNE     13$             ;BR IF YES
6168  026104  162704  000100            SUB     #64.,R4         ;SET POINTER TO BEGINNING OF TABLE
6169  026110  012703  000020            MOV     #16.,R3         ;NUMBER OF ADDRESSES IN TABLE
6170  026114  104411            2$:     RDLIN                   ;GET ADDRESS FROM OPERATOR
6171  026116  012601                    MOV     (SP)+,R1        ;TEXT POINTER
6172  026120  005737  001262            TST     CFLAG           ;'CONTROL C' ENTERED ?
6173  026124  001341                    BNE     14$             ;BR IF IT WAS
6174  026126  013702  001350            MOV     CYLIMT,R2       ;UPPER LIMIT OF INPUT
6175  026132  004537  027672            JSR     R5,CK.DIG       ;CHECK THE DIGIT(S)
6176  026136  026300                    12$                     ;CARRIAGE RETURN ONLY ENTERED
6177  026140  026320                    13$                     ;PERIOD ONLY ENTERED
6178  026142  026300                    12$                     ;ILLEGAL INPUT
6179  026144  026156                    4$                      ;TERMINATED WITH A CARRIAGE RETURN
6180  026146  026162                    5$                      ;TERMINATED WITH A ","
6181  026150  026152                    3$                      ;TERMINATED WITH A "."
6182  026152  010214            3$:     MOV     R2,(R4)         ;CYLINDER ADDRESS
6183  026154  000461                    BR      13$             ;EXIT, PERIOD ENTERED
6184  026156  010214            4$:     MOV     R2,(R4)         ;CYLINDER ADDRESS
6185  026160  000442                    BR      11$             ;FINISHED WITH THIS ADDRESS, 'CR' ENTERED
6186  026162  010214            5$:     MOV     R2,(R4)         ;CYLINDER ADDRESS
6187  026164  013702  001346            MOV     TRKLMT,R2       ;UPPER LIMIT OF INPUT
6188  026170  004537  027672            JSR     R5,CK.DIG       ;CHECK THE DIGIT(S)
6189  026174  026300                    12$                     ;CARRIAGE RETURN ONLY ENTERED
6190  026176  026320                    13$                     ;PERIOD ONLY ENTERED
6191  026200  026300                    12$                     ;ILLEGAL INPUT
6192  026202  026216                    7$                      ;TERMINATED WITH A CARRIAGE RETURN
6193  026204  026224                    8$                      ;TERMINATED WITH A ","
6194  026206  026210                    6$                      ;TERMINATED WITH A "."
6195  026210  110264  000003  6$:       MOVB    R2,3(R4)        ;TRACK ADDRESS
6196  026214  000441                    BR      13$             ;EXIT, ENTRY TERMINATED BY PERIOD
6197  026216  110264  000003  7$:       MOVB    R2,3(R4)        ;TRACK ADDRESS
6198  026222  000421                    BR      11$             ;FINISHED WITH THIS ADDRESS, 'CR' ENTERED
6199  026224  110264  000003  8$:       MOVB    R2,3(R4)        ;TRACK ADDRESS
6200  026230  013702  001344            MOV     SECLMT,R2       ;UPPER LIMIT OF INPUT
6201  026234  004537  027672            JSR     R5,CK.DIG       ;CHECK THE DIGIT(S)
6202  026240  026300                    12$                     ;CARRIAGE RETURN ONLY ENTERED
6203  026242  026320                    13$                     ;PERIOD ONLY ENTERED
6204  026244  026300                    12$                     ;ILLEGAL INPUT
6205  026246  026262                    10$                     ;TERMINATED WITH A CARRIAGE RETURN
6206  026250  026300                    12$                     ;TERMINATED WITH A ","
6207  026252  026254                    9$                      ;TERMINATED WITH A "."
6208  026254  110264  000002  9$:       MOVB    R2,2(R4)        ;SECTOR ADDRESS
6209  026260  000417                    BR      13$             ;EXIT, ENTRY TERMINATED BY PERIOD
6210  026262  110264  000002  10$:      MOVB    R2,2(R4)        ;SECTOR ADDRESS
6211  026266  005303            11$:    DEC     R3              ;MORE ENTRIES ?
```

```
6212  026270  001413                        BEQ     13$             ;BR IF NOT
6213  026272  062704  000004                ADD     #4,R4           ;INCREMENT THE TABLE POINTER
6214  026276  000706                        BR      2$              ;CONTINUE
6215  026300  012714  177777        12$:    MOV     #-1,(R4)        ;CLEAR PRESENT TABLE ENTRY
6216  026304  012764  177777  000002        MOV     #-1,2(R4)       ;CLEAR PRESENT TABLE ENTRY
6217  026312  104401  053336                TYPE    ,BADENT         ;'INVALID ENTRY'
6218  026316  000676                        BR      2$              ;TRY AGAIN
6219  026320                        13$:
6220  026320  012604                        MOV     (SP)+,R4        ;;POP STACK INTO R4
6221  026322  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
6222  026324  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
6223  026326  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
6224  026330  000207                        RTS     PC              ;RETURN
6225
6226                                ;PARAMETER ENTRY ROUTINE
6227                                ;CALL
6228                                ;       MOV     #ADR,R3         ;PARAMETER TABLE ADDRESS
6229                                ;       JSR     PC,PARENT       ;GET THE PARAMETERS
6230
6231  026332  010346                PARENT: MOV     R3,-(SP)        ;SAVE THE PARAMETER TABLE ADDRESS
6232  026334  005037  001262                CLR     CFLAG           ;CLEAR THE 'CONTROL C' FLAG
6233  026340  012337  026350        1$:     MOV     (R3)+,3$        ;ADDRESS OF PARAMETER NAME
6234  026344  001442                        BEQ     9$              ;BR IF AT END OF TABLE
6235  026346  104401                        TYPE                    ;TYPE THE PARAMETER NAME
6236  026350  000000                3$:     .WORD   0               ;ADDRESS OF PARAMETER NAME TEXT
6237  026352  012302                        MOV     (R3)+,R2        ;MAXIMUM PARAMETER VALUE
6238  026354  012305                        MOV     (R3)+,R5        ;ADDRESS OF PARAMETER
6239  026356  011546                        MOV     (R5),-(SP)      ;CURRENT VALUE OF PARAMETER
6240  026360  104405                        TYPDS                   ;TYPE THE CURRENT VALUE OF THE PARAMETER
6241  026362  104401  053636                TYPE    ,SLASH          ;'/'
6242  026366  104411                        RDLIN                   ;READ THE KEYBOARD
6243  026370  012601                        MOV     (SP)+,R1        ;INPUT ASCII STRING ADDRESS
6244  026372  005737  001262                TST     CFLAG           ;'CONTROL C' ENTERED ?
6245  026376  001021                        BNE     8$              ;BR IF IT WAS
6246  026400  004537  027672                JSR     R5,CK.DIG       ;CHECK THE DIGIT(S)
6247  026404  026340                        1$                      ;CARRIAGE RETURN ONLY ENTERED
6248  026406  026452                        9$                      ;PERIOD ONLY ENTERED
6249  026410  026424                        6$                      ;ILLEGAL INPUT
6250  026412  026420                        5$                      ;TERMINATED WITH A CARRIAGE RETURN
6251  026414  026424                        6$                      ;TERMINATED WITH A "/"
6252  026416  026436                        7$                      ;TERMINATED WITH A "."
6253  026420  010215                5$:     MOV     R2,(R5)         ;MOVE NEW VALUE TO PARAMETER LOCATION
6254  026422  000746                        BR      1$              ;GET MORE PARAMETERS
6255  026424  104401  053336        6$:     TYPE    ,BADENT         ;'BAD ENTRY'
6256  026430  162703  000006                SUB     #6,R3           ;DECREMENT THE TABLE POINTER
6257  026434  000741                        BR      1$              ;TRY AGAIN
6258  026436  010215                7$:     MOV     R2,(R5)         ;NEW VALUE
6259  026440  000404                        BR      9$              ;EXIT
6260  026442  005037  001262        8$:     CLR     CFLAG           ;CLEAR THE 'CONTROL C' FLAG
6261  026446  011603                        MOV     (SP),R3         ;RELOAD THE PARAMETER TABLE ADDRESS
6262  026450  000733                        BR      1$              ;TRY AGAIN
6263  026452  005726                9$:     TST     (SP)+           ;CORRECT THE STACK POINTER
6264  026454  000207                        RTS     PC              ;RETURN
6265
6266                                ;TYPEOUT ASSIGN/DEASSIGN ERROR MESSAGE
6267                                ;CALL
```

```
6268                                    ;       MOV     #MESADR,ASNMSG  ;ERROR MESSAGE ADDRESS
6269                                    ;       JSR     PC,ASNERR
6270                                    ;       RETURN
6271
6272  026456  104401  053047            ASNERR: TYPE    ,QUES           ;QUESTION MARK
6273  026462  104401  052220                    TYPE    ,UNTMSG         ;TYPE 'DRIVE'
6274  026466  010446                            MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
6275                                                                    ;;TYPE DRIVE NUMBER
6276  026470  104403                            TYPOS                   ;;GO TYPE--OCTAL ASCII
6277  026472     002                            .BYTE   2               ;;TYPE 2 DIGIT(S)
6278  026473     000                            .BYTE   0               ;;SUPPRESS LEADING ZEROS
6279  026474  104401                            TYPE                    ;TYPE SPECIFIC MESSAGE
6280  026476  000000            ASNMSG: .WORD   0                       ;MESSAGE ADDRESS
6281  026500  104401  001165              TYPE    ,$CRLF
6282  026504  000207                    RTS     PC
6283
6284                                    ;DEASSIGN DRIVE IF A FATAL ERROR OCCURS
6285                                    ;CALL
6286                                    ;       JSR     PC,DROP
6287                                    ;       RETURN
6288
6289  026506  005004            DROP:   CLR     R4              ;CLEAR R4 FOR DRIVE NUMBER
6290  026510  111004                    MOVB    (R0),R4         ;MOVE DRIVE NUMBER TO R4
6291  026512  146437  033360  001462    BICB    ATABIT(R4),ASNLST+1 ;REMOVE DRIVE FROM ASSIGNED LIST
6292  026520  006304                    ASL     R4              ;MAKE DRIVE NUMBER INTO A TABLE INDEX
6293  026522  010064  001464            MOV     R0,DUNIT(R4)    ;PUT DRIVE IN DROP LIST
6294  026526  104401  001165              TYPE    ,$CRLF
6295  026532  104401  001165              TYPE    ,$CRLF
6296  026536  104401  052622              TYPE    ,DROPNG         ;TYPE 'DROPPING DRIVE'
6297  026542  104401  052733              TYPE    ,DRNUM          ;'DRIVE #'
6298  026546  006204                    ASR     R4              ;DRIVE NUMBER
6299  026550  010446                    MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
6300                                                            ;;TYPE DRIVE NUMBER
6301  026552  104403                    TYPOS                   ;;GO TYPE--OCTAL ASCII
6302  026554     002                    .BYTE   2               ;;TYPE 2 DIGIT(S)
6303  026555     000                    .BYTE   0               ;;SUPPRESS LEADING ZEROS
6304  026556  104401  001165              TYPE    ,$CRLF
6305  026562  000207                    RTS     PC
6306
6307                                    ;ROUTINE TO DEASSIGN DRIVE IF ERRORS BECOMES EXCESSIVE
6308
6309  026564  032777  000020  152346  ABNRML: BIT     #SW04,@SWR      ;SEE IF SWITCH 4 SET
6310  026572  001006                    BNE     1$              ;BR IF IT'S SET
6311  026574  023760  001406  000056    CMP     MAXER,$TOTAL(R0)  ;CHECK TOTAL ERROR VALUE
6312  026602  103002                    BHIS    1$              ;BR IF ERRORS DONOT EXCEED MAX
6313  026604  000137  026506            JMP     DROP            ;DEASSING THE DRIVE
6314  026610  000207                    1$:     RTS     PC      ;RETURN
6315
6316                                    ;ROUTINE TO CHECK FOR END OF PASS AND END OF TEST
6317
6318  026612  005737  001430            EOP:    TST     ENDET           ;END OF PASS DETERMINED BY SEEKS OR WORDS ?
6319  026616  001412                    BEQ     EOP1            ;BR IF SEEKS
6320  026620  026037  000054  001374    CMP     $READ+2(R0),ENDCON+2  ;CHECK MSW OF WORDS READ COUNT
6321  026626  101017                    BHI     EOP2            ;BR IF MSW GREATER THAN LIMIT
6322  026630  103405                    BLO     EOP1            ;BR IF MSW LESS THAN LIMIT
6323  026632  026037  000052  001372    CMP     $READ(R0),ENDCON  ;CHECK LSW AGAINST LIMIT
```

```
6324  026640  103012                        BHIS    EOP2            ;BR IF EQUAL OR GREATER
6325  026642  000510                        BR      EOPX            ;EXIT
6326  026644  026037  000044  001400  EOP1:  CMP     $POSIT+2(R0),ENDSEK+2  ;CHECK MSW OF SEEK COUNT
6327  026652  101005                        BHI     EOP2            ;BR IF MSW GREATER THAN LIMIT
6328  026654  103503                        BLO     EOPX            ;EXIT IF MSW LESS THAN LIMIT
6329  026656  026037  000042  001376        CMP     $POSIT(R0),ENDSEK  ;CHECK LSW OF SEEK COUNT
6330  026664  103477                        BLO     EOPX            ;EXIT IF LSW LESS THAN LIMIT
6331  026666  104401  001165          EOP2: TYPE    ,$CRLF          ;CR-LF
6332  026672  104401  052656                TYPE    ,ENDPAS         ;END OF PASS FOR THE DRIVE
6333  026676  016046  000070                MOV     $PASSC(R0),-(SP)  ;PUT PASS COUNT ON THE STACK
6334  026702  104405                        TYPOS                   ;CONVERT PASS COUNT TO DECIMAL AND TYPE IT
6335  026704  111037  001246                MOVB    (R0),UNIT       ;STORE THE DRIVE NUMBER
6336  026710  032777  000020  152222        BIT     #SW04,@SWR      ;SWITCH 4 SET ?
6337  026716  001017                        BNE     1$              ;BR IF SET
6338  026720  026037  000070  001402        CMP     $PASSC(R0),PASCNT  ;SEE IF AT END OF TEST
6339  026726  103413                        BLO     1$              ;BR IF NOT
6340  026730  104401  052672                TYPE    ,ENDTST         ;TYPE 'END OF TEST'
6341  026734  104401  052733                TYPE    ,DRNUM          ;'DRIVE #'
6342  026740  013746  001246                MOV     UNIT,-(SP)      ;;SAVE UNIT FOR TYPEOUT
6343                                                                ;;TYPE DRIVE NUMBER
6344  026744  104403                        TYPOS                   ;;GO TYPE--OCTAL ASCII
6345  026746     002                        .BYTE   2               ;;TYPE 2 DIGIT(S)
6346  026747     000                        .BYTE   0               ;;SUPPRESS LEADING ZEROS
6347  026750  104401  001165                TYPE    ,$CRLF          ;CR-LF
6348  026754  000431                        BR      3$              ;DEASSIGN THE DRIVE
6349  026756  104401  052733          1$:   TYPE    ,DRNUM          ;'DRIVE #'
6350  026762  013746  001246                MOV     UNIT,-(SP)      ;;SAVE UNIT FOR TYPEOUT
6351                                                                ;;TYPE DRIVE NUMBER
6352  026766  104403                        TYPOS                   ;;GO TYPE--OCTAL ASCII
6353  026770     002                        .BYTE   2               ;;TYPE 2 DIGIT(S)
6354  026771     000                        .BYTE   0               ;;SUPPRESS LEADING ZEROS
6355  026772  104401  001165                TYPE    ,$CRLF          ;CR-LF
6356  026776  004737  022736                JSR     PC,TYPEST       ;TYPE THE DRIVE'S STATISTICS
6357  027002  010346                        MOV     R3,-(SP)        ;SAVE R3
6358  027004  010446                        MOV     R4,-(SP)        ;SAVE R4
6359  027006  010004                        MOV     R0,R4           ;DRIVE'S BLOCK ADDRESS
6360  027010  062704  000036                ADD     #$OPERC,R4      ;ADD THE STARTING ADDR OF SECTIONS TO CLEAR
6361  027014  012703  000010                MOV     #8.,R3          ;NUMBER OF LOCNS TO BE CLEARED
6362                                                                ;(ERROR COUNTERS NOT CLEARED)
6363  027020  005024                  2$:   CLR     (R4)+           ;CLEAR THE LOCN
6364  027022  005303                        DEC     R3              ;DECREMENT THE LOCATION COUNTER
6365  027024  001375                        BNE     2$              ;BR IF MORE TO GO
6366  027026  012604                        MOV     (SP)+,R4        ;RESTORE R4
6367  027030  012603                        MOV     (SP)+,R3        ;RESTORE R3
6368  027032  005260  000070                INC     $PASSC(R0)      ;INCREMENT THE PASS COUNT
6369  027036  000412                        BR      EOPX            ;EXIT
6370  027040  104401  001165          3$:   TYPE    ,$CRLF
6371  027044  005004                        CLR     R4              ;CLEAR R4 FOR DRIVE NUMBER
6372  027046  111004                        MOVB    (R0),R4         ;MOVE DRIVE NUMBER
6373  027050  146437  033360  001462        BICB    ATABIT(R4),ASNLST  ;DELETE DRIVE FROM ASSIGNED LIST
6374  027056  006304                        ASL     R4              ;MAKE DRIVE NUMBER INTO TABLE INDEX
6375  027060  010064  001464                MOV     R0,DUNIT(R4)    ;PUT BLOCK ADDRESS INTO DROP LIST
6376  027064  000207                  EOPX: RTS     PC              ;RETURN
6377
6378                                        ;ROUTINE TO GET THE REMAINDER OF THE RANDOM NUMBER
6379                                        ;CALL
```

```
6380                                    ;       MOV     NUMBER,R5       ;DIVISOR INTO R5
6381                                    ;       JSR     PC,GETREM
6382                                    ;       RETURN                  ;REMAINDER IS IN R5
6383
6384  027066  013746  032524           GETREM: MOV     $LONUM,-(SP)    ;STORE RANDOM NUMBER ON THE STACK FOR DIVIDE
6385  027072  013746  032522                   MOV     $HINUM,-(SP)    ;UPPER PART
6386  027076  010546                           MOV     R5,-(SP)        ;PUT THE DIVISOR ONTO THE STACK
6387  027100  004737  027114                   JSR     PC,LINKDV       ;DIVIDE THE RANDOM NUMBERS
6388  027104  012605                           MOV     (SP)+,R5        ;PUT THE REMAINDER INTO R5
6389  027106  005726                           TST     (SP)+           ;ADJUST THE STACK POINTER
6390  027110  000240                           NOP                     ;FOR DEBUGGING HALT
6391  027112  000207                           RTS     PC
6392
6393                                    ;LINK ROUTINE TO THE DIVISION UTILITY SUBROUTINE
6394                                    ;       THIS ROUTINE ALLOWS THE 'SYSMAC' DIVIDE ROUTINE
6395                                    ;       CALLING SEQUENCE TO BE USED
6396
6397  027114  104412                   LINKDV: SAVREG                  ;STORE R0 - R5
6398  027116  016605  000026                   MOV     26(SP),R5       ;DIVISOR
6399  027122  005004                           CLR     R4              ;OTHER DIVISOR WORD
6400  027124  016602  000030                   MOV     30(SP),R2       ;UPPER DIVIDEND WORD
6401  027130  016603  000032                   MOV     32(SP),R3       ;LOWER DIVIDEND WORD
6402  027134  005000                           CLR     R0              ;CLEAR OTHER DIVIDEND REGISTERS
6403  027136  005001                           CLR     R1
6404  027140  004737  027162                   JSR     PC,M.DPID       ;GO TO THE DIVIDE ROUTINE
6405  027144  010166  000030                   MOV     R1,30(SP)       ;REMAINDER ON THE STACK
6406  027150  010366  000032                   MOV     R3,32(SP)       ;QUOTIENT ON THE STACK
6407  027154  104413                           RESREG                  ;RESTORE R0 - R5
6408  027156  012616                           MOV     (SP)+,(SP)      ;MOVE RETURN UP THE STACK
6409  027160  000207                           RTS     PC
6410
6411                                    ;       DIVISION UTILITY SUBROUTINE
6412                                    ;       R0-R1-R2-R3=DIVIDEND
6413                                    ;       R4-R5=DIVISOR
6414                                    ;       R0-R1=REMAINDER AFTER DIVISION
6415                                    ;       R2-R3=QUOTIENT AFTER DIVISION
6416                                    ;       ENTER WITH JSR  PC,M.DPID
6417                                    ;
6418  027162  012746  000040           M.DPID: MOV     #40,-(SP)       ;COUNTER FOR DIVISION CYCLES
6419  027166  010446                           MOV     R4,-(SP)        ;HIGH ORDER
6420  027170  010546                           MOV     R5,-(SP)        ;LOW ORDER DIVISOR TO THE STACK
6421  027172  005466  000002                   NEG     2(SP)           ;FORM NEGATIVE
6422  027176  005416                           NEG     @SP             ;VERSION OF THE DIVISOR
6423  027200  005666  000002                   SBC     2(SP)
6424  027204  061601                           ADD     @SP,R1
6425  027206  005500                           ADC     R0              ;PERFORM THE INITIAL SUBTRACTION
6426  027210  066600  000002                   ADD     2(SP),R0
6427  027214  103445                           BCS     M.DP50          ;IF CARRY THEN OVERFLOW HAS OCCURRED
6428  027216  005046                           CLR     -(SP)           ;THIS IS A LONGER LASTING CARRY BIT
6429  027220  006103           M.DP40: ROL     R3
6430  027222  006102                   ROL     R2
6431  027224  006101                   ROL     R1
6432  027226  006100                   ROL     R0
6433  027230  005716                   TST     @SP             ;TEST "CARRY" INDICATOR
6434  027232  001410                   BEQ     M.DP41          ;IF NO "CARRY" THEN ADD ELSE SUBTRACT
6435  027234  005016                   CLR     @SP             ;CLEAR UP FOR NEXT TIME
```

```
6436  027236  066601  000002                  ADD     2(SP),R1
6437  027242  005500                          ADC     R0              ;ADD -(DIVISOR)
6438  027244  005516                          ADC     @SP     ;  I    ;SET "CARRY"
6439  027246  066600  000004                  ADD     4(SP),R0;<-
6440  027252  000404                          BR      M.DP42
6441  027254  060501              M.DP41:     ADD     R5,R1
6442  027256  005500                          ADC     R0              ;ADD +(DIVISOR)
6443  027260  005516                          ADC     @SP     ;  I    ;SET "CARRY"
6444  027262  060400                          ADD     R4,R0   ;<-
6445  027264  005516              M.DP42:     ADC     @SP             ;SET "CARRY"
6446  027266  005716                          TST     @SP             ;TEST THE UPDATE INDICATOR
6447  027270  001401                          BEQ     .+4     ;->     ;IF ZERO FORGET IT
6448  027272  005203                          INC     R3      ;  I    ;NO CARRY POSSIBLE HERE
6449  027274  005366  000006                  DEC     6(SP)   ;<-     ;DECREMENT COUNTER
6450  027300  003347                          BGT     M.DP40          ;BRANCH IF MORE TO DO
6451  027302  006003                          ROR     R3
6452  027304  103404                          BCS     M.DP44
6453  027306  060501                          ADD     R5,R1
6454  027310  005500                          ADC     R0
6455  027312  060400                          ADD     R4,R0
6456  027314  000241                          CLC
6457  027316  006103              M.DP44:     ROL     R3
6458  027320  062706  000010                  ADD     #10,SP          ;ADJUST STACK BY 4 WORDS
6459  027324  000242                          CLV
6460  027326  000207                          RTS     PC
6461  027330  062706  000006      M.DP50:     ADD     #6,SP
6462  027334  000262                          SEV
6463  027336  000207                          RTS     PC
6464
6465
6466                                ;ROUTINE TO REPLACE LEADING ZEROS IN A NUMERIC STRING WITH SPACES
6467                                ;CALL
6468                                ;       MOV     #ADR,-(SP)      ;ADDRESS OF NUMBER (IN ASCII)
6469                                ;       JSR     R5,REPLZ
6470                                ;       .WORD   N               ;'N' IS NUMBER OF DIGITS TO BE TYPED
6471
6472  027340  010046              REPLZ:  MOV     R0,-(SP)        ;SAVE R0
6473  027342  012746  000012              MOV     #10.,-(SP)      ;MAXIMUM NUMBER OF DIGITS TO BE TYPED
6474  027346  162516                      SUB     (R5)+,(SP)      ;SUBTACT DIGITS TO FORM INDEX
6475  027350  016600  000006              MOV     6(SP),R0        ;ADDRESS OF NUMBER TO R0
6476  027354  122710  000060      1$:     CMPB    #'0,(R0)        ;BYTE EQUAL TO ASCII '0' ?
6477  027360  001004                      BNE     2$              ;BR IF NOT
6478  027362  112710  000040              MOVB    #40,(R0)        ;REPLACE THE ZERO WITH A SPACE
6479  027366  005200                      INC     R0              ;INCREMENT THE BYTE ADDRESS
6480  027370  000771                      BR      1$              ;GO BACK AND LOOK FOR MORE LEADING ZEROS
6481  027372  105710              2$:     TSTB    (R0)            ;SEE IF ZERO BYTE TERMINATOR
6482  027374  001003                      BNE     3$              ;BR IF NOT
6483  027376  005300                      DEC     R0              ;BACKUP STRING POINTER
6484  027400  112710  000060              MOVB    #'0,(R0)        ;PUT A ZERO BACK IN
6485  027404  016637  000006  027420  3$: MOV     6(SP),4$        ;PUT ADDRESS IN LOCATION FOR TYPEOUT
6486  027412  062637  027420              ADD     (SP)+,4$        ;BEGINNING OF SIGNIFICANT DIGITS
6487  027416  104401                      TYPE                    ;TYPE THE NUMBER
6488  027420  000000              4$:     .WORD   0               ;ADDRESS OF NUMBER
6489  027422  012600                      MOV     (SP)+,R0        ;RESTORE R0
6490  027424  012616                      MOV     (SP)+,(SP)      ;MOVE RETURN ADDRESS
6491  027426  000205                      RTS     R5              ;RETURN
```

```
6492
6493                                    ;TYPE NUMERICAL ASCIZ STRING SUPRESS LEADING ZEROS
6494
6495                                    ;CALL
6496                                    ;       MOV     #NUMADR,-(SP)   ;FIRST ADDRESS OF ASCIZ STRING
6497                                    ;       JSR     PC,$SUPRS
6498
6499  027430  010046                    $SUPRS: MOV     R0,-(SP)        ;SAVE R0
6500  027432  016600  000004                    MOV     4(SP),R0        ;PICKUP THE POINTER
6501  027436  105710                    1$:     TSTB    (R0)            ;TERMINATOR ?
6502  027440  001403                            BEQ     2$              ;BR IF YES
6503  027442  122720  000060                    CMPB    #'0,(R0)+       ;IS THIS AN ASCII '0' ?
6504  027446  001773                            BEQ     1$              ;BR IF YES
6505  027450  005300                    2$:     DEC     R0              ;BACKUP BY '1'
6506  027452  010037  027460                    MOV     R0,3$           ;SAVE FOR TYPING
6507  027456  104414                            DISPLY                  ;GO PRINT
6508  027460  000000                    3$:     .WORD   0               ;ASCIZ POINTER GOES HERE
6509  027462  012600                            MOV     (SP)+,R0        ;RESTORE R0
6510  027464  012616                            MOV     (SP)+,(SP)      ;RESTORE THE STACK
6511  027466  000207                            RTS     PC              ;RETURN
6512
6513                                    ;ROUTINE TO TYPE AT PRIORITY 4
6514
6515  027470  013746  177776            TYPRI4: MOV     @#PS,-(SP)      ;SAVE THE PRESENT STATUS
6516  027474  012737  000200  177776            MOV     #200,@#PS       ;CHANGE THE PRIORITY TO 4
6517  027502  012537  027512                    MOV     (R5)+,1$        ;MESSAGE ADDRESS
6518  027506  004737  031074                    JSR     PC,$TYPE        ;TYPE THE MESSAGE
6519  027512  000000                    1$:     .WORD   0               ;MESSAGE ADDRESS GOES HERE
6520  027514  000205                            RTS     R5              ;RETURN
6521
6522                                    ;ROUTINE TO TYPE ERRORS
6523                                    ;CALL
6524                                    ;       DISPLY                  ;MUST DEFINED IN 'TRAP' TABLE
6525                                    ;       MESADR                  ;ADDRESS OF MESSAGE
6526                                    ;       RETURN
6527
6528  027516  032777  020000  151414  $DSPLY: BIT     #BIT13,@SWR     ;INHIBIT ERROR TYPEOUT ?
6529  027524  001004                            BNE     1$              ;BR IF YES
6530  027526  005037  177776                    CLR     @#PS            ;SET PRIORITY TO ZERO
6531  027532  000137  031074                    JMP     $TYPE           ;TYPE THE MESSAGE
6532  027536  062716  000002            1$:     ADD     #2,(SP)         ;INCREMENT THE RETURN
6533  027542  000002                            RTI                     ;RETURN
6534
6535                                    ;THIS ROUTINE IS USED TO CHECK IF AN
6536                                    ;ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.
6537                                    ;CALL
6538                                    ;       MOV     #ADR,R1         ;ADDRESS OF ASCII CHARACTER
6539                                    ;       JSR     R5,CK.OCT       ;CHECK THE CHARACTER
6540                                    ;       RETURN1                 ;CHARACTER IS NOT BETWEEN 0-7
6541                                    ;       RETURN2                 ;CHARACTER IS IN R2 AS A
6542                                    ;                               ;OCTAL DIGIT
6543
6544  027544  121127  000060            CK.OCT: CMPB    (R1),#'0        ;LESS THAN ZERO?
6545  027550  103407                            BLO     1$              ;YES -- BRANCH
6546  027552  121127  000067                    CMPB    (R1),#'7        ;GREATER THAN SEVEN?
6547  027556  101004                            BHI     1$              ;YES -- BRANCH
```

```
6548  027560  111102                  MOVB    (R1),R2         ;GET THE CHARACTER
6549  027562  042702  177770          BIC     #^C7,R2         ;STRIP AWAY THE ASCII
6550  027566  005725                  TST     (R5)+           ;ADJUST FOR RETURN
6551  027570  000205          1$:     RTS     R5              ;RETURN
6552
6553                                  ;THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER
6554                                  ;AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.
6555                                  ;CALL
6556                                  ;       MOV     #ADR,R1         ;ADDRESS OF ASCII CHARACTER
6557                                  ;       JSR     R5,CK.DEC       ;CHECK THE CHARACTER
6558                                  ;       RETURN1                 ;NOT BETWEEN 0 AND 9
6559                                  ;       RETURN2                 ;BETWEEN 0 AND 9
6560                                  ;                               ;R2 = DIGIT
6561
6562  027572  121127  000060  CK.DEC: CMPB    (R1),#'0        ;LESS THAN ZERO?
6563  027576  103407                  BLO     1$              ;YES -- BRANCH
6564  027600  121127  000071          CMPB    (R1),#'9        ;GREATER THAN NINE?
6565  027604  101004                  BHI     1$              ;YES -- BRANCH
6566  027606  111102                  MOVB    (R1),R2         ;GET THE CHARACTER
6567  027610  042702  000060          BIC     #'0,R2          ;STRIP AWAY THE ASCII
6568  027614  005725                  TST     (R5)+           ;ADJUST FOR RETURN
6569  027616  000205          1$:     RTS     R5              ;RETURN
6570
6571                                  ;THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO
6572                                  ;DETERMINE WHAT IT IS.
6573                                  ;CALL
6574                                  ;       MOV     #ADR,R1         ;ADDRESS OF ASCII CHARACTER
6575                                  ;       JSR     R5,CK.CHR       ;CHECK CHARACTER
6576                                  ;       RETURN  ADR1            ;UNKNOWN CHARACTER
6577                                  ;       RETURN  ADR2            ;CARRIAGE RETURN * (R1)=ADR+1
6578                                  ;       RETURN  ADR3            ;COMMA * (R1)=ADR+1
6579                                  ;       RETURN  ADR4            ;PERIOD * (R1)=ADR+1
6580                                  ;       RETURN  ADR5            ;DIGIT BETWEEN 0 AND 7.
6581                                  ;       RETURN  ADR6            ;DIGIT BETWEEN 8 AND 9.
6582                                  ;                               ;R2 = DIGIT * (R1)=ADR+1
6583
6584  027620  105711          CK.CHR: TSTB    (R1)            ;"CARRIAGE RETURN"?
6585  027622  001417                  BEQ     3$              ;YES -- BRANCH
6586  027624  121127  000054          CMPB    (R1),#',        ;"COMMA"?
6587  027630  001413                  BEQ     2$              ;YES -- BRANCH
6588  027632  121127  000056          CMPB    (R1),#'.        ;"PERIOD"?
6589  027636  001407                  BEQ     1$              ;YES -- BRANCH
6590  027640  004537  027572          JSR     R5,CK.DEC       ;"DIGIT"?
6591  027644  000410                  BR      4$              ;NO -- BRANCH
6592  027646  004537  027544          JSR     R5,CK.OCT       ;OCTAL ?
6593  027652  005725                  TST     (R5)+           ;DIGIT BETWEEN 8-9
6594  027654  005725                  TST     (R5)+           ;DIGIT BETWEEN 0-7
6595  027656  005725          1$:     TST     (R5)+           ;PERIOD
6596  027660  005725          2$:     TST     (R5)+           ;COMMA
6597  027662  005725          3$:     TST     (R5)+           ;CARRIAGE RETURN
6598  027664  005201                  INC     R1              ;MOVE POINTER TO NEXT CHARACTER
6599  027666  011505          4$:     MOV     (R5),R5         ;UNKNOWN CHARACTER
6600  027670  000205                  RTS     R5              ;RETURN
6601
6602                                  ;THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL
6603                                  ;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

027672  010446          CK.DIG: MOV     R4,-(SP)        ;SAVE R4
027674  010346                  MOV     R3,-(SP)        ;SAVE R3
027676  010246                  MOV     R2,-(SP)        ;SAVE THE MAX. SIZE ON THE STACK
027700  005002                  CLR     R2              ;START WITH 0
027702  005003                  CLR     R3
027704  005004                  CLR     R4
027706  004537  027620          JSR     R5,CK.CHR       ;CHECK ONE CHARACTER
027712  030006                  6$                      ;ILLEGAL CHARACTER
027714  030014                  9$                      ;CARRIAGE RETURN
027716  030006                  6$                      ;"-"
027720  030010                  7$                      ;","
027722  027726                  1$                      ;DIGIT 0-7
027724  027726                  1$                      ;DIGIT 8-9
027726  062705  000004  1$:     ADD     #4,R5           ;STEP RETURN POINTER PAST "CR" & "PERIOD" RETURNS
027732  006303          2$:     ASL     R3              ;INPUT NUMBER *2
027734  010346                  MOV     R3,-(SP)        ;SAVE *2
027736  006303                  ASL     R3              ;*4
027740  006303                  ASL     R3              ;*8
027742  062603                  ADD     (SP)+,R3        ;(*2)+(*8) = *10
027744  060203                  ADD     R2,R3           ;UPDATE THE INPUT NUMBER
027746  004537  027620          JSR     R5,CK.CHR       ;CHECK ONE CHARACTER
027752  030012                  8$                      ;ILLEGAL CHARACTER
027754  027776                  5$                      ;CARRIAGE RETURN
027756  027774                  4$                      ;"-"
027760  027766                  3$                      ;","
027762  027732                  2$                      ;DIGIT 0-7
027764  027732                  2$                      ;DIGIT 8-9
027766  105711          3$:     TSTB    (R1)            ;DOES A "CR" FOLLOW THE "PERIOD"
027770  001010                  BNE     8$              ;BR IF NOT
027772  005724                  TST     (R4)+           ;INCREMENT THE RETURN
027774  005724          4$:     TST     (R4)+           ;INCREMENT THE RETURN
027776  005724          5$:     TST     (R4)+           ;INCREMENT THE RETURN
030000  020316                  CMP     R3,(SP)         ;CHECK THE MAGNITUDE OF THE NUMBER
030002  101004                  BHI     9$              ;BR IF ENTERED NUMBER TOO LARGE
030004  000402                  BR      8$              ;BYPASS INCREMENT
030006  005725          6$:     TST     (R5)+           ;INCREMENT RETURN PAST INVALID RETURN
030010  005725          7$:     TST     (R5)+           ;INCREMENT RETURN
030012  060405          8$:     ADD     R4,R5           ;SETUP RETURN POINTER
030014  010302          9$:     MOV     R3,R2           ;ENTERED VALUE
030016  005726                  TST     (SP)+           ;CLEAN MAX. SIZE OFF OF STACK
030020  012603                  MOV     (SP)+,R3        ;RESTORE R3
030022  012604                  MOV     (SP)+,R4        ;RESTORE R4
030024  011505                  MOV     (R5),R5         ;GET RETURN ADDRESS
030026  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

030030  016637  000002  030054  $SB2D:  MOV     2(SP),1$        ;SAVE THE BINARY NUMBER
030036  012746  030054                  MOV     #1$,-(SP)       ;SET THE POINTER
030042  004737  032622                  JSR     PC,$DB2D        ;CALL THE DOUBLE LENGTH CONVERT
030046  012666  000002                  MOV     (SP)+,2(SP)     ;PICKUP THE POINTER
030052  000207                          RTS     PC              ;RETURN
030054  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

030060  016637  000002  030104  $SB2O:  MOV     2(SP),1$        ;SAVE THE BINARY NUMBER
030066  012746  030104                  MOV     #1$,-(SP)       ;SET THE POINTER
030072  004737  033016                  JSR     PC,$DB2O        ;CALL THE DOUBLE LENGTH CONVERT
030076  012666  000002                  MOV     (SP)+,2(SP)     ;PICKUP THE POINTER
030102  000207                          RTS     PC              ;RETURN
030104  000000  000000          1$:     .WORD   0,0

                                        ;KEYBOARD INTERRUPT INITIALIZATION ROUTINE
                                        ;CALL
                                        ;       JSR     PC,$TKINT
                                        ;       RETURN

030110  012737  030140  000060  $TKINT: MOV     #$TKSRV,TKVEC   ;SETUP VECTOR
030116  012737  000240  000062          MOV     #PR5,TKVEC+2    ;PRIORITY TO 5
030124  005777  151016                  TST     @$TKB           ;CLEAR THE BUFFER
030130  012777  000100  151006          MOV     #BIT06,@$TKS    ;SET INTERRUPT ENABLE
030136  000207                          RTS     PC              ;RETURN

                                        ;KEYBOARD INTERRUPT SERVICE ROUTINE
                                        ;CALL
                                        ;       ENTER VIA INTERRUPT

030140  104410          $TKSRV: RDCHR                           ;READ THE KEYBOARD
030142  112637  030270                  MOVB    (SP)+,5$        ;GET THE CHARACTER
030146  023727  030270  000003          CMP     5$,#3           ;'CONTROL C' ?
030154  001012                          BNE     1$              ;BR IF NOT
030156  104401  001165                  TYPE    ,$CRLF          ;CR-LF
030162  104401  030562                  TYPE    ,$CNTLC         ;'↑C'
030166  012737  177777  001262          MOV     #-1,CFLAG       ;SET THE 'CONTROL C' FLAG
```

```
6716  030174  005077  150744                      CLR     @$TKS           ;CLEAR THE TTY INTERRUPT
6717  030200  000432                              BR      4$              ;EXIT
6718  030202  023727  001140  000176  1$:         CMP     SWR,#SWREG      ;SOFTWARE SWITCH REGISTER IN USE ?
6719  030210  001024                              BNE     3$              ;BR IF NOT
6720  030212  023727  030270  000007              CMP     5$,#7           ;'CONTROL G' ?
6721  030220  001020                              BNE     3$              ;BR IF NOT
6722  030222  104401  001165                      TYPE    ,$CRLF          ;CR-LF
6723  030226  104401  032375                      TYPE    ,$CNTLG         ;'↑G'
6724  030232  013746  177776                      MOV     PS,-(SP)        ;PUT THE STATUS WORD ON THE STACK
6725  030236  012746  030252                      MOV     #2$,-(SP)       ;RETURN ADDRESS
6726  030242  005077  150676                      CLR     @$TKS           ;CLEAR THE TTY INTERRUPT ENABLE
6727  030246  000137  032036                      JMP     $GTSWR          ;GET THE SWITCH REGISTER ENTRY
6728  030252  012777  000100  150664  2$:         MOV     #100,@$TKS      ;ENABLE TTY KEYBOARD INTERRUPT
6729  030260  000402                              BR      4$              ;EXIT
6730  030262  104401  030270          3$:         TYPE    ,5$             ;ECHO THE CHARACTER
6731  030266  000002                  4$:         RTI                     ;RETURN
6732
6733  030270  000000                  5$:         .WORD   0               ;ENTERED CHARACTER
6734
6735                                  ;THIS ROUTINE WILL INPUT A STRING FROM THE TTY
6736                                  ;CALL:
6737                                  ;       RDLIN                   ;;INPUT A STRING FROM THE TTY
6738                                  ;       RETURN HERE             ;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
6739                                  ;                               ;TERMINATOR WILL BE A BYTE OF ALL 0'S
6740
6741  030272  010346                  $RDLIN: MOV     R3,-(SP)        ;SAVE R3
6742  030274  005046                          CLR     -(SP)           ;CLEAR THE RUBOUT KEY
6743  030276  012703  030550          1$:     MOV     #$TTYIN,R3      ;GET ADDRESS
6744  030302  022703  030562          2$:     CMP     #$TTYIN+10.,R3  ;BUFFER FULL?
6745  030306  101467                          BLOS    4$              ;BR IF YES
6746  030310  104410                          RDCHR                   ;GO READ ONE CHARACTER FROM THE TTY
6747  030312  112613                          MOVB    (SP)+,(R3)      ;GET CHARACTER
6748  030314  122713  000177                  CMPB    #177,(R3)       ;IS IT A RUBOUT
6749  030320  001022                          BNE     5$              ;BR IF NO
6750  030322  005716                          TST     (SP)            ;IS THIS THE FIRST RUBOUT?
6751  030324  001007                          BNE     6$              ;BR IF NO
6752  030326  112737  000134  030546          MOVB    #'\,9$          ;TYPE A BACK SLASH
6753  030334  104401  030546                  TYPE    ,9$
6754  030340  012716  177777                  MOV     #-1,(SP)        ;SET THE RUBOUT KEY
6755  030344  005303                  6$:     DEC     R3              ;BACKUP BY ONE
6756  030346  020327  030550                  CMP     R3,#$TTYIN      ;STACK EMPTY?
6757  030352  103445                          BLO     4$              ;BR IF YES
6758  030354  111337  030546                  MOVB    (R3),9$         ;SETUP TO TYPEOUT THE DELETED CHAR.
6759  030360  104401  030546                  TYPE    ,9$             ;GO TYPE
6760  030364  000746                          BR      2$              ;GO READ ANOTHER CHAR.
6761  030366  005716                  5$:     TST     (SP)            ;RUBOUT KEY SET?
6762  030370  001406                          BEQ     7$              ;BR IF NO
6763  030372  112737  000134  030546          MOVB    #'\,9$          ;TYPE A BACK SLASH
6764  030400  104401  030546                  TYPE    ,9$
6765  030404  005016                          CLR     (SP)            ;CLEAR THE RUBOUT KEY
6766  030406  122713  000025          7$:     CMPB    #25,(R3)        ;IS CHARACTER A CTRL U?
6767  030412  001003                          BNE     10$             ;BR IF NO
6768  030414  104401  032370                  TYPE    ,$CNTLU         ;TYPE A CONTROL "U"
6769  030420  000726                          BR      1$              ;GO START OVER
6770  030422  122713  000003          10$:    CMPB    #3,(R3)         ;IS CHARACTER A CTRL C ?
6771  030426  001006                          BNE     8$              ;BR IF NOT
```

```
6772  030430  012737  177777  001262          MOV     #-1,CFLAG       ;SET CNTRL C FLAG
6773  030436  104401  030562                  TYPE    ,$CNTLC         ;ECHO IT
6774  030442  000427                          BR      1$              ;EXIT
6775  030444  122713  000012          8$:     CMPB    #12,(R3)        ;IS CHARACTER A "LF"?
6776  030450  001011                          BNE     3$              ;BRANCH IF NO
6777  030452  105013                          CLRB    (R3)            ;CLEAR THE CHARACTER
6778  030454  104401  001165                  TYPE    ,$CRLF          ;TYPE A "CR" & "LF"
6779  030460  104401  030550                  TYPE    ,$TTYIN         ;TYPE THE INPUT STRING
6780  030464  000706                          BR      2$              ;GO PICKUP ANOTHER CHACTER
6781  030466  104401  001164          4$:     TYPE    ,$QUES          ;TYPE A '?'
6782  030472  000701                          BR      1$              ;CLEAR THE BUFFER AND LOOP
6783  030474  111337  030546          3$:     MOVB    (R3),9$         ;ECHO THE CHARACTER
6784  030500  104401  030546                  TYPE    ,9$
6785  030504  122723  000015                  CMPB    #15,(R3)+       ;CHECK FOR RETURN
6786  030510  001274                          BNE     2$              ;LOOP IF NOT RETURN
6787  030512  105063  177777                  CLRB    -1(R3)          ;CLEAR RETURN (THE 15)
6788  030516  104401  001166                  TYPE    ,$LF            ;TYPE A LINE FEED
6789  030522  005726                  11$:    TST     (SP)+           ;CLEAN RUBOUT KEY FROM THE STACK
6790  030524  012603                          MOV     (SP)+,R3        ;RESTORE R3
6791  030526  011646                          MOV     (SP),-(SP)      ;ADJUST THE STACK AND PUT ADDRESS OF THE
6792  030530  016666  000004  000002          MOV     4(SP),2(SP)     ;       FIRST ASCII CHARACTER ON IT
6793  030536  012766  030550  000004          MOV     #$TTYIN,4(SP)
6794  030544  000002                          RTI                     ;RETURN
6795  030546     000                  9$:     .BYTE   0               ;STORAGE FOR ASCII CHAR. TO TYPE
6796  030547     000                          .BYTE   0               ;TERMINATOR
6797  030550  000012                  $TTYIN: .BLKB   10.             ;RESERVE 10 BYTES FOR TTY INPUT
6798  030562  041536  005015     000  $CNTLC: .ASCIZ  /^C/<CR><LF>    ;CONTROL "C"
6799
6800          030570                          .EVEN
6801
6802                                  ;;**************************************************************
6803
6804                                  .SBTTL  MACRO ROUTINES
6805
6806                                  .SBTTL  ERROR HANDLER ROUTINE
6807
6808                                  ;;**************************************************************
6809                                  ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
6810                                  ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
6811                                  ;*AND GO TO $ERRTYP ON ERROR
6812                                  ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
6813                                  ;*SW15=1         HALT ON ERROR
6814                                  ;*SW13=1         INHIBIT ERROR TYPEOUTS
6815                                  ;*SW10=1         BELL ON ERROR
6816                                  ;*CALL
6817                                  ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
6818
6819  030570                          $ERROR:
6820  030570  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
6821  030572  010337  001244                  MOV     R3,ATTN         ;SAVE THE ATTENTION REGISTER CONTENTS
6822  030576  010137  001242                  MOV     R1,DRIVE        ;DRIVE NUMBER
6823  030602  032777  020000  150330          BIT     #SW13,@SWR      ;INHIBIT PRINTOUTS ?
6824  030610  001002                          BNE     .+6             ;BR IF YES
6825  030612  004737  023510                  JSR     PC,$TIME        ;TYPE THE TIME
6826  030616  105237  001103          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
6827  030622  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
```

```
6828  030624  013777  001102  150310          MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
6829  030632  032777  002000  150300          BIT     #BIT10,@SWR     ;;BELL ON ERROR?
6830  030640  001402                          BEQ     1$              ;;NO - SKIP
6831  030642  104401  001160                  TYPE    ,$BELL          ;;RING BELL
6832  030646  005237  001112          1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
6833  030652  011637  001116                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
6834  030656  162737  000002  001116          SUB     #2,$ERRPC
6835  030664  117737  150226  001114          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
6836  030672  032777  020000  150240          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
6837  030700  001004                          BNE     20$             ;;SKIP TYPEOUTS
6838  030702  004737  030740                  JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
6839  030706  104401  001165                  TYPE    ,$CRLF
6840  030712                          20$:
6841  030712  005777  150222          2$:     TST     @SWR            ;;HALT ON ERROR
6842  030716  100002                          BPL     3$              ;;SKIP IF CONTINUE
6843  030720  000000                          HALT                    ;;HALT ON ERROR!
6844  030722  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
6845  030724                          3$:
6846  030724  023737  000042  000046          CMP     @#42,@#46       ;ARE WE IN ACT-11 AUTO MODE?
6847  030732  001001                          BNE     .+4             ;BRANCH IF NOT
6848  030734  000000                          HALT                    ;HALT ON ERROR IF ACT AUTO MODE
6849  030736  000002                          RTI                     ;RETURN
6850
6851                                  .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
6852
6853                                  ;;**************************************************************
6854                                  ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
6855                                  ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
6856                                  ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
6857
6858  030740                          $ERRTYP:
6859  030740  104401  001165                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
6860  030744  010046                          MOV     R0,-(SP)        ;;SAVE R0
6861  030746  005000                          CLR     R0              ;;PICKUP THE ITEM INDEX
6862  030750  153700  001114                  BISB    @#$ITEMB,R0
6863  030754  001004                          BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
6864                                                                  ;;TYPE THE PC OF THE ERROR
6865  030756  013746  001116                  MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
6866                                                                  ;;ERROR ADDRESS
6867  030762  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
6868  030764  000426                          BR      6$              ;;GET OUT
6869  030766  005300                  1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
6870  030770  006300                          ASL     R0              ;;      WORK FOR THE ERROR TABLE
6871  030772  006300                          ASL     R0
6872  030774  006300                          ASL     R0
6873  030776  062700  004026                  ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
6874  031002  012037  031012                  MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
6875  031006  001404                          BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
6876  031010  104401                          TYPE                    ;;TYPE THE "ERROR MESSAGE"
6877  031012  000000                  2$:     .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
6878  031014  104401  001165                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
6879  031020  012037  031030          3$:     MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
6880  031024  001404                          BEQ     5$              ;;SKIP TYPEOUT IF 0
6881  031026  104401                          TYPE                    ;;TYPE THE "DATA HEADER"
6882  031030  000000                  4$:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
6883  031032  104401  001165                  TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
```

```
6884  031036  011000                5$:     MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
6885  031040  001004                        BNE     7$              ;;GO TYPE THE DATA
6886  031042  012600                6$:     MOV     (SP)+,R0        ;;RESTORE R0
6887  031044  104401  001165                TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
6888  031050  000207                        RTS     PC              ;;RETURN
6889  031052                        7$:
6890  031052  013046                        MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
6891  031054  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
6892  031056  005710                        TST     (R0)            ;;IS THERE ANOTHER NUMBER?
6893  031060  001770                        BEQ     6$              ;;BR IF NO
6894  031062  104401  031070                TYPE    ,8$             ;;TYPE TWO(2) SPACES
6895  031066  000771                        BR      7$              ;;LOOP
6896  031070  020040    000         8$:     .ASCIZ  /  /            ;;TWO(2) SPACES
6897          031074                        .EVEN
6898
6899                                .SBTTL  TYPE ROUTINE
6900
6901                                ;;**************************************************************
6902                                ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TEPMINATE WITH A 0 BYTE.
6903                                ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
6904                                ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
6905                                ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
6906                                ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
6907                                ;*
6908                                ;*CALL:
6909                                ;*1) USING A TRAP INSTRUCTION
6910                                ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
6911                                ;*OR
6912                                ;*      TYPE
6913                                ;*      MESADR
6914                                ;*
6915
6916  031074  105737  001157        $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
6917  031100  100002                        BPL     1$              ;;BR IF YES
6918  031102  000000                        HALT                    ;;HALT HERE IF NO TERMINAL
6919  031104  000407                        BR      3$              ;;LEAVE
6920  031106  010046                1$:     MOV     R0,-(SP)        ;;SAVE R0
6921  031110  017600  000002                MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
6922  031114  112046                2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
6923  031116  001005                        BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
6924  031120  005726                        TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
6925  031122  012600                60$:    MOV     (SP)+,R0        ;;RESTORE R0
6926  031124  062716  000002        3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
6927  031130  000002                        RTI                     ;;RETURN
6928  031132  122716  000011        4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
6929  031136  001430                        BEQ     8$
6930  031140  122716  000200                CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
6931  031144  001006                        BNE     5$
6932  031146  005726                        TST     (SP)+           ;;POP  <CR><LF> EQUIV
6933  031150  104401                        TYPE                    ;;TYPE A CR AND LF
6934  031152  001165                        $CRLF
6935  031154  105037  031310                CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
6936  031160  000755                        BR      2$              ;;GET NEXT CHARACTER
6937  031162  004737  031244        5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
6938  031166  123726  001156        6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
6939  031172  001350                        BNE     2$              ;;IF NO GO GET NEXT CHAR.
```

```
6940  031174  013746  001154          MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
6941                                                          ;;AND THE NULL CHAR.
6942  031200  105366  000001  7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
6943  031204  002770                  BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
6944  031206  004737  031244          JSR     PC,$TYPEC       ;;GO TYPE A NULL
6945  031212  105337  031310          DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
6946  031216  000770                  BR      7$              ;;LOOP
6947
6948                                  ;HORIZONTAL TAB PROCESSOR
6949
6950  031220  112716  000040  8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
6951  031224  004737  031244  9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
6952  031230  132737  000007  031310  BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
6953  031236  001372                  BNE     9$              ;;TAB STOP
6954  031240  005726                  TST     (SP)+           ;;POP SPACE OFF STACK
6955  031242  000724                  BR      2$              ;;GET NEXT CHARACTER
6956  031244  105777  147700  $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
6957  031250  100375                  BPL     $TYPEC
6958  031252  116677  000002  147672  MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
6959  031260  122766  000015  000002  CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
6960  031266  001003                  BNE     1$              ;;BRANCH IF NO
6961  031270  105037  031310          CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
6962  031274  000406                  BR      $TYPEX          ;;EXIT
6963  031276  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
6964  031304  001402                  BEQ     $TYPEX          ;;BRANCH IF YES
6965  031306  105227                  INCB    (PC)+           ;;COUNT THE CHARACTER
6966  031310  000000          $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
6967  031312  000207          $TYPEX: RTS     PC
6968
6969
6970                                  .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
6971
6972                                  ;;*****************************************************************
6973                                  ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
6974                                  ;*OCTAL (ASCII) NUMBER AND TYPE IT.
6975                                  ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
6976                                  ;*CALL:
6977                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
6978                                  ;*      TYPOS                   ;;CALL FOR TYPEOUT
6979                                  ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
6980                                  ;*      .BYTE   M               ;;M=1 OR 0
6981                                  ;*                                      ;;1=TYPE LEADING ZEROS
6982                                  ;*                                      ;;0=SUPPRESS LEADING ZEROS
6983                                  ;*
6984                                  ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
6985                                  ;*$TYPOS OR $TYPOC
6986                                  ;*CALL:
6987                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
6988                                  ;*      TYPON                   ;;CALL FOR TYPEOUT
6989                                  ;*
6990                                  ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
6991                                  ;*CALL:
6992                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
6993                                  ;*      TYPOC                   ;;CALL FOR TYPEOUT
6994
6995  031314  017646  000000  $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
```

```
 6996  031320  116637  000001  031537          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
 6997  031326  112637  031541                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
 6998  031332  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
 6999  031336  000406                          BR      $TYPON
 7000  031340  112737  000001  031537  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
 7001  031346  112737  000006  031541          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
 7002  031354  112737  000005  031536  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
 7003  031362  010346                          MOV     R3,-(SP)        ;;SAVE R3
 7004  031364  010446                          MOV     R4,-(SP)        ;;SAVE R4
 7005  031366  010546                          MOV     R5,-(SP)        ;;SAVE R5
 7006  031370  113704  031541                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
 7007  031374  005404                          NEG     R4
 7008  031376  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
 7009  031402  110437  031540                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
 7010  031406  113704  031537                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
 7011  031412  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
 7012  031416  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
 7013  031420  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
 7014  031422  000404                          BR      3$              ;;GO DO MSB
 7015  031424  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
 7016  031426  006105                          ROL     R5
 7017  031430  006105                          ROL     R5
 7018  031432  010503                          MOV     R5,R3
 7019  031434  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
 7020  031436  105337  031540                  DECB    $OMODE          ;;TYPE THIS DIGIT?
 7021  031442  100016                          BPL     7$              ;;BR IF NO
 7022  031444  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
 7023  031450  001002                          BNE     4$              ;;TEST FOR 0
 7024  031452  005704                          TST     R4              ;;SUPPRESS THIS 0?
 7025  031454  001403                          BEQ     5$              ;;BR IF YES
 7026  031456  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
 7027  031460  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
 7029  031464  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
 7029  031470  110337  031534                  MOVB    R3,8$           ;;SAVE FOR TYPING
 7030  031474  104401  031534                  TYPE    ,8$             ;;GO TYPE THIS DIGIT
 7031  031500  105337  031536          7$:     DECB    $OCNT           ;;COUNT BY 1
 7032  031504  003347                          BGT     2$              ;;BR IF MORE TO DO
 7033  031506  002402                          BLT     6$              ;;BR IF DONE
 7034  031510  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
 7035  031512  000744                          BR      2$              ;;GO DO THE LAST DIGIT
 7036  031514  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
 7037  031516  012604                          MOV     (SP)+,R4        ;;RESTORE R4
 7038  031520  012603                          MOV     (SP)+,R3        ;;RESTORE R3
 7039  031522  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
 7040  031530  012616                          MOV     (SP)+,(SP)
 7041  031532  000002                          RTI                     ;;RETURN
 7042  031534     000                  8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
 7043  031535     000                          .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
 7044  031536     000          $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
 7045  031537     000          $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
 7046  031540  000000          $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
 7047
 7048                          .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
 7049
 7050                          ;;**************************************************************
 7051                          ;*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

031542                                  $TYPDS:
031542  010046                         MOV     R0,-(SP)        ;;PUSH R0 ON STACK
031544  010146                         MOV     R1,-(SP)        ;;PUSH R1 ON STACK
031546  010246                         MOV     R2,-(SP)        ;;PUSH R2 ON STACK
031550  010346                         MOV     R3,-(SP)        ;;PUSH R3 ON STACK
031552  010546                         MOV     R5,-(SP)        ;;PUSH R5 ON STACK
031554  012746  020200                 MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
031560  016605  000020                 MOV     20(SP),R5       ;;GET THE INPUT NUMBER
031564  100004                         BPL     1$              ;;BR IF INPUT IS POS.
031566  005405                         NEG     R5              ;;MAKE THE BINARY NUMBER POS.
031570  112766  000055  000001         MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
031576  005000                 1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
031600  012703  031756                 MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
031604  112723  000040                 MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
031610  005002                 2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
031612  016001  031746                 MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
031616  160105                 3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
031620  002402                         BLT     4$              ;;BR IF DONE
031622  005202                         INC     R2              ;;INCREASE THE BCD DIGIT BY 1
031624  000774                         BR      3$
031626  060105                 4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
031630  005702                         TST     R2              ;;CHECK IF BCD DIGIT=0
031632  001002                         BNE     5$              ;;FALL THROUGH IF 0
031634  105716                         TSTB    (SP)            ;;STILL DOING LEADING 0'S?
031636  100407                         BMI     7$              ;;BR IF YES
031640  106316                 5$:     ASLB    (SP)            ;;MSD?
031642  103003                         BCC     6$              ;;BR IF NO
031644  116663  000001  177777         MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
031652  052702  000060         6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
031656  052702  000040         7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
031662  110223                         MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
031664  005720                         TST     (R0)+           ;;JUST INCREMENTING
031666  020027  000010                 CMP     R0,#10          ;;CHECK THE TABLE INDEX
031672  002746                         BLT     2$              ;;GO DO THE NEXT DIGIT
031674  003002                         BGT     8$              ;;GO TO EXIT
031676  010502                         MOV     R5,R2           ;;GET THE LSD
031700  000764                         BR      6$              ;;GO CHANGE TO ASCII
031702  105726                 8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
031704  100003                         BPL     9$              ;;BR IF NO
031706  116663  177777  177776         MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
031714  105013                 9$:     CLRB    (R3)            ;;SET THE TERMINATOR
031716  012605                         MOV     (SP)+,R5        ;;POP STACK INTO R5
031720  012603                         MOV     (SP)+,R3        ;;POP STACK INTO R3
031722  012602                         MOV     (SP)+,R2        ;;POP STACK INTO R2
031724  012601                         MOV     (SP)+,R1        ;;POP STACK INTO R1
031726  012600                         MOV     (SP)+,R0        ;;POP STACK INTO R0
031730  104401  031756                 TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
031734  016666  000002  000004         MOV     2(SP),4(SP)     ;;ADJUST THE STACK
```

```
7108  031742  012616                        MOV     (SP)+,(SP)
7109  031744  000002                        RTI                             ;;RETURN TO USER
7110  031746  023420                $DTBL:  10000.
7111  031750  001750                        1000.
7112  031752  000144                        100.
7113  031754  000012                        10.
7114  031756  000004                $DBLK:  .BLKW   4
7115
7116                                  .SBTTL  TTY INPUT ROUTINE
7117
7118                                  ;;**************************************************************
7119                                  .ENABL  LSB
7120
7121                                  ;;**************************************************************
7122                                  ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
7123                                  ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
7124                                  ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
7125                                  ;*WHEN OPERATING IN TTY FLAG MODE.
7126  031766  022737  000176  001140 $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
7127  031774  001074                        BNE     15$             ;;BRANCH IF NO
7128  031776  105777  147142                TSTB    @$TKS           ;;CHAR THERE?
7129  032002  100071                        BPL     15$             ;;IF NO, DON'T WAIT AROUND
7130  032004  117746  147136                MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
7131  032010  042716  177600                BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
7132  032014  022726  000007                CMP     #7,(SP)+        ;;IS IT A CONTROL G?
7133  032020  001062                        BNE     15$             ;;NO, RETURN TO USER
7134  032022  123727  001134  000001        CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
7135  032030  001456                        BEQ     15$             ;;BRANCH IF YES
7136
7137  032032  104401  032375                TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
7138  032036  104401  032402 $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
7139  032042  013746  000176                MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
7140  032046  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
7141  032050  104401  032413                TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
7142  032054  005046                 19$:   CLR     -(SP)           ;;CLEAR COUNTER
7143  032056  005046                        CLR     -(SP)           ;;THE NEW SWR
7144  032060  105777  147060          7$:   TSTB    @$TKS           ;;CHAR THERE?
7145  032064  100375                        BPL     7$              ;;IF NOT TRY AGAIN
7146
7147  032066  117746  147054                MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
7148  032072  042716  177600                BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
7149
7150
7151
7152  032076  021627  000025          9$:   CMP     (SP),#25        ;;IS IT A CONTROL-U?
7153  032102  001005                        BNE     10$             ;;BRANCH IF NOT
7154  032104  104401  032370                TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
7155  032110  062706  000006          20$:  ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
7156  032114  000757                        BR      19$             ;;LET'S TRY IT AGAIN
7157
7158
7159  032116  021627  000015          10$:  CMP     (SP),#15        ;;IS IT A <CR>?
7160  032122  001022                        BNE     16$             ;;BRANCH IF NO
7161  032124  005766  000004                TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
7162  032130  001403                        BEQ     11$             ;;BRANCH IF YES
7163  032132  016677  000002  147000        MOV     2(SP),@SWR      ;;SAVE NEW SWR
```

```
7164  032140  062703  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
7165  032144  104401  001165          14$:    TYPE    ,$CRLF          ;;ECHO (CR) AND (LF)
7166  032150  123727  001135  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
7167  032156  001003                          BNE     15$             ;;BRANCH IF NOT
7168  032160  012777  000100  146756          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
7169  032166  000002                  15$:    RTI                     ;;RETURN
7170  032170  004737  031244          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
7171  032174  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
7172  032200  002420                          BLT     18$             ;;BRANCH IF YES
7173  032202  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
7174  032206  003015                          BGT     18$             ;;BRANCH IF YES
7175  032210  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
7176  032214  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
7177  032220  001403                          BEQ     17$             ;;BRANCH IF YES
7178  032222  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
7179  032224  006316                          ASL     (SP)            ;;   CHAR OVER TO MAKE
7180  032226  006316                          ASL     (SP)            ;;   ROOM FOR NEW ONE.
7181  032230  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
7182  032234  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
7183  032240  000707                          BR      7$              ;;GET THE NEXT ONE
7184  032242  104401  001164          18$:    TYPE    ,$QUES          ;;TYPE ?(CR)(LF)
7185  032246  000720                          BR      20$             ;;SIMULATE CONTROL-U
7186                                  .DSABL  LSB
7187
7188
7189                                  ;;*****************************************************************
7190                                  ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
7191                                  ;*CALL:
7192                                  ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
7193                                  ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
7194                                  ;*                              ;;WITH PARITY BIT STRIPPED OFF
7195                                  ;
7196
7197  032250  011646          $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
7198  032252  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
7199  032260  105777  146660          1$:     TSTB    @$TKS           ;;WAIT FOR
7200  032264  100375                          BPL     1$              ;;A CHARACTER
7201  032266  117766  146654  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
7202  032274  042766  177600  000004          BIC     #^C(177),4(SP)  ;;GET RID OF JUNK IF ANY
7203  032302  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
7204  032310  001013                          BNE     3$              ;;BRANCH IF NO
7205  032312  105777  146626          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
7206  032316  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
7207  032320  117746  146622                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
7208  032324  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
7209  032330  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
7210  032334  001366                          BNE     2$              ;;IF NOT DISCARD IT
7211  032336  000750                          BR      1$              ;;YES, RESUME
7212  032340  026627  000004  000140  3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
7213  032346  002407                          BLT     4$              ;;BRANCH IF YES
7214  032350  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
7215  032356  003003                          BGT     4$              ;;BRANCH IF YES
7216  032360  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
7217  032366  000002                  4$:     RTI                     ;;GO BACK TO USER
7218  032370  052536  005015  000     $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL "U"
7219  032375  136  006507  000012     $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL "G"
```

```
7220  032402  005015  053523  020122  $MSWR:  .ASCIZ  <15><12>/SWR = /
7221  032410  020075      000
7222  032413      040  047040  053505  $MNEW:  .ASCIZ  /  NEW = /
7223  032420  036440  000040
7224
7225                                  .SBTTL  RANDOM NUMBER GENERATOR ROUTINE
7226
7227                                  ;;*****************************************************************
7228                                  ;*THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
7229                                  ;*WITH A RANGE OF 0 TO 2(+33)-1.
7230                                  ;*CALL:
7231                                  ;*      JSR     PC,$RAND        ;;CALL THE ROUTINE
7232                                  ;*      RETURN                  ;;RETURN HERE THE RANDOM
7233                                  ;*                              ;;NUMBER WILL BE IN
7234                                  ;*                              ;;$HINUM,$LONUM
7235
7236  032424                          $RAND:
7237  032424  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
7238  032426  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
7239  032430  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
7240  032432  013700  032524                  MOV     $LONUM,R0       ;;SET R0 WITH LOW
7241  032436  013701  032522                  MOV     $HINUM,R1       ;;SET R1 WITH HIGH
7242  032442  012702  177771                  MOV     #-7,R2          ;;SET SHIFT COUNT
7243  032446  006300                  1$:     ASL     R0              ;;SHIFT R0 LEFT AND
7244  032450  006101                          ROL     R1              ;;ROTATE CARRY INTO R1 AND
7245  032452  005202                          INC     R2              ;;CHECK FOR DONE
7246  032454  001374                          BNE     1$              ;;CONTINUE SHIFT LOOP
7247  032456  063700  032524                  ADD     $LONUM,R0       ;;ADD NUMBER TO MAKE X 129
7248  032462  005501                          ADC     R1              ;;PROPOGATE CARRY
7249  032464  063701  032522                  ADD     $HINUM,R1       ;;ADD NUMBER TO MAKE X 129
7250  032470  062700  001057                  ADD     #1057,R0        ;;ADD LOW CONSTANT
7251  032474  005501                          ADC     R1              ;;PROPOGATE CARRY
7252  032476  062701  047401                  ADD     #47401,R1       ;;ADD HIGH CONSTANT
7253  032502  010037  032524                  MOV     R0,$LONUM       ;;SAVE R0
7254  032506  010137  032522                  MOV     R1,$HINUM       ;;SAVE R1
7255  032512  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
7256  032514  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
7257  032516  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
7258  032520  000207                          RTS     PC              ;;RETURN
7259  032522  176543          $HINUM: .WORD   176543
7260  032524  123456          $LONUM: .WORD   123456
7261
7262                                  .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES
7263
7264                                  ;;*****************************************************************
7265                                  ;*SAVE R0-R5
7266                                  ;*CALL:
7267                                  ;*      SAVREG
7268                                  ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
7269                                  ;*
7270                                  ;*TOP---(+16)
7271                                  ;* +2---(+18)
7272                                  ;* +4---R5
7273                                  ;* +6---R4
7274                                  ;* +8---R3
7275                                  ;*+10---R2
```

```
7276                                    ;*+12---R1
7277                                    ;*+14---R0
7278
7279  032526                            $SAVREG:
7280  032526  010046                            MOV     R0,-(SP)        ;;PUSH R0 ON STACK
7281  032530  010146                            MOV     R1,-(SP)        ;;PUSH R1 ON STACK
7282  032532  010246                            MOV     R2,-(SP)        ;;PUSH R2 ON STACK
7283  032534  010346                            MOV     R3,-(SP)        ;;PUSH R3 ON STACK
7284  032536  010446                            MOV     R4,-(SP)        ;;PUSH R4 ON STACK
7285  032540  010546                            MOV     R5,-(SP)        ;;PUSH R5 ON STACK
7286  032542  016646  000022                    MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
7287  032546  016646  000022                    MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
7288  032552  016646  000022                    MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
7289  032556  016646  000022                    MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
7290  032562  000002                            RTI
7291
7292                                    ;*RESTORE R0-R5
7293                                    ;*CALL:
7294                                    ;*      RESREG
7295  032564                            $RESREG:
7296  032564  012666  000022                    MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
7297  032570  012666  000022                    MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
7298  032574  012666  000022                    MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
7299  032600  012666  000022                    MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
7300  032604  012605                            MOV     (SP)+,R5        ;;POP STACK INTO R5
7301  032606  012604                            MOV     (SP)+,R4        ;;POP STACK INTO R4
7302  032610  012603                            MOV     (SP)+,R3        ;;POP STACK INTO R3
7303  032612  012602                            MOV     (SP)+,R2        ;;POP STACK INTO R2
7304  032614  012601                            MOV     (SP)+,R1        ;;POP STACK INTO R1
7305  032616  012600                            MOV     (SP)+,R0        ;;POP STACK INTO R0
7306  032620  000002                            RTI
7307
7308                                    .SBTTL  DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
7309
7310                                    ;;*****************************************************************
7311                                    ;*THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
7312                                    ;*DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
7313                                    ;*POSITIVE.
7314                                    ;*CALL
7315                                    ;*      MOV     #PNTR,-(SP)     ;;POINTER TO LOW WORD OF BINARY NUMBER
7316                                    ;*      JSR     PC,@#$DB2D
7317                                    ;*      RETURN                  ;;THE FIRST ADDRESS OF ASCIZ
7318                                                                    ;;IS ON THE STACK
7319
7320
7321  032622  104412                    $DB2D:  SAVREG                  ;;SAVE REGISTERS
7322  032624  016602  000002                    MOV     2(SP),R2        ;;PICKUP THE DATA POINTER
7323  032630  012700  033002                    MOV     #$DECVL,R0      ;;GET ADDRESS OF "$DECVL" STRING
7324  032634  010066  000002                    MOV     R0,2(SP)        ;;PUT ADDRESS OF ASCIZ STRING ON STACK
7325  032640  012201                            MOV     (R2)+,R1        ;;PICKUP THE BINARY NUMBER
7326  032642  012202                            MOV     (R2)+,R2
7327  032644  012737  000012  032720            MOV     #10.,4$         ;;SET UP TO DO 10 CONVERSIONS
7328  032652  012704  032732                    MOV     #$TNPWR,R4      ;;ADDRESS OF TEN POWER
7329  032656  012705  032734                    MOV     #$TNPWR+2,R5
7330  032662  005003                    1$:     CLR     R3              ;;CLEAR PARTIAL
7331  032664  161401                    2$:     SUB     (R4),R1         ;;SUBTRACT TEN POWER
```

```
7332  032666  005602                        SBC     R2
7333  032670  161502                        SUB     (R5),R2
7334  032672  002402                        BLT     3$              ;;BR IF TEN POWER TO LARGE
7335  032674  005203                        INC     R3              ;;ADD 1 TO PARTIAL.
7336  032676  000772                        BR      2$              ;;LOOP
7337  032700  062401              3$:       ADD     (R4)+,R1        ;;RESTORE SUBTRACTED VALUE
7338  032702  005502                        ADC     R2
7339  032704  062402                        ADD     (R4)+,R2
7340  032706  022525                        CMP     (R5)+,(R5)+     ;;MOVE TO NEXT TEN POWER
7341  032710  052703  000060                BIS     #'0,R3          ;;CHANGE PARTIAL TO ASCII
7342  032714  110320                        MOVB    R3,(R0)+        ;;SAVE IT
7343  032716  005327                        DEC     (PC)+           ;;DONE?
7344  032720  000000              4$:       .WORD   0
7345  032722  001357                        BNE     1$              ;;BR IF NO
7346  032724  105020                        CLRB    (R0)+           ;;TERMINATOR
7347  032726  104413                        RESREG                  ;;RESTORE REGISTERS
7348  032730  000207                        RTS     PC              ;;RETURN
7349  032732  145000              $TNPWR:   145000                  ;;1.0E09
7350  032734  035632                        35632
7351  032736  160400                        160400                  ;;1.0E08
7352  032740  002765                        2765
7353  032742  113200                        113200                  ;;1.0E07
7354  032744  000230                        230
7355  032746  041100                        041100                  ;;1.0E06
7356  032750  000017                        17
7357  032752  103240                        103240                  :;1.0E05
7358  032754  000001                        1
7359  032756  023420                        23420                   ;;1.0E04
7360  032760  000000                        0
7361  032762  001750                        1750                    ;;1.0E03
7362  032764  000000                        0
7363  032766  000144                        144                     ;;1.0E02
7364  032770  000000                        0
7365  032772  000012                        12                      ;;1.0E01
7366  032774  000000                        0
7367  032776  000001                        1                       ;;1.0E00
7368  033000  000000                        0
7369  033002  000014              $DECVL:   .BLKB   12.             ;;RESERVE STORAGE FOR ASCIZ STRING
7370
7371                                        .SBTTL  DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
7372
7373                                        ;;**************************************************************
7374                                        ;*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
7375                                        ;*UNSIGNED OCTAL ASCIZ NUMBER.
7376                                        ;*CALL
7377                                        ;*      MOV     #PNTR,-(SP)     ;;POINTER TO LOW WORD OF BINARY NUMBER
7378                                        ;*      JSR     PC,@#$DB20      ;;CALL THE ROUTINE
7379                                        ;*      RETURN                  ;;THE ADDRESS OF THE FIRST ASCIZ CHAR. IS ON THE STACk
7380
7381
7382  033016  104412              $DB20:    SAVREG                  ;;SAVE ALL REGISTERS
7383  033020  016601  000002                MOV     2(SP),R1        ;;PICKUP THE POINTER TO LOW WORD
7384  033024  012705  033135                MOV     #$OCTVL+13.,R5  ;;POINTER TO DATA TABLE
7385  033030  012704  000014                MOV     #12.,R4         ;;DO ELEVEN CHARACTERS
7386  033034  012703  177770                MOV     #^C7,R3         ;;MASK
7387  033040  012100                        MOV     (R1)+,R0        ;;LOWER WORD
```

```
7388  033042  012101                        MOV     (R1)+,R1        ;;HIGH WORD
7389  033044  005002                        CLR     R2              ;;TERMINATOR
7390  033046  110245                1$:     MOVB    R2,-(R5)        ;;PUT CHARACTER IN DATA TABLE
7391  033050  010002                        MOV     R0,R2           ;;GET THIS DIGIT
7392  033052  005304                        DEC     R4              ;;COUNT THIS CHARACTER
7393  033054  003007                        BGT     3$              ;;BR IF NOT THE LAST DIGIT
7394  033056  001405                        BEQ     2$              ;;BR IF IT IS THE LAST DIGIT
7395  033060  005205                        INC     R5              ;;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
7396  033062  010566  000002               MOV     R5,2(SP)        ;;ASCIZ CHAR. & PUT IT ON THE STACK
7397  033066  104413                        RESREG                  ;;RESTORE ALL REGISTERS
7398  033070  000207                        RTS     PC              ;;RETURN TO USER
7399  033072  006203                2$:     ASR     R3              ;;POSITION THE MASK FOR THE LAST DIGIT
7400  033074  006001                3$:     ROR     R1              ;;POSITION THE BINARY NUMBER FOR
7401  033076  006000                        ROR     R0              ;;      THE NEXT OCTAL DIGIT
7402  033100  006001                        ROR     R1
7403  033102  006000                        ROR     R0
7404  033104  006001                        ROR     R1
7405  033106  006000                        ROR     R0
7406  033110  040302                        BIC     R3,R2           ;;MASK OUT ALL JUNK
7407  033112  062702  000060                ADD     #'0,R2          ;;MAKE THIS CHAR. ASCII
7408  033116  000753                        BR      1$              ;;GO PUT IT IN THE DATA TABLE
7409  033120  000016                $OCTVL: .BLKB   14.              ;;RESERVE DATA TABLE
7410
7411                                  .SBTTL  TRAP DECODER
7412
7413                                  ;;**************************************************************
7414                                  ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
7415                                  ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
7416                                  ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
7417                                  ;*GO TO THAT ROUTINE.
7418
7419  033136  010046                $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
7420  033140  016600  000002                MOV     2(SP),R0        ;;GET TRAP ADDRESS
7421  033144  005740                        TST     -(R0)           ;;BACKUP BY 2
7422  033146  111000                        MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
7423  033150  006300                        ASL     R0              ;;POSITION FOR INDEXING
7424  033152  016000  033172                MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
7425  033156  000200                        RTS     R0              ;;GO TO ROUTINE
7426
7427
7428                                  ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
7429
7430  033160  011646                $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
7431  033162  016666  000004  000002        MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
7432  033170  000002                        RTI                     ;;RESTORE THE PSW
7433
7434                                  .SBTTL  TRAP TABLE
7435
7436                                  ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
7437                                  ;*BY THE "TRAP" INSTRUCTION.
7438
7439                                  ;       ROUTINE
7440                                  ;       -------
7441  033172  033160                $TRPAD: .WORD   $TRAP2
7442  033174  031074                        $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
7443  033176  031340                        $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
```

```
7444  033200  031314                        $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
7445  033202  031354                        $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
7446  033204  031542                        $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
7447
7448  033206  032036                        $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING
7449
7450  033210  031766                        $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
7451  033212  032250                        $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
7452  033214  030272                        $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
7453  033216  032526                        $SAVREG ;;CALL=SAVREG   TRAP+12(104412) SAVE R0-R5 ROUTINE
7454  033220  032564                        $RESREG ;;CALL=RESREG   TRAP+13(104413) RESTORE R0-R5 ROUTINE
7455  033222  027516                        $DSPLY  ;;CALL=DISPLY   TRAP+14(104414) ROUTINE TO TYPE ERROR MESSAGES
7456           000032              $TERM=.-$TRPAD
7457
7458                              ;;**********************************************************************
7459
7460
7461
7462                              .SBTTL  SINGLE/DUAL PORT RH11/RP04/5/6 DRIVER (REV 1.0)
7463
7464                              ;COPYRIGHT (C) 1976
7465                              ;DIGITAL EQUIPMENT CORP.
7466                              ;MAYNARD, MA 01754
7467                              ;AUTHOR(S): JIM LACEY/CHUCK HESS
7468
7469                              ;;**********************************************************************
7470
7471                              ;STORAGE FOR RPDS1, RPER1, RPER2, AND RPER3 ON AN ERROR "2"
7472                                      ;RPERRS   = RPDS1
7473                                      ;RPERRS+2 = RPER1
7474                                      ;RPERRS+4 = RPER2
7475                                      ;RPERRS+6 = RPER3
7476
7477  033224  000000  000000  000000  RPERRS: .WORD   0,0,0,0
7478  033232  000000
7479
7480                              ;TABLE OF DRIVE ACTIVE INDICATORS (DRVACT=8 BYTES)
7481                                      ;DRVACT=0 IF DRIVE IS IDLE
7482                                      ;DRVACT>0 IF DRIVE IS ACTIVE WITH A COMMAND
7483                                      ;DRVACT<0 IF DRIVE IS ACTIVE WITH AN ERROR RECOVERY OPERATION
7484
7485  033234      000                 DRVACT: .BYTE   0               ;DRIVE 0
7486  033235      000                         .BYTE   0               ;DRIVE 1
7487  033236      000                         .BYTE   0               ;DRIVE 2
7488  033237      000                         .BYTE   0               ;DRIVE 3
7489  033240      000                         .BYTE   0               ;DRIVE 4
7490  033241      000                         .BYTE   0               ;DRIVE 5
7491  033242      000                         .BYTE   0               ;DRIVE 6
7492  033243      000                         .BYTE   0               ;DRIVE 7
7493
7494                              ;TABLE OF DRIVE STATUS INDICATORS (DRVSTA=8 BYTES)
7495                                      ;DRVSTA=0 IF DRIVE IS OFFLINE OR NONEXSITENT
7496                                      ;DRVSTA>0 IF DRIVE IS ONLINE
7497                                      ;DRVSTA<0 IF DRIVE IS UNSAFE
7498
7499  033244      000                 DRVSTA: .BYTE   0               ;DRIVE 0
```

```
7500  033245  000                     .BYTE   0               ;DRIVE 1
7501  033246  000                     .BYTE   0               ;DRIVE 2
7502  033247  000                     .BYTE   0               ;DRIVE 3
7503  033250  000                     .BYTE   0               ;DRIVE 4
7504  033251  000                     .BYTE   0               ;DRIVE 5
7505  033252  000                     .BYTE   0               ;DRIVE 6
7506  033253  000                     .BYTE   0               ;DRIVE 7
7507
7508                          ;TABLE OF DRIVE TYPES (DRVTYP=8 BYTES)
7509                                  ;DRVTYP=0 IF DRIVE IS NONEXISTENT (DRVSTA=0, ALSO)
7510                                  ;DRVTYP=1 IF DRIVE IS RP04
7511                                  ;DRVTYP=2 IF DRIVE IS RP05
7512                                  ;DRVTYP=4 IF DRIVE IS RP06
7513                                  ;DRVTYP=-1 IF NOT RP04/5/6
7514
7515  033254  000             DRVTYP: .BYTE   0               ;DRIVE 0
7516  033255  000                     .BYTE   0               ;DRIVE 1
7517  033256  000                     .BYTE   0               ;DRIVE 2
7518  033257  000                     .BYTE   0               ;DRIVE 3
7519  033260  000                     .BYTE   0               ;DRIVE 4
7520  033261  000                     .BYTE   0               ;DRIVE 5
7521  033262  000                     .BYTE   0               ;DRIVE 6
7522  033263  000                     .BYTE   0               ;DRIVE 7
7523
7524                          ;TABLE OF DUAL PORT INITIALIZATION INDICATORS
7525                                  ;DPINT=0 IF INITIALIZATION IS NOT ACTIVE ON THE DRIVE
7526                                  ;DPINT<0 IF INITIALIZATION IS IN PROGRESS
7527
7528  033264  000             DPINT:  .BYTE   0               ;DRIVE 0
7529  033265  000                     .BYTE   0               ;DRIVE 1
7530  033266  000                     .BYTE   0               ;DRIVE 2
7531  033267  000                     .BYTE   0               ;DRIVE 3
7532  033270  000                     .BYTE   0               ;DRIVE 4
7533  033271  000                     .BYTE   0               ;DRIVE 5
7534  033272  000                     .BYTE   0               ;DRIVE 6
7535  033273  000                     .BYTE   0               ;DRIVE 7
7536
7537                          ;TABLE OF PENDING DUAL PORT REQUESTS
7538                                  ;DPRQS=0 IF THAT A DUAL PORT REQUEST IS NOT PENDING FOR THAT DRIVE
7539                                  ;DPRQS<0 IF THAT A DUAL PORT REQUEST IS PENDING FOR THAT DRIVE
7540
7541  033274  000             DPRQS:  .BYTE   0               ;DRIVE 0
7542  033275  000                     .BYTE   0               ;DRIVE 1
7543  033276  000                     .BYTE   0               ;DRIVE 2
7544  033277  000                     .BYTE   0               ;DRIVE 3
7545  033300  000                     .BYTE   0               ;DRIVE 4
7546  033301  000                     .BYTE   0               ;DRIVE 5
7547  033302  000                     .BYTE   0               ;DRIVE 6
7548  033303  000                     .BYTE   0               ;DRIVE 7
7549
7550                          ;TRANSFER WAIT FLAG (TRNSWT=1 WORD)
7551                                  ;THIS IS A ONE WORD QUEUE. IT WILL CONTAIN THE ADDRESS OF
7552                                  ;"DPB" OF THE I/O OPERATION.
7553
7554  033304  000000          TRNSWT: .WORD   0
7555
```

```
7556                            ;SEARCH WAIT KEYS (SRCHWT=1 WORD)
7557                                    ;THIS IS A ONE WORD QUEUE THAT WILL CONTAIN A KEY FOR EACH OF
7558                                    ;THE DRIVES THAT ARE PERFORMING A SEARCH COMMAND FOR THE I/O
7559                                    ;REQUEST THAT IS AT THE TOP OF THEIR REQUEST QUEUE.
7560                                    ;EACH DRIVE IS ASSIGNED ONE BIT, STARTING AT BIT00 FOR DRIVE 0.
7561
7562  033306  000000                SRCHWT: .WORD   0
7563
7564                            ;RP04/5/6 DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
7565                                    ;ACTDRV=0 IF DRIVER IS INACTIVE
7566                                    ;ACTDRV>0 IF DRIVER IS ACTIVE
7567
7568  033310      000           ACTDRV: .BYTE   0
7569
7570                            ;SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)
7571                                    ;ACTSTR=0 IF SOFTWARE TIMER ROUTINE IS INACTIVE
7572                                    ;ACTSTR>0 IF SOFTWARE TIMER ROUTINE IS ACTIVE
7573
7574  033311      000           ACTSTR: .BYTE   0
7575
7576                            ;UNLOAD FLAG (ULDFLG=8 BYTES)
7577                                    ;ULDFLG=0 IF NO UNLOAD COMMAND
7578                                    ;ULDFLG>0 IF UNLOAD COMMAND IN PROGRESS
7579                                    ;ULDFLG<0 IF UNLOAD COMMAND IN WAIT QUEUE
7580
7581  033312      000           ULDFLG: .BYTE   0               ;DRIVE 0
7582  033313      000                   .BYTE   0               ;DRIVE 1
7583  033314      000                   .BYTE   0               ;DRIVE 2
7584  033315      000                   .BYTE   0               ;DRIVE 3
7585  033316      000                   .BYTE   0               ;DRIVE 4
7586  033317      000                   .BYTE   0               ;DRIVE 5
7587  033320      000                   .BYTE   0               ;DRIVE 6
7588  033321      000                   .BYTE   0               ;DRIVE 7
7589
7590                            ;LOOK AHEAD COUNT (LACNT=8 BYTES)
7591                                    ;LACNT WILL INDICATE THE NUMBER OF LOOK AHEADS PERFORMED
7592
7593  033322      000           LACNT:  .BYTE   0               ;DRIVE 0
7594  033323      000                   .BYTE   0               ;DRIVE 1
7595  033324      000                   .BYTE   0               ;DRIVE 2
7596  033325      000                   .BYTE   0               ;DRIVE 3
7597  033326      000                   .BYTE   0               ;DRIVE 4
7598  033327      000                   .BYTE   0               ;DRIVE 5
7599  033330      000                   .BYTE   0               ;DRIVE 6
7600  033331      000                   .BYTE   0               ;DRIVE 7
7601
7602                            ;SAVE REGISTERS FLAG (SAVEFG =1 WORD)
7603                                    ;SAVEFG <0 IF SAVE THE RH11/RP04/5/6 REGISTERS WHEN THE
7604                                    ;OPERATION IS COMPLETED AS PER (DPB+14).
7605                                    ;SAVEFG=0 IF SAVE THE RH11/RP04/5/6 REGISTERS, AS PER
7606                                    ;(DPB+14), AFTER AN ERROR.
7607
7608  033332  000000                SAVEFG: .WORD   0
7609
7610                            ;SEEK FLAG (SEEKFG=1 WORD)
7611                                    ;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

033334  000000                  SEEKFG: .WORD   0

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

033336  177777                  TIMER:  .WORD   -1       ;DRIVE 0
033340  177777                          .WORD   -1       ;DRIVE 1
033342  177777                          .WORD   -1       ;DRIVE 2
033344  177777                          .WORD   -1       ;DRIVE 3
033346  177777                          .WORD   -1       ;DRIVE 4
033350  177777                          .WORD   -1       ;DRIVE 5
033352  177777                          .WORD   -1       ;DRIVE 6
033354  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

033356  177777                  DTUW:   .WORD   -1

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

033360  001                     ATABIT: .BYTE   1        ;DRIVE 0
033361  002                             .BYTE   2        ;DRIVE 1
033362  004                             .BYTE   4        ;DRIVE 2
033363  010                             .BYTE   10       ;DRIVE 3
033364  020                             .BYTE   20       ;DRIVE 4
033365  040                             .BYTE   40       ;DRIVE 5
033366  100                             .BYTE   100      ;DRIVE 6
033367  200                             .BYTE   200      ;DRIVE 7

                                ;RP04/5/6 TO RH11 "MASSBUS CONTROL BUS PARITY ERRORS" (MCPE) ALLOWED BEFORE
                                ;CALLING IT FATAL (MCPEMX=1 WORD)

033370  000003                  MCPEMX: .WORD   3

                                ;STORAGE FOR RPADR (THE FIRST ADDRESS (776700) OF THE RH11/RP04/5/6),
                                ;RPVEC (THE VECTOR ADDRESS (254)), AND RPVEC+2 (THE BR LEVEL (5)).

033372  176700                  RPADR:  .WORD   176700
033374  000254  000240          RPVEC:  .WORD   254,5*32.

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

033400  000004                  MXLACT: .WORD   4
                                ;MAXIMUM DELTA DELAY IS 8 SECTORS (MXDLTA=1 WORD)

033402  001000                  MXDLTA: .WORD   8.*64.
                                ;MINIMUM DELTA DELAY IS 2 SECTORS (MNDLTA=1 WORD)
```

```
7668  033404  000200                  MNDLTA: .WORD   2*64.
7669                                  ;MAXIMUM SEARCH FOR I/O WINDOW IS 5 SECTORS (MXWNDW=1 WORD)
7670
7671  033406  000005                  MXWNDW: .WORD   5
7672
7673                                  ;DEFINITIONS OF THE RH11/RPO4/5/6 ADDRESS INDEXES
7674
7675          000000                  RPCS1=0                         ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00)
7676          000002                  RPWC=2                          ;WORD COUNT REGISTER (NOT A DRIVE REG)
7677          000004                  RPBA=4                          ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
7678          000006                  RPDA=6                          ;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 05)
7679          000010                  RPCS2=10                        ;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
7680          000012                  RPDS1=12                        ;DRIVE STATUS REGISTER (DRIVE REG 01)
7681          000014                  RPER1=14                        ;ERROR REGISTER #1 (DRIVE REG. 02)
7682          000016                  RPAS=16                         ;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 04)
7683          000020                  RPLA=20                         ;LOOK AHEAD REGISTER (DRIVE REG. 07)
7684          000022                  RPDB=22                         ;DATA BUFFER REGISTER (NOT A DRIVE REG.)
7685          000024                  RPMR=24                         ;MAINTAINABILITY REGISTER (DRIVE REG. 03)
7686          000026                  RPDT=26                         ;DRIVE TYPE REGISTER (DRIVE REG. 06)
7687          000030                  RPSN=30                         ;SERIAL NUMBER REGISTER (DRIVE REG. 10)
7688          000032                  RPOF=32                         ;OFFSET REGISTER (DRIVE REG. 11)
7689          000034                  RPCA=34                         ;DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
7690          000036                  RPCC=36                         ;CURRENT CYLINDER ADDRESS REGISTER (DRIVE REG. 13)
7691          000040                  RPER2=40                        ;ERROR REGISTER #2 (DRIVE REG. 14)
7692          000042                  RPER3=42                        ;ERROR REGISTER #3 (DRIVE REG. 15)
7693          000044                  RPEC1=44                        ;ECC POSITION REGISTER (DRIVE REG. 16)
7694          000046                  RPEC2=46                        ;ECC PATTERN REGISTER (DRIVE REG.  17)
7695
7696                                  ;RH11/RPO4/5/6 DRIVER INITIALIZATION CODE
7697                                          ;THIS ROUTINE WILL DETERMINE WHICH RPO4/5/6 DRIVES ARE
7698                                          ;AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
7699                                          ;TO THE PROPER STATE FOR EACH DRIVE.
7700                                          ;NOTE: THIS ROUTINE CALLS DRVINT
7701                                  ;
7702                                  ;CALL
7703                                  ;
7704                                  ;       JSR     PC,RPINIT
7705                                  ;       RETURN
7706                                  ;
7707                                  ;NOTE: THE 'P' OR 'L' CLOCK MUST BE STARTED
7708                                  ;
7709  033410  104412                  RPINIT: SAVREG                          ;SAVE R0 - R5
7710  033412  013746  177776                  MOV     @#PS,-(SP)             ;SAVE THE PRESENT PROCESSOR STATUS
7711  033416  012737  000240  177776          MOV     #(5*32.),@#PS          ;CHANGE THE PRIORITY TO 5
7712  033424  004737  041472                  JSR     PC,CLRQUE              ;CLEAR ALL REQUEST QUEUES
7713  033430  012701  033224                  MOV     #RPERRS,R1             ;FIRST ADDRESS TO BE CLEARED
7714  033434  012702  033334                  MOV     #SEEKFG,R2             ;LAST ADDRESS TO BE CLEARED
7715  033440  005021                  1$:     CLR     (R1)+                  ;CLEAR
7716  033442  020102                          CMP     R1,R2                  ;ARE WE DONE?
7717  033444  101775                          BLOS    1$                     ;BRANCH IF NO
7718  033446  012702  033356                  MOV     #DTUW,R2               ;LAST ADDRESS
7719  033452  012721  177777          2$:     MOV     #-1,(R1)+              ;INITIALIZE
7720  033456  020102                          CMP     R1,R2                  ;DONE?
7721  033460  101774                          BLOS    2$                     ;LOOP IF NO
7722  033462  005037  033244                  CLR     DRVSTA                 ;SET ALL DRIVES TO OFFLINE
7723  033466  005037  033246                  CLR     DRVSTA+2
```

```
7724  033472  005037  033250                  CLR     DRVSTA+4
7725  033476  005037  033252                  CLR     DRVSTA+6
7726  033502  013703  033374                  MOV     RPVEC,R3        ;SETUP THE RH11/RP04/5/6 VECTOR
7727  033506  012723  036254                  MOV     #ISR,(R3)+
7728  033512  013713  033376                  MOV     RPVEC+2,(R3)
7729  033516  013704  033372                  MOV     RPADR,R4        ;FIRST ADDRESS OF RH11/RP04
7730  033522  012764  000040  000010          MOV     #BIT05,RPCS2(R4)  ;MASSBUS INIT
7731  033530  005001                          CLR     R1              ;START WITH DRIVE 0
7732  033532  004037  033622          3$:     JSR     R0,DRVINT       ;INIT THE DRIVE
7733  033536  000401                          BR      4$              ;'DVA' NOT SET OR PARITY ERROR
7734  033540  000402                          BR      5$              ;NORMAL RETURN
7735  033542  105061  033244          4$:     CLRB    DRVSTA(R1)      ;SET DRIVE STATUS TO OFFLINE
7736  033546  005201          5$:     INC     R1              ;GO TO NEXT DRIVE
7737  033550  042701  177770                  BIC     #^C7,R1         ;MASK OUT UNUSED BITS
7738  033554  001366                          BNE     3$              ;BR IF MORE DRIVES TO GO
7739  033556  012701  000007                  MOV     #7,R1           ;START WITH DRIVE 7
7740  033562  005037  177776                  CLR     @#PS            ;CLEAR THE PROCESSOR STATUS
7741  033566  105761  033264          6$:     TSTB    DPINT(R1)       ;WAITING FOR DRIVE TO SWITCH PORTS ?
7742  033572  001405                          BEQ     8$              ;BR NOT WAITING
7743  033574  004737  041126                  JSR     PC,SET.IE       ;SET INTERRUPT
7744  033600  105761  033264          7$:     TSTB    DPINT(R1)       ;DRIVE SWITCHED PORTS ?
7745  033604  001375                          BNE     7$              ;BR IF NOT
7746  033606  005301          8$:     DEC     R1              ;GO TO THE NEXT DRIVE
7747  033610  100366                          BPL     6$              ;CHECK NEXT DRIVE
7748  033612  012637  177776                  MOV     (SP)+,@#PS      ;RESTORE THE PROCESSOR STATUS
7749  033616  104413                          RESREG                  ;RESTORE R0 - R5
7750  033620  000207                          RTS     PC              ;BYE-BYE
7751
7752                                  ;DRIVE INITILIZATION ROUTINE
7753                                  ;THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
7754                                  ;AN RP04/5/6. IF IT IS, A "READ-IN PRESET" IS ISSUED AND FMT22
7755                                  ;IS SET TO A "1". THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
7756                                  ;INSURE THEY ARE ALL ON A "1". AND DEPENDING ON THEIR STATE,
7757                                  ;DRVSTA IS SET TO THE PROPER CONDITION.
7758                                  ;CALL
7759                                  ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
7760                                  ;       MOV     RPADR,R4        ;UNIBUS ADDRESS OF RH11/RP04/5/6 (RPCS1)
7761                                  ;       JSR     R0,DRVINT       ;CALLED BY A JSR
7762                                  ;       RETURN1                 ;ERROR OCCURRED (PARITY)
7763                                  ;       RETURN2                 ;NORMAL RETURN
7764                                  ;
7765
7766  033622  010546          DRVINT: MOV     R5,-(SP)        ;SAVE R5
7767  033624  105061  033244          CLRB    DRVSTA(R1)      ;START DRIVE STATUS AS OFFLINE
7768  033630  105061  033254          CLRB    DRVTYP(R1)      ;CLEAR THE DRIVE TYPE INDICATOR
7769  033634  105061  033312          CLRB    ULDFLG(R1)      ;CLEAR THE UNLOAD FLAG
7770  033640  010164  000010          MOV     R1,RPCS2(R4)    ;SELECT A DRIVE
7771  033644  112764  000111  000000  MOVB    #111,RPCS1(R4)  ;DO A DRIVE CLEAR COMMAND (& SEIZE DRIVE)
7772  033652  032764  010000  000010  BIT     #BIT12,RPCS2(R4)  ;NONEXISTENT DRIVE?
7773  033660  001403                  BEQ     1$              ;NO---BRANCH
7774  033662  004737  041126          JSR     PC,SET.IE       ;GO SET "IE" WITHOUT A "TRE"
7775  033666  000520                  BR      6$              ;LEAVE THIS ROUTINE
7776  033670  105061  033244  1$:     CLRB    DRVSTA(R1)      ;SET DRIVE STATUS TO OFFLINE
7777  033674  032764  004000  000000  BIT     #BIT11,RPCS1(R4)  ;SEE IF DRIVE AVAILABLE
7778  033702  001514                  BEQ     7$              ;BR IF DRIVE NOT AVAILABLE
7779  033704  004037  040446          JSR     R0,RD.RP        ;READ THE DRIVE TYPE REG.
```

```
7780  033710  000026                          RPDT
7781  033712  034154                          8$                          ;ERROR RETURN ADDRESS
7782  033714  012605                          MOV     (SP)+,R5            ;PUT DRIVE TYPE IN R5
7783  033716  112761  000001  033254          MOVB    #1,DRVTYP(R1)       ;SET RP04 INDICATOR
7784  033724  022705  020020                  CMP     #20020,R5           ;IS IT A SINGLE PORT RP04?
7785  033730  001431                          BEQ     2$                  ;BRANCH IF YES
7786  033732  022705  024020                  CMP     #24020,R5           ;IS IT A DUAL PORT RP04?
7787  033736  001426                          BEQ     2$                  ;BR IF YES
7788  033740  112761  000002  033254          MOVB    #2,DRVTYP(R1)       ;SET RP05 INDICATOR
7789  033746  022705  020021                  CMP     #20021,R5           ;SINGLE PORT RP05 ?
7790  033752  001420                          BEQ     2$                  ;BR IF YES
7791  033754  022705  024021                  CMP     #24021,R5           ;DUAL PORT RP05 ?
7792  033760  001415                          BEQ     2$                  ;BR IF YES
7793  033762  112761  000004  033254          MOVB    #4,DRVTYP(R1)       ;SET RP06 INDICATOR
7794  033770  022705  020022                  CMP     #20022,R5           ;SINGLE PORT RP06 ?
7795  033774  001407                          BEQ     2$                  ;BR IF YES
7796  033776  022705  024022                  CMP     #24022,R5           ;DUAL PORT RP06 ?
7797  034002  001404                          BEQ     2$                  ;BR IF YES
7798  034004  112761  177777  033254          MOVB    #-1,DRVTYP(R1)      ;SET INDICATOR TO 'OTHER'
7799  034012  000446                          BR      6$                  ;EXIT
7800  034014  012746  000121          2$:     MOV     #121.,-(SP)         ;DO A "READ-IN PRESET"
7801  034020  004037  040622                  JSR     R0,WRT.RP
7802  034024  000000                          RPCS1
7803  034026  034154                          8$
7804  034030  012746  010000                  MOV     #BIT12,-(SP)        ;SET FMT22=1
7805  034034  004037  040622                  JSR     R0,WRT.RP
7806  034040  000032                          RPOF
7807  034042  034154                          8$
7808  034044  004037  040446                  JSR     R0,RD.RP            ;READ RPDS1
7809  034050  000012                          RPDS1
7810  034052  034154                          8$
7811  034054  012605                          MOV     (SP)+,R5            ;AND SAVE IT IN R5
7812  034056  100015                          BPL     4$                  ;BRANCH IF ATA=0
7813  034060  116164  033360  000016          MOVB    ATABIT(R1),RPAS(R4) ;CLEAR ATTENTION BIT
7814  034066  004037  040446                  JSR     R0,RD.RP            ;FIND OUT WHY ATA=1
7815  034072  000014                          RPER1
7816  034074  034154                          8$
7817  034076  006126                          ROL     (SP)+               ;IS IT UNSAFE?
7818  034100  100004                          BPL     4$                  ;BR IF NOT
7819  034102  112761  177777  033244          MOVB    #-1,DRVSTA(R1)      ;SET UNSAFE INDICATOR
7820  034110  000407                          BR      6$                  ;EXIT
7821  034112  005105                  4$:     COM     R5                  ;CHECK MOL, DPR, DRY, AND VV
7822  034114  042705  167077                  BIC     #^C<BIT12!BIT08!BIT07!BIT06>,R5
7823  034120  001003                          BNE     6$                  ;BRANCH IF MOL, DPR, DRY, OR VV IS CLEAR
7824  034122  112761  000001  033244          MOVB    #1,DRVSTA(R1)       ;SET DRIVE STATUS TO ONLINE
7825  034130  005720                  6$:     TST     (R0)+               ;STEP OVER THE ERROR RETURN
7826  034132  000410                          BR      8$                  ;EXIT
7827  034134  006301                  7$:     ASL     R1                  ;CHANGE INDEX TO ADDRESS WORDS
7828  034136  012761  003720  033336          MOV     #2000.,TIMER(R1)    ;START 2 SEC TIMER
7829  034144  006201                          ASR     R1                  ;RESTORE R1
7830  034146  105161  033264                  COMB    DPINT(R1            ;SET PORT INITIALIZE INIDICATOR
7831  034152  005720                          TST     (R0)+
7832  034154  012605                  8$:     MOV     (SP)+,R5            ;RESTORE R5
7833  034156  000200                          RTS     R0                  ;EXIT
7834
7835                                  ;REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
```

```
                                        ;
                                        ;CALL
                                        ;
                                        ;       JSR     R0,@#RP04       ;CALL THE RP04/5/6 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

034160  013746  177776          RP04:  MOV     @#PS,-(SP)      ;SAVE THE CALLING STATUS
034164  013737  033376  177776         MOV     RPVEC+2,@#PS    ;DON'T ALLOW ANY RP04/5/6 INTERRUPTS
034172  112737  000001  033310         MOVB    #1,ACTDRV       ;SET "ACTIVE DRIVER" FLAG
034200  104412                         SAVREG                  ;SAVE R0 - R5
034202  011002                         MOV     (R0),R2         ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
034204  005062  000016                 CLR     16(R2)          ;CLEAR THE STATUS/ERROR INDICATOR
034210  111201                         MOVB    (R2),R1         ;PICKUP THE DRIVE NUMBER
034212  013704  033372                 MOV     RPADR,R4        ;UNIBUS ADDRESS OF RPCS1
034216  105761  033244                 TSTB    DRVSTA(R1)      ;CHECK DRIVES STATUS
034222  003014                         BGT     1$              ;BRANCH IF ONLINE
034224  105761  033312                 TSTB    ULDFLG(R1)      ;UNLOAD COMMAND IN QUEUE?
034230  001036                         BNE     3$              ;BRANCH IF YES
034232  105761  033264                 TSTB    DPINT(R1)       ;TRYING TO INIT THE DRIVE
034236  001042                         BNE     5$              ;BR IF YES
034240  004037  033622                 JSR     R0,DRVINT       ;GO INIT. THE DRIVE
034244  000434                         BR      4$              ;ERROR RETURN
034246  105761  033244                 TSTB    DRVSTA(R1)      ;IS DRIVE STATUS ONLINE?
034252  003445                         BLE     6$              ;BR IF NOT
034254  105761  033274          1$:    TSTB    DPRQS(R1)       ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
034260  001031                         BNE     5$              ;BR IF YES
034262  010164  000010                 MOV     R1,RPCS2(R4)    ;SELECT THE DRIVE
034266  004037  041570                 JSR     R0,DRVQUE       ;PUT THIS REQUEST IN QUEUE
034272  000460                         BR      9$              ;QUEUE IS FULL
034274  122762  000103  000002         CMPB    #103,2(R2)      ;IS THIS REQ. FOR AN UNLOAD?
034302  001003                         BNE     2$              ;BR IF NO
034304  112761  177777  033312         MOVB    #-1,ULDFLG(R1)  ;SET THE "UNLOAD IN QUEUE" FLAG
034312  105761  033234          2$:    TSTB    DRVACT(R1)      ;IS THIS DRIVE ACTIVE?
034316  001043                         BNE     8$              ;BR IF YES
034320  004737  034452                 JSR     PC,OPT          ;CALL THE OPTIMIZER
034324  000440                         BR      8$
034326  012762  120000  000016  3$:    MOV     #BIT15!BIT13,16(R2) ;SET THE "UNLOAD IN QUEUE" ERROR FLAG
034334  000434                         BR      8$              ;EXIT
034336  004737  035562          4$:    JSR     PC,CI7          ;GO HANDLE THE PARITY ERROR
034342  000431                         BR      8$
034344  004037  041570          5$:    JSR     R0,DRVQUE       ;PUT REQUEST IN QUEUE
034350  000431                         BR      9$              ;QUEUE IS FULL
034352  032714  000100                 BIT     #BIT06,(R4)     ;IS 'IE' SET ALREADY ?
034356  001023                         BNE     8$              ;BR IF IT IS
034360  004737  041126                 JSR     PC,SET.IE       ;SET INTERRUPT
034364  000420                         BR      8$              ;RETURN, REQUEST IN QUEUE
034366  105761  033244          6$:    TSTB    DRVSTA(R1)      ;SEE IF DRIVE OFFLINE OR UNSAFE
034372  002412                         BLT     7$              ;BR IF UNSAFE
034374  012762  140000  000016         MOV     #BIT15!BIT14,16(R2)  ;SET OFFLINE ERROR INDICATOR
034402  105761  033254                 TSTB    DRVTYP(R1)      ;SEE IF OFFLINE OR NONEXISTENT
034406  001007                         BNE     8$              ;BR IF OFFLINE
034410  012762  100002  000016         MOV     #BIT15!BIT01,16(R2)  ;REPORT DRIVE NONEXISTENT
034416  000403                         BR      8$              ;GO TO EXIT
```

```
 7892  034420  012762  110000  000016  7$:     MOV     #BIT15!BIT12,16(R2)  ;DRIVE IS UNSAFE
 7893  034426  104413                  8$:     RESREG                       ;RESTORE R0 - R5
 7894  034430  005720                          TST     (R0)+                ;SETUP FOR NORMAL RETURN
 7895  034432  000401                          BR      10$                  ;FINISH UP, THEN EXIT
 7896  034434  104413                  9$:     RESREG                       ;RESTORE R0 - R5
 7897  034436  005720                  10$:    TST     (R0)+                ;CORRECT THE RETURN ADDRESS
 7898  034440  105037  033310                  CLRB    ACTDRV               ;CLEAR "ACTIVE DRIVER" FLAG
 7899  034444  012637  177776                  MOV     (SP)+,@#PS           ;RETURN "PS" TO USER LEVEL
 7900  034450  000200                          RTS     R0                   ;RETURN TO CALLER
 7901
 7902                                  ;OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
 7903                                  ;
 7904                                  ;CALL
 7905                                  ;       MOV     #DRVNUM,R1           ;DRIVE NUMBER TO R1
 7906                                  ;       JSR     PC,OPT               ;SETUP A COMMAND
 7907                                  ;
 7908  034452  104412                  OPT:    SAVREG                       ;SAVE R0 - R5
 7909  034454  013746  177776                  MOV     @#PS,-(SP)           ;SAVE PROC. STATUS
 7910  034460  146137  033360  033306          BICB    ATABIT(R1),SRCHWT            ;CLEAR "SEARCH WAIT" KEY
 7911  034466  004737  041644                  JSR     PC,GETREQ            ;GET "DPB" POINTER OF REQUEST
 7912  034472  005702                          TST     R2                   ;IS THERE A REQUEST IN QUEUE?
 7913  034474  001505                          BEQ     7$                   ;NO--BRANCH TO EXIT
 7914  034476  032764  004000  000000          BIT     #BIT11,RPCS1(R4)             ;IS DVA SET?
 7915  034504  001407                          BEQ     10$                  ;BRANCH IF NOT
 7916  034506  032764  000100  000012          BIT     #BIT6,RPDS1(R4)      ;IS VV SET ?
 7917  034514  001003                          BNE     10$                  ;BR IF IT IS
 7918  034516  004037  033622          9$:     JSR     R0,DRVINT            ;SEE IF DRIVE STILL ONLINE ?
 7919  034522  000470                          BR      6$                   ;PARITY OR 'DVA' NOT SET
 7920  034524  105761  033244          10$:    TSTB    DRVSTA(R1)           ;IS DRIVE ONLINE?
 7921  034530  003014                          BGT     1$                   ;YES--BRANCH
 7922  034532  004737  041666                  JSR     PC,POPQUE            ;NO--REMOVE REQUEST FROM QUEUE
 7923  034536  012762  140000  000016          MOV     #BIT15!BIT14,16(R2)  ;SET OFFLINE STATUS/ERROR INDICATOR
 7924  034544  105761  033244                  TSTB    DRVSTA(R1)           ;IS DRIVE UNSAFE ?
 7925  034550  100064                          BPL     8$                   ;BR TO EXIT IF NOT
 7926  034552  012762  110000  000016          MOV     #BIT15!BIT12,16(R2)  ;SET UNSAFE STATUS/ERROR INDICATOR
 7927  034560  000460                          BR      8$                   ;BRANCH TO EXIT
 7928  034562  012746  000111          1$:     MOV     #111,-(SP)           ;LOAD COMMAND ONTO THE STACK
 7929  034566  004037  040622                  JSR     R0,WRT.RP            ;LOAD THE REGISTER
 7930  034572  000000                          RPCS1                        ;REGISTER INCREMENT
 7931  034574  034704                          6$                           ;ERROR RETURN ADDRESS
 7932  034576  032714  004000                  BIT     #BIT11,(R4)          ;DRIVE AVAILABLE ?
 7933  034602  001427                          BEQ     5$                   ;BR IF NOT
 7934  034604  122762  000150  000002          CMPB    #150,2(R2)           ;IS THE REQUEST FOR I/O?
 7935  034612  002403                          BLT     2$                   ;YES--BRANCH
 7936  034614  004737  035146                  JSR     PC,CI4               ;CALL THE COMMAND INITIATOR
 7937  034620  000440                          BR      8$                   ;BRANCH TO EXIT
 7938  034622  005737  033356          2$:     TST     DTUW                 ;DATA TRANSFER UNDERWAY?
 7939  034626  002012                          BGE     4$                   ;YES--GO START A SEARCH
 7940  034630  005737  033334                  TST     SEEKFG               ;DO IMPLIED SEEKS?
 7941  034634  100404                          BMI     3$                   ;YES---BRANCH
 7942  034636  004037  036116                  JSR     R0,LA                ;NO--DO LOOK AHEAD
 7943  034642  000427                          BR      8$                   ;RETURN HERE ON A PARITY ERROR
 7944  034644  000403                          BR      4$                   ;GO START A SEARCH
 7945  034646  004737  034732          3$:     JSR     PC,CI1               ;START A DATA TRANSFER
 7946  034652  000423                          BR      8$
 7947  034654  004737  035040          4$:     JSR     PC,CI3               ;START A SEARCH
```

```
                                        BR      8$                      ;GO TO THE EXIT
                                5$:     MOVB    #-1,DPRQS(R1)           ;SET PORT REQUEST INDICATOR
                                        MOV     R1,R3                   ;SET UP TO ADDRESS WORDS
                                        ASL     R3                      ;CONVERT TO WORD INDEX
                                        MOV     #10000.,TIMER(R3)       ;START 10 SEC TIMER
                                        BR      7$                      ;EXIT
                                6$:     JSR     PC,CI7                  ;PROCESS THE PARITY ERROR
                                7$:     BIT     #BIT06,(R4)             ;SEE IF 'IE' ALREADY SET
                                        BNE     8$                      ;BR IF SET
                                        JSR     PC,SET.IE               ;SET "IE" WITHOUT A "TRE"
                                8$:     MOV     (SP)+,@#PS              ;RESTORE PROC. STATUS
                                        RESREG                          ;RESTORE R0 - R5
                                        RTS     PC

                                ;COMMAND INITIATOR
                                ;
                                ;CALL
                                ;       MOV     #DRVNUM,R1              ;DRIVE NUMBER
                                ;       MOV     #DPB,R2                 ;ADDRESS OF DPB
                                ;       JSR     PC,CI?                  ;CI?= CI1,CI3, OR CI4
                                ;WHERE:
                                ;CI1=DATA TRANSFER
                                ;CI2=SEARCH REQUESTED BY DATA XFER
                                ;CI4=NOT DATA TRANSFER

                                CI1:    JSR     PC,POPQUE               ;REMOVE REQUEST FROM "DRIVES WAIT" QUEUE
                                        MOV     R2,TRNSWT               ;PUT REQ. IN TRANSFER WAIT QUEUE
                                        MOV     R2,R3                   ;DPB ADDRESS TO R3
                                        MOV     RPADR,R4                ;RPCS1 ADDRESS
                                        MOV     R1,RPCS2(R4)            ;SELECT DRIVE
                                        ADD     #4,R3                   ;DESIRED WORD COUNT
                                        ADD     #2,R4                   ;RPWC ADDRESS
                                        MOV     (R3)+,(R4)+             ;LOAD WORD COUNT
                                        MOV     (R3)+,(R4)+             ;LOAD BUFFER ADDRESS
                                        MOV     (R3)+,-(SP)             ;LOAD SECTOR AND TRACK
                                        JSR     R0,WRT.RP               ;CALL THE LOAD(WRITE) ROUTINE
                                        RPDA                            ;INDEX OF REGISTER TO LOAD
                                        CI7                             ;ERROR RETURN ADDRESS
                                        MOV     (R3)+,-(SP)             ;LOAD CYLINDER ADDRESS
                                        JSR     R0,WRT.RP
                                        RPCA
                                        CI7
                                        MOV     2(R2),-(SP)             ;LOAD "COMMAND+GO", "A17&A16", AND "PSEL"
                                        JSR     R0,WRT.RP
                                        RPCS1
                                        CI7
                                        MOV     R1,DTUW                 ;SET "DATA TRANSFER UNDERWAY"
                                        JMP     CI5
                                CI3:    MOV     RPADR,R4                ;RPCS1 ADDRESS
                                        MOV     R1,RPCS2(R4)            ;SELECT DRIVE
                                        MOV     12(R2),-(SP)            ;DESIRED CYLINDER ADDRESS
                                        JSR     R0,WRT.RP
                                        RPCA
                                        CI7
                                        MOVB    10(R2),R3               ;PICKUP SECTOR ADDRESS
                                        SUB     MXWNDW,R3               ;BACKUP BY MAX. SEARCH FOR I/O WINDOW
```

```
8004  035074  002002                        BGE     1$
8005  035076  062703  000026                ADC     #22.,R3
8006  035102  010346                1$:     MOV     R3,-(SP)        ;COMBINE THE ADJUSTED SECTOR WITH
8007  035104  116266  000011  000001        MOVB    11(R2),1(SP)    ;THE DESIRED TRACK
8008  035112  004037  040622                JSR     R0,WRT.RP       ;LOAD DESIRED TRACK & SECTOR
8009  035116  000006                        RPDA
8010  035120  035562                        CI7
8011  035122  012746  000131                MOV     #131,-(SP)      ;START A SEARCH
8012  035126  004037  040622                JSR     R0,WRT.RP
8013  035132  000000                        RPCS1
8014  035134  035562                        CI7
8015  035136  156137  033360  033306        BISB    ATABIT(R1),SRCHWT       ;SET "SEARCH WAIT" KEY
8016  035144  000567                        BR      CI5
8017  035146  013704  033372        CI4:    MOV     RPADR,R4        ;RPCS1 ADDRESS
8018  035152  010164  000010                MOV     R1,RPCS2(R4)    ;SELECT DRIVE
8019  035156  116203  000002                MOVB    2(R2),R3        ;PICKUP THE REQUESTED COMMAND
8020  035162  122703  000131                CMPB    #131,R3         ;IS IT A SEARCH COMMAND?
8021  035166  001007                        BNE     1$              ;BRANCH IF NO
8022  035170  016246  000010                MOV     10(R2),-(SP)    ;LOAD DESIRED TRACK & SECTOR
8023  035174  004037  040622                JSR     R0,WRT.RP
8024  035200  000006                        RPDA
8025  035202  035562                        CI7
8026  035204  000403                        BR      2$              ;GO LOAD CYLINDER
8027  035206  122703  000105        1$:     CMPB    #105,R3         ;IS IT A SEEK COMMAND
8028  035212  001007                        BNE     3$              ;BRANCH IF NO
8029  035214  016246  000012        2$:     MOV     12(R2),-(SP)    ;LOAD DESIRED CYLINDER
8030  035220  004037  040622                JSR     R0,WRT.RP
8031  035224  000034                        RPCA
8032  035226  035562                        CI7
8033  035230  000546                        BR      CI6
8034  035232  122703  000115        3$:     CMPB    #115,R3         ;IS IT AN "OFFSET" COMMAND?
8035  035236  001013                        BNE     4$              ;BR IF NO
8036  035240  004037  040446                JSR     R0,RD.RP        ;MERGE THE OFFSET VALUE INTO RPOF
8037  035244  000032                        RPOF                    ;BUT DON'T CHANGE THE UPPER
8038  035246  035562                        CI7
8039  035250  116216  000001                MOVB    1(R2),(SP)      ;BYTE WHEN LOADING THE
8040  035254  004037  040622                JSR     R0,WRT.RP       ;REGISTER (RPOF)
8041  035260  000032                        RPOF
8042  035262  035562                        CI7
8043  035264  000530                        BR      CI6             ;GO START THE COMMAND
8044  035266  122703  000107        4$:     CMPB    #107,R3         ;IS IT A "RECALIBRATE" COMMAND?
8045  035272  001525                        BEQ     CI6             ;BRANCH IF YES
8046  035274  122703  000117                CMPB    #117,R3         ;IS IT A RETURN TO CENTER?
8047  035300  001522                        BEQ     CI6             ;BRANCH IF YES
8048  035302  122703  000103                CMPB    #103,R3         ;IS IT AN "UNLOAD" COMMAND?
8049  035306  001016                        BNE     5$              ;BRANCH IF NO
8050  035310  112761  000001  033234        MOVB    #1,DRVACT(R1)   ;SET THE DRIVE ACTIVE INDICATOR
8051  035316  105061  033244                CLRB    DRVSTA(R1)      ;PUT DRIVE STATUS TO OFFLINE
8052  035322  112761  000001  033312        MOVB    #1,ULDFLG(R1)   ;SET "UNLOAD IN PROGRESS" FLAG
8053  035330  010346                        MOV     R3,-(SP)        ;START THE "UNLOAD" COMMAND
8054  035332  004037  040622                JSR     R0,WRT.RP
8055  035336  000000                        RPCS1
8056  035340  035562                        CI7
8057  035342  000207                        RTS     PC              ;RETURN TO USER
8058  035344  122703  000143        5$:     CMPB    #143,R3         ;IS IT A "SET FORMAT" COMMAND?
8059  035350  001014                        BNE     6$              ;BRANCH IF NO
```

```
8060  035352  004037  040446                      JSR     R0,RD.RP        ;READ THE OFFSET REGISTER
8061  035356  000032                              RPOF
8062  035360  035562                              CI7
8063  035362  116266  000001  000001              MOVB    1(R2),1(SP)     ;COMBINE "FMT22", "ECI", AND "HCI"
8064  035370  004037  040622                      JSR     R0,WRT.RP       ;LOAD "FMT22", "ECI", AND/OR "HCI".
8065  035374  000032                              RPOF
8066  035376  035562                              CI7
8067  035400  000436                              BR      12$
8068  035402  122703  000141          6$:         CMPB    #141,R3         ;IS IT A "GET REGISTER" COMMAND?
8069  035406  001023                              BNE     10$             ;BRANCH IF NO
8070  035410  016203  000006          7$:         MOV     6(R2),R3        ;POINTS TO 1ST ADDRESS OF WHERE
8071                                                                        ;TO PUT THE REGISTER(S)
8072  035414  116237  000010  035432              MOVB    10(R2),9$       ;INIT. THE INDEX FOR THE FIRST REG.
8073  035422  116205  000011                      MOVB    11(R2),R5       ;INDEX OF LAST REG. TO MOVE
8074  035426  004037  040446          8$:         JSR     R0,RD.RP        ;READ RPO4/5/6 REGISTER
8075  035432  000000          9$:     RPCS1                           ;INDEX OF REG. TO READ
8076  035434  035562                              CI7
8077  035436  012623                              MOV     (SP)+,(R3)+     ;GET THE CONTENTS OF RH11/RPO4/5/6 REG.
8078  035440  023705  035432                      CMP     9$,R5           ;LAST REG. BEEN READ?
8079  035444  001414                              BEQ     12$             ;GET OUT IF YES
8080  035446  062737  000002  035432              ADD     #2,9$           ;INCREASE THE INDEX BY 2
8081  035454  000764                              BR      8$              ;LOOP--MORE TO READ
8082  035456  122703  000145          10$:        CMPB    #145,R3         ;IS IT A "SELECT DRIVE" COMMAND?
8083  035462  001405                              BEQ     12$             ;BRANCH IF YES
8084  035464  010346                  11$:        MOV     R3,-(SP)        ;LOAD THE COMMAND
8085  035466  004037  040622                      JSR     R0,WRT.RP
8086  035472  000000                              RPCS1
8087  035474  035562                              CI7
8088  035476  004737  041666          12$:        JSR     PC,POPQUE       ;REMOVE REQ. FROM QUEUE
8089  035502  052762  000200  000016              BIS     #BIT07,16(R2)   ;SET THE "DONE" BIT
8090  035510  005737  033332                      TST     SAVEFG          ;SAVE THE RH11/RPO4/5/6 REGISTERS?
8091  035514  100002                              BPL     13$             ;BRANCH IF NO
8092  035516  004737  041010                      JSR     PC,SVRH11       ;YES--GO SAVE THE REGISTERS
8093  035522  000207                  13$:        RTS     PC              ;RETURN TO USER
8094  035524  006301                  CI5:        ASL     R1
8095  035526  012761  001750  033336              MOV     #1000.,TIMER(R1)        ;SET A ONE SECOND TIMER
8096  035534  006201                              ASR     R1
8097  035536  112761  000001  033234              MOVB    #1,DRVACT(R1)   ;SET THE DRIVE ACTIVE
8098  035544  000207                              RTS     PC              ;RETURN TO THE USER
8099  035546  010346                  CI6:        MOV     R3,-(SP)        ;LOAD THE COMMAND
8100  035550  004037  040622                      JSR     R0,WRT.RP
8101  035554  000000                              RPCS1
8102  035556  035562                              CI7
8103  035560  000761                              BR      CI5
8104  035562  032764  010000  000010  CI7:        BIT     #BIT12,RPCS2(R4)  ;DRIVE NON-EXISTENT ?
8105  035570  001034                              BNE     CI8             ;BR IF YES
8106  035572  005702                  1$:         TST     R2              ;ANYTHING IN QUEUE ?
8107  035574  001405                              BEQ     CI7B            ;BR IF NOT
8108  035576  012762  104000  000016              MOV     #BIT15!BIT11,16(R2)     ;SET "PARITY" ERROR INDICATOR
8109  035604  004737  041010                      JSR     PC,SVRH11       ;GO SAVE THE RH11/RPO4/5/6 REGISTERS
8110  035610  012746  000111          CI7B:       MOV     #111,-(SP)      ;DO A "DRIVE CLEAR"
8111  035614  004037  040622                      JSR     R0,WRT.RP
8112  035620  000000                              RPCS1
8113  035622  035662                              CI8
8114  035624  004737  041550                      JSR     PC,EMPTYQ       ;EMPTY THE QUEUE
8115  035630  105061  033312                      CLRB    ULDFLG(R1)      ;CLEAR THE UNLOAD IN QUEUE FLAG
```

```
8116  035634  105061  033234                  CLRB    DRVACT(R1)          ;DRIVE IS IDLE
8117  035640  020137  033356                  CMP     R1,DTUW             ;IF THIS DRIVE HAD AN I/O REQUEST
8118  035644  001005                          BNE     1$                  ;IN PROGRESS CLEAR ALL OF THE FLAGS
8119  035646  005037  033304                  CLR     TRNSWT
8120  035652  012737  177777  033356          MOV     #-1,DTUW
8121  035660  000207                  1$:     RTS     PC
8122  035662  104412                  CI8:    SAVREG                      ;SAVE R0 - R5
8123  035664  032764  010000  000010          BIT     #BIT12,RPCS2(R4)    ;IS 'NED' SET ?
8124  035672  001002                          BNE     1$                  ;BR IF YES
8125  035674  005001                          CLR     R1
8126  035676  005003                          CLR     R3
8127  035700  105761  033234          1$:     TSTB    DRVACT(R1)          ;DRIVE ACTIVE?
8128  035704  001443                          BEQ     5$                  ;BRANCH IF NO
8129  035706  013702  033304                  MOV     TRNSWT,R2           ;GET THE "TRANSFER WAIT" QUEUE
8130  035712  020137  033356                  CMP     R1,DTUW             ;DID THIS DRIVE HAVE AN I/O IN PROGRESS?
8131  035716  001402                          BEQ     2$                  ;BRANCH IF YES
8132  035720  004737  041644                  JSR     PC,GETREQ           ;GET THE DPB POINTER
8133  035724  005702                  2$:     TST     R2                  ;QUEUE ENTRY FOR DRIVE ?
8134  035726  001415                          BEQ     4$                  ;BR IF NOT
8135  035730  032764  010000  000010          BIT     #BIT12,RPCS2(R4)    ;'NED' SET ?
8136  035736  001404                          BEQ     3$                  ;BR IF NOT
8137  035740  012762  100002  000016          MOV     #BIT15!BIT01,16(R2) ;SET 'DRIVE NON-EXISTENT' INDICATOR
8138  035746  000405                          BR      4$                  ;CONTINUE
8139  035750  012762  102000  000016  3$:     MOV     #BIT15!BIT10,16(R2) ;SET "NON-CLEARABLE PARITY" ERROR INDICATOR
8140  035756  004737  041010                  JSR     PC,SVRH11           ;SAVE RH11/RP04/5/6 REGISTERS
8141  035762  012763  177777  033336  4$:     MOV     #-1,TIMER(R3)       ;STOP THE TIMER
8142  035770  105061  033234                  CLRB    DRVACT(R1)          ;SET "DRIVE ACTIVE" TO IDLE
8143  035774  020137  033356                  CMP     R1,DTUW             ;IS THIS DRIVE SETUP FOR A TRANSFER
8144  036000  001005                          BNE     5$                  ;BR IF NOT
8145  036002  012737  177777  033356          MOV     #-1,DTUW            ;RESET THE INDICATOR
8146  036010  005037  033304                  CLR     TRNSWT              ;CLEAR THE TRANSFER QUEUE
8147  036014  105061  033312          5$:     CLRB    ULDFLG(R1)          ;CLEAR UNLOAD FLAG
8148  036020  032764  010000  000010          BIT     #BIT12,RPCS2(R4)    ;'NED' SET ?
8149  036026  001021                          BNE     6$                  ;BR IF YES
8150  036030  005201                          INC     R1                  ;MOVE TO THE NEXT DRIVE
8151  036032  062703  000002                  ADD     #2,R3
8152  036036  042701  177770                  BIC     #^C7,R1
8153  036042  001316                          BNE     1$                  ;BRANCH IF MORE DRIVES
8154  036044  012737  177777  033356          MOV     #-1,DTUW            ;NO DATA TRANSFERS UNDERWAY
8155  036052  005037  033304                  CLR     TRNSWT              ;CLEAR THE 'TRANSFER WAIT' QUEUE
8156  036056  004737  041472                  JSR     PC,CLRQUE           ;CLEAR ALL OF THE REQUEST QUEUES
8157  036062  012764  000040  000010          MOV     #BIT05,RPCS2(R4)    ;DO A MASSBUS INIT.
8158  036070  000406                          BR      7$                  ;CONTINUE
8159  036072  004737  041550          6$:     JSR     PC,EMPTYQ           ;CLEAR THE DRIVE'S QUEUE
8160  036076  105061  033244                  CLRB    DRVSTA(R1)          ;SET DRIVE TO OFFLINE
8161  036102  105061  033254                  CLRB    DRVTYP(R1)          ;CLEAR THE DRIVE TYPE INDICATOR
8162  036106  004737  041126          7$:     JSR     PC,SET.IE           ;SET "IE" WITHOUT "TRE"
8163  036112  104413                          RESREG                      ;RESTORE R0 - R5
8164  036114  000207                          RTS     PC                  ;RETURN
8165
8166                                  ;LOOK AHEAD ROUTINE
3167                                  ;
8168                                  ;CALL
8169                                  ;       MOV     #DRVNUM,R1          ;DRIVE NUMBER
8170                                  ;       MOV     #DPB,R2             ;POINT TO DPB
8171                                  ;       JSR     R0,LA               ;GO CHECK THE WINDOW
```

```
8172                                        ;       RETURN1                 ;ERROR RETURN
8173                                        ;       RETURN2                 ;START A SEARCH
8174                                        ;       RETURN3                 ;START A DATA TRANSFER
8175
8176  036116  013704  033372          LA:   MOV     RPADR,R4                ;GET RPCS1'S ADDRESS
8177  036122  010164  000010                MOV     R1,RPCS2(R4)            ;SELECT DRIVE
8178  036126  004037  040446                JSR     R0,RD.RP                ;READ CURRENT CYLINDER
8179  036132  000036                        RPCC
8180  036134  036246                        4$                              ;ERROR RETURN ADDRESS
8181  036136  022662  000012                CMP     (SP)+,12(R2)            ;IS CURRENT CYLINDER=DESIRED
8182                                                                        ;CYLINDER?
8183  036142  001037                        BNE     3$                      ;EXIT IF NO
8184  036144  105261  033322                INCB    LACNT(R1)               ;INCREMENT THE LOOK AHEAD COUNT
8185  036150  126137  033322  033400        CMPB    LACNT(R1),MXLACT        ;EXCEED MAX?
8186  036156  003026                        BGT     2$                      ;BRANCH IF YES
8187  036160  116203  000010                MOVB    10(R2),R3               ;GET DESIRED SECTOR ADDRESS AND
8188  036164  000303                        SWAB    R3                      ;MULT. BY 64--ALIGN WITH
8189  036166  006203                        ASR     R3                      ;LOOK AHEAD REGISTER
8190  036170  006203                        ASR     R3
8191  036172  012737  000340  177776        MOV     #340,@#PS               ;PRIORITY LEVEL "7"
8192  036200  004037  040446                JSR     R0,RD.RP                ;READ LOOK AHEAD REGISTER
8193  036204  000020                        RPLA
8194  036206  036246                        4$
8195  036210  162603                        SUB     (SP)+,R3                ;CALCULATE THE DELTA
8196  036212  002002                        BGE     1$
8197  036214  062703  002600                ADD     #<22.*64.>,R3           ;MAKE THE DELTA POSITIVE
8198  036220  023703  033402          1$:   CMP     MXDLTA,R3               ;CHECK THE DELTA TO SEE
8199  036224  002406                        BLT     3$                      ;IF IT IS WITHIN THE
8200  036226  023703  033404                CMP     MNDLTA,R3               ;WINDOW---IF YES, ZERO
8201  036232  002003                        BGE     3$                      ;THE LOOK AHEAD COUNT
8202  036234  105061  033322          2$:   CLRB    LACNT(R1)               ;AND TAKE THE I/O EXIT
8203  036240  005720                        TST     (R0)+
8204  036242  005720                3$:     TST     (R0)+                   ;ADJUST THE RETURN ADDRESS
8205  036244  000402                        BR      5$                      ;EXIT
8206  036246  004737  035562          4$:   JSR     PC,CI7                  ;PROCESS THE ERROR
8207  036252  000200                5$:     RTS     R0                      ;RETURN
8208
8209                                        ;INTERRUPT SERVICE ROUTINE
8210
8211  036254  112737  000001  033310  ISR:  MOVB    #1,ACTDRV               ;SET "ACTIVE DRIVER" FLAG
8212  036262  104412                        SAVREG                          ;SAVE R0 - R5
8213  036264  013704  033372                MOV     RPADR,R4                ;ADDRESS OF RHSCS1
8214  036270  013701  033356                MOV     DTUW,R1                 ;GET "DATA TRANSFER UNDERWAY" INDICATOR
8215  036274  002403                        BLT     1$                      ;BRANCH IF NO DATA TRANSFER UNDERWAY
8216  036276  004737  036320                JSR     PC,TD                   ;CALL TRANSFER DONE
8217  036302  000402                        BR      2$                      ;EXIT
8218  036304  004737  036602          1$:   JSR     PC,SC                   ;CALL SPECIAL CONDITIONS
8219  036310  104413                2$:     RESREG                          ;RESTORE R0 - R5
8220  036312  105037  033310                CLRB    ACTDRV                  ;CLEAR "ACTIVE DRIVER" FLAG
8221  036316  000002                        RTI                             ;RETURN
8222
8223                                        ;TRANSFER DONE ROUTINE
8224
8225  036320  105061  033234          TD:   CLRB    DRVACT(R1)              ;SET DRIVE ACTIVE INDICATOR TO IDLE
8226  036324  012737  177777  033356        MOV     #-1,DTUW                ;NO DATA TRANSFERS UNDERWAY
8227  036332  006301                        ASL     R1
```

```
8228  036334  012761  177777  033336          MOV     #-1,TIMER(R1)   ;CANCEL TIMEOUT
8229  036342  006201                          ASR     R1
8230  036344  013702  033304                  MOV     TRNSWT,R2       ;GET "DPB" ADDRESS FROM THE
8231  036350  005037  033304                  CLR     TRNSWT          ;TRANSFER WAIT QUEUE--CLEAR QUEUE
8232  036354  052762  000200  000016          BIS     #BIT07,16(R2)   ;SET DONE
8233  036362  010164  000010                  MOV     R1,RPCS2(R4)    ;SELECT THE DRIVE
8234  036366  004037  040446                  JSR     R0,RD.RP        ;TRANSFER ERROR(TRE=1)?
8235  036372  000000                          RPCS1
8236  036374  035562                          CI7
8237  036376  006126                          ROL     (SP)+
8238  036400  100421                          BMI     3$              ;BR IF YES
8239  036402  005737  033332                  TST     SAVEFG          ;SAVE THE RH11/RP04/5/6 REGISTERS?
8240  036406  100002                          BPL     1$              ;BRANCH IF NO
8241  036410  004737  041010                  JSR     PC,SVRH11       ;YES--SAVE THE REGISTERS
8242  036414  004737  036474          1$:     JSR     PC,WC           ;SEE IF WRITE CHECK TO BE PUT IN QUEUE
8243  036420  004737  041644                  JSR     PC,GETREQ       ;GET DPB POINTER
8244  036424  005702                          TST     R2              ;ENTRY FOR DRIVE ?
8245  036426  001403                          BEQ     2$              ;BR IF NOT
8246  036430  004737  034452                  JSR     PC,OPT          ;CALL OPTIMIZER
8247  036434  000462                          BR      SC              ;CHECK OTHER DRIVES
8248  036436  012714  000113          2$:     MOV     #113,(R4)       ;RELEASE THE DRIVE
8249  036442  000457                          BR      SC              ;CHECK FOR OTHER DRIVES
8250  036444  052762  100100  000016  3$:     BIS     #BIT15!BIT06,16(R2) ;SET DATA ERROR FLAG
8251  036452  004737  041550                  JSR     PC,EMPTYQ       ;EMPTY THE "DRIVE'S WAIT" QUEUE
8252  036456  004737  041010                  JSR     PC,SVRH11       ;SAVE THE RH11/RP04/5/6 REGISTERS
8253  036462  012714  040111                  MOV     #40111,(R4)     ;ISSUE A "DRIVE CLEAR"
8254  036466  012714  000113                  MOV     #113,(R4)       ;ISSUE A RELEASE TO THE DRIVE
8255  036472  000443                          BR      SC              ;CHECK FOR OTHER DRIVES
8256
8257                                  ;FORCED WRITE CHECK ROUTINE
8258
8259  036474  005737  001424          WC:     TST     AUTOCK          ;AUTOMATIC WRITE CHECKS ?
8260  036500  001437                          BEQ     2$              ;BR IF NOT
8261  036502  122762  000002  000024          CMPB    #2,$CODE(R2)    ;LAST OPERATION WRITE DATA ?
8262  036510  001404                          BEQ     1$              ;BR IF IT WAS
8263  036512  122762  000003  000024          CMPB    #3,$CODE(R2)    ;LAST OPERATION WRITE HEADER & DATA ?
8264  036520  001027                          BNE     2$              ;BR IF NOT
8265  036522  004037  041570          1$:     JSR     R0,DRVQUE       ;PUT THE OPERATION IN THE QUEUE
8266  036526  000424                          BR      2$              ;QUEUE IS FULL
8267  036530  005062  000016                  CLR     16(R2)          ;CLEAR 'DONE' BIT IN DPB
8268  036534  116262  000234  000027          MOVB    $RPCS1(R2),$PREV0(R2)   ;SAVE WRITE OPERATION CODE
8269  036542  016262  000012  000034          MOV     $CYL(R2),$PREVA+2(R2)   ;SAVE CYLINDER
8270  036550  016262  000010  000032          MOV     $SEC(R2),$PREVA(R2)   ;SAVE SECTOR AND TRACK ADDRESSES
8271  036556  142762  000002  000024          BICB    #2,$CODE(R2)    ;CHANGE WRITE TO CHECK
8272  036564  142762  000020  000002          BICB    #20,$COMND(R2)  ;CHANGE DRIVER CODE TO WRITE CHECK
8273  036572  152762  000010  000002          BISB    #10,$COMND(R2)  ;FINISH CHANGING CODE TO WRITE CHECK
8274  036600  000207                  2$:     RTS     PC              ;EXIT
8275
8276                                  ;SPECIAL CONDITION ROUTINE
8277
8278  036602  116403  000016          SC:     MOVB    RPAS(R4),R3     ;READ "RPAS"
8279  036606  001012                          BNE     2$              ;BRANCH IF ANY 'ATA' BITS SET
8280  036610  004037  040446                  JSR     R0,RD.RP        ;READ CONTROL AND STATUS REGISTER
8281  036614  000000                          RPCS1
8282  036616  035662                          CI8
8283  036620  106126                          ROLB    (SP)+           ;IS "IE"=1?
```

```
8284  036622  100403                        BMI     1$              ;YES, NO DRIVES TO CHECK
8285  036624  104001                        ERROR   1               ;REPORT AN ILLEGAL INTERRUPT
8286  036626  004737  041126                JSR     PC,SET.IE       ;SET INTERRUPT ENABLE
8287  036632  000207                1$:     RTS     PC              ;RETURN
8288  036634  005046                2$:     CLR     -(SP)           ;PROCESS ALL DRIVES THAT HAVE
8289  036636  110316                        MOVB    R3,(SP)         ;AN "ATA"=1
8290  036640  012703  000001                MOV     #1,R3
8291  036644  005001                        CLR     R1
8292  036646  030316                SC3:    BIT     R3,(SP)         ;ATA=1?
8293  036650  001005                        BNE     SC5             ;YES--BRANCH
8294  036652  005201                SC4:    INC     R1              ;MOVE TO THE NEXT DRIVE
8295  036654  106303                        ASLB    R3
8296  036656  001373                        BNE     SC3             ;BRANCH IF MORE TO CHECK?
8297  036660  005726                        TST     (SP)+           ;CLEAN OFF THE STACK
8298  036662  000207                        RTS     PC              ;RETURN TO USER
8299  036664  105761  033264        SC5:    TSTB    DPINT(R1)       ;INITIALIZING THE DRIVE ?
8300  036670  001402                        BEQ     1$              ;BR IF NOT
8301  036672  000137  037560                JMP     SC13            ;PROCESS THE DRIVE
8302  036676  105761  033274        1$:     TSTB    DPRQS(R1)       ;PORT REQUEST OUTSTANDING ?
8303  036702  001402                        BEQ     2$              ;BR IF NOT
8304  036704  000137  037560                JMP     SC13            ;START THE OUTSTANDING COMMAND
8305  036710  105761  033244        2$:     TSTB    DRVSTA(R1)      ;CHECK THE DRIVE STATUS
8306  036714  003025                        BGT     5$              ;BRANCH IF ONLINE
8307  036716  105761  033312                TSTB    ULDFLG(R1)      ;UNLOAD IN PROGRESS?
8308  036722  003422                        BLE     5$              ;BRANCH IF NOT
8309  036724  004737  041644                JSR     PC,GETREQ       ;GET DPB POINTER
8310  036730  004737  041010                JSR     PC,SVRH11       ;SAVE THE RH11/RP04/5/6 REGISTERS
8311  036734  004737  037510                JSR     PC,SC12         ;SAVE RPDS1, RPER1, RPER2, AND RPER3
8312                                                                ;ALSO DO A DRIVE INIT (DRVINT)
8313  036740  105761  033244                TSTB    DRVSTA(R1)      ;DID DRIVE COME ONLINE?
8314  036744  003416                        BLE     6$              ;NO---BRANCH
8315  036746  032737  040000  033224        BIT     #BIT14,RPERRS   ;WAS THERE AN ERROR?
8316  036754  001002                        BNE     3$              ;BR IF ERROR
8317  036756  000137  037420                JMP     SC11            ;NO ERROR
8318  036762  013705  033226        3$:     MOV     RPERRS+2,R5     ;YES -- PICKUP RPER1 AND
8319  036766  000476                        BR      SC6A            ;GO PROCESS THE ERROR
8320  036770  105761  033234        5$:     TSTB    DRVACT(R1)      ;DRIVE ACTIVE WITH COMMAND OR ERROR RECOVERY ?
8321  036774  001027                        BNE     SC6             ;BR IF EITHER
8322  036776  004737  037510                JSR     PC,SC12         ;SAVE RPDS1, RPER1, RPER2, AND RPER3
8323                                                                ;ALSO DO A DRVINT
8324  037002  105761  033264        6$:     TSTB    DPINT(R1)       ;TRYING TO INIT THE DRIVE ?
8325  037006  001321                        BNE     SC4             ;BR IF YES, CHECK ON MORE DRIVES
6326  037010  105761  033244                TSTB    DRVSTA(R1)      ;CHECK ON DRIVE'S STATUS
8327  037014  100412                        BMI     7$              ;BR IF UNSAFE
8328  037016  032737  020000  033232        BIT     #BIT13,RPERRS+6 ;ADDRESS PLUG CHANGED ?
8329  037024  001011                        BNE     8$              ;BR IF YES
8330  037026  012746  000113                MOV     #113,-(SP)      ;RELEASE COMMAND
8331  037032  004037  040622                JSR     R0,WRT.RP       ;WRITE THE COMMAND INTO RPCS1
8332  037036  000000                        RPCS1                   ;REGISTER INDEX
8333  037040  037370                        SC8                     ;PARITY EXIT ADDRESS
8334  037042  011605                7$:     MOV     (SP),R5         ;PICKUP (RPAS) BEFORE THE ERROR CALL
8335  037044  104002                        ERROR   2               ;REPORT THE UNEXPECTED ATTENTION
8336  037046  000701                        BR      SC4             ;GO CHECK FOR MORE ATA'S
8337  037050                        8$:
8338  037050  104005                        ERROR   5               ;REPORT THE ADDRESS PLUG CHANGE
8339  037052  000677                        BR      SC4             ;CHECK FOR MORE DRIVES
```

```
8340  037054  006301                        SC6:    ASL     R1                      ;SETUP TO ADDRESS WORDS
8341  037056  012761  177777  033336                MOV     #-1,TIMER(R1)           ;STOP THE TIMER
8342  037064  006201                                ASR     R1                      ;RESTORE THE DRIVE ADDRESS
8343  037066  004737  041644                        JSR     PC,GETREQ               ;GET THE DPB POINTER FROM THE QUEUE
8344  037072  010164  000010                        MOV     R1,RPCS2(R4)            ;SELECT DRIVE
8345  037076  004037  040446                        JSR     R0,RD.RP                ;READ THE RPO4'S STATUS REG.
8346  037102  000012                                RPDS1
8347  037104  037370                                SCB
8348  037106  011605                                MOV     (SP),R5                 ;AND PUT IT IN R5
8349  037110  006126                                ROL     (SP)+                   ;WAS THERE AN ERROR?
8350  037112  100407                                BMI     1$                      ;BR IF ERROR
8351  037114  105761  033234                        TSTB    DRVACT(R1)              ;CHECK DRIVE'S STATE
8352  037120  003137                                BGT     SC11                    ;BR IF DRIVE ACTIVE WITH ORDER
8353  037122  052762  100210  000016                BIS     #BIT15!BIT07!BIT03,16(R2)  ;INFORM USER OF ERROR RECOVER COMPLETION
8354  037130  000470                                BR      SC7
8355  037132  004037  040446                1$:     JSR     R0,RD.RP                ;READ ERROR REGISTER #1
8356  037136  000014                                RPER1
8357  037140  037370                                SCB
8358  037142  012605                                MOV     (SP)+,R5                ;AND SAVE IT IN R5
8359  037144  004737  041010                        JSR     PC,SVRH11               ;SAVE RH11/RPO4/5/6 REGISTERS
8360  037150  012746  000111                        MOV     #111,-(SP)              ;ISSUE A DRIVE CLEAR
8361  037154  004037  040622                        JSR     R0,WRT.RP
8362  037160  000000                                RPCS1
8363  037162  037370                                SCB
8364  037164  006105                        SC6A:   ROL     R5                      ;WAS "UNSAFE" CONDITION =1?
8365  037166  100406                                BMI     1$                      ;BRANCH IF YES
8366  037170  005702                                TST     R2                      ;ANYTHING IN QUEUE ?
8367  037172  001447                                BEQ     SC7                     ;BR IF NOT
8368  037174  052762  100240  000016                BIS     #BIT15!BIT07!BIT05,16(R2)       ;INFORM USER OF ERROR
8369  037202  000443                                BR      SC7
8370  037204  004037  040446                1$:     JSR     R0,RD.RP                ;READ DRIVE STATUS REG. #1
8371  037210  000012                                RPDS1
8372  037212  037370                                SCB
8373  037214  011605                                MOV     (SP),R5                 ;SAVE RPDS1 IN R5
8374  037216  006126                                ROL     (SP)+                   ;"ERR"=1?
8375  037220  100011                                BPL     2$                      ;BR IF NO--UNSAFE CLEARED
8376  037222  112761  177777  033244                MOVB    #-1,DRVSTA(R1)          ;DRIVE IS UNSAFE
8377  037230  004737  041010                        JSR     PC,SVRH11               ;SAVE RH11/RPO4/5/6 REGISTERS
8378  037234  052762  110000  000016                BIS     #BIT15!BIT12,16(R2)     ;INFORM USER OF UNSAFE ERROR
8379  037242  000423                                BR      SC7
8380  037244  032705  010000                2$:     BIT     #BIT12,R5               ;"MOL" = 1 ?
8381  037250  001015                                BNE     3$                      ;BR IF YES
8382  037252  112761  177777  033234                MOVB    #-1,DRVACT(R1)          ;ACTIVE ERROR RECOVER
8383  037260  112761  000001  033244                MOVB    #1,DRVSTA(R1)           ;ONLINE
8384  037266  006301                                ASL     R1
8385  037270  012761  072460  033336                MOV     #30000.,TIMER(R1) ;START 30 SECOND TIMER
8386  037276  006201                                ASR     R1
8387  037300  000137  036652                        JMP     SC4
8388  037304  052762  100220  000016        3$:     BIS     #BIT15!BIT07!BIT04,16(R2)       ;INFORM USER OF ERROR
8389  037312  105061  033234                SC7:    CLRB    DRVACT(R1)              ;DRIVE IS IDLE
8390  037316  004737  041550                        JSR     PC,EMPTYQ               ;DUMP THE QUEUE
8391  037322  105761  033312                        TSTB    ULDFLG(R1)              ;UNLOAD IN PROGRESS OR QUEUE?
8392  037326  003002                                BGT     1$                      ;BR IF NOT
8393  037330  105061  033312                        CLRB    ULDFLG(R1)              ;CLEAR UNLOAD FLAG
8394  037334  116164  033360  000016        1$:     MOVB    ATABIT(R1),RPAS(R4)     ;CLEAR ATTENTION BIT
8395  037342  105761  033244                        TSTB    DRVSTA(R1)              ;IS THE DRIVE UNSAFE ?
```

```
8396  037346  100406                        BMI     2$                ;BR IF IT IS
8397  037350  012746  000113                MOV     #113,-(SP)        ;RELEASE COMMAND
8398  037354  004037  040622                JSR     R0,WRT.RP         ;WRITE THE COMMAND INTO RPCS1
8399  037360  000000                        RPCS1                     ;REGISTER INDEX
8400  037362  037370                        SC8                       ;PARITY EXIT ADDRESS
8401  037364  000137  036652        2$:     JMP     SC4               ;CHECK FOR MORE DRIVES
8402  037370  105761  033234        SC8:    TSTB    DRVACT(R1)        ;IS DRIVE IDLE?
8403  037374  001405                        BEQ     1$                ;YES--BRANCH
8404  037376  004737  041644                JSR     PC,GETREQ         ;GET DPB POINTER
8405  037402  004737  035562                JSR     PC,CI7            ;PROCESS THE PARITY ERROR
8406  037406  000402                        BR      2$                ;CONTINUE
8407  037410  004737  035610        1$:     JSR     PC,CI7B           ;PROCESS THE UNCORRECTABLE PARITY ERROR
8408  037414  000137  036652        2$:     JMP     SC4               ;CHECK MORE DRIVES
8409  037420  105761  033312        SC11:   TSTB    ULDFLG(R1)        ;"UNLOAD IN PROGRESS"?
8410  037424  003402                        BLE     1$                ;BRANCH IF NO
8411  037426  105061  033312                CLRB    ULDFLG(R1)        ;CLEAR UNLOAD FLAG
8412  037432  105061  033234        1$:     CLRB    DRVACT(R1)        ;SET DRIVE IDLE
8413  037436  136137  033360  033306        BITB    ATABIT(R1),SRCHWT ;DOING A SEARCH OPERATION FOR
8414                                                                  ;AN I/O COMMAND?
8415  037444  001012                        BNE     2$                ;BRANCH IF YES
8416  037446  004737  041666                JSR     PC,POPQUE         ;REMOVE REQUEST FROM QUEUE
8417  037452  052762  000200  000016        BIS     #BIT07,16(R2)     ;SET "DONE" BIT
8418  037460  005737  033332                TST     SAVEFG            ;SAVE THE REGISTERS?
8419  037464  100002                        BPL     2$                ;BRANCH IF NO
8420  037466  004737  041010                JSR     PC,SVRH11         ;YES--SAVE ALL OF THE RH11/RP04/5/6 REG'S
8421  037472  116164  033360  000016  2$:   MOVB    ATABIT(R1),RPAS(R4) ;CLEAR ATTENTION BIT
8422  037500  004737  034452                JSR     PC,GPT            ;START A REQUEST
8423  037504  000137  036652                JMP     SC4               ;CHECK FOR MORE DRIVES
8424  037510  010164  000010        SC12:   MOV     R1,RPCS2(R4)      ;SELECT DRIVE
8425  037514  016437  000012  033224        MOV     RPDS1(R4),RPERRS  ;SAVE THE FOUR REGISTERS THAT
8426  037522  016437  000014  033226        MOV     RPER1(R4),RPERRS+2 ;WILL TELL US SOMETHING
8427  037530  016437  000040  033230        MOV     RPER2(R4),RPERRS+4
8428  037536  016437  000042  033232        MOV     RPER3(R4),RPERRS+6
8429  037544  004037  033622                JSR     R0,DRVINT         ;INIT. THE STATE OF THE DRIVE
8430  037550  000401                        BR      1$                ;TAKE ERROR EXIT
8431  037552  000207                        RTS     PC                ;RETURN
8432  037554  005726                1$:     TST     (SP)+             ;POP PC OFF OF THE STACK
8433  037556  000704                        BR      SC8               ;PROCESS THE PARITY ERROR
8434  037560  006301                SC13:   ASL     R1                ;SETUP TO ADDRESS WORDS
8435  037562  012761  177777  033336        MOV     #-1,TIMER(R1)     ;STOP THE TIMER
8436  037570  006201                        ASR     R1                ;
8437  037572  010164  000010                MOV     R1,RPCS2(R4)      ;SELECT THE DRIVE
8438  037576  116164  033360  000016        MOVB    ATABIT(R1),RPAS(R4) ;CLEAR THE ATTENTION BIT
8439  037604  032714  004000                BIT     #BIT11,(R4)       ;DRIVE AVAILABLE ?
8440  037610  001006                        BNE     1$                ;BR IF AVAILABLE
8441  037612  006301                        ASL     R1
8442  037614  012761  023420  033336        MOV     #10000.,TIMER(R1) ;START 10 SEC TIMER AGAIN
8443  037622  006201                        ASR     R1
8444  037624  000433                        BR      3$                ;EXIT
8445  037626  105761  033264        1$:     TSTB    DPINT(R1)         ;INITIALIZING THE DRIVE ?
8446  037632  001424                        BEQ     2$                ;BR IF NOT
8447  037634  105061  033264                CLRB    DPINT(R1)         ;CLEAR THE INIT INDICATOR
8448  037640  004037  033622                JSR     R0,DRVINT         ;GO INIT THE DRIVE
8449  037644  000240                        NOP                       ;DUMMY PARITY ERROR RETURN
8450  037646  105761  033244                TSTB    DRVSTA(R1)        ;DRIVE ONLINE ?
8451  037652  003014                        BGT     2$                ;BR IF YES -- START ORDER
```

```
8452  037654  005702                          TST     R2              ;QUEUE ENTRY FOR THE DRIVE
8453  037656  001416                          BEQ     3$              ;BR IF NOT
8454  037660  004737  041644                  JSR     PC,GETREQ       ;GET DPB ADDRESS
8455  037664  052762  140000  000016          BIS     #BITIS!BIT14,16(R2)  ;INFORM USER THAT DRIVE OFFLINE
8456  037672  004737  041010                  JSR     PC,SVRH11       ;SAVE THE REGISTERS
8457  037676  004737  041550                  JSR     PC,EMPTYQ       ;EMPTY THE REQUEST QUEUE
8458  037702  000404                          BR      3$
8459  037704  105061  033274          2$:     CLRB    DPRQS(R1)       ;CLEAR THE PORT REQUEST INDICATOR
8460  037710  004737  034452                  JSR     PC,OPT          ;START THE PENDING REQUEST
8461  037714  000137  036652          3$:     JMP     SC4             ;PROCESS OTHER DRIVES
8462
8463                                  ;RP04/5/6 TIMER ROUTINE
8464                                  ;CALL
8465                                  ;       MOV     #TIME,-(SP)     ;ELASPED TIME IN MILLISECONDS ON THE STACK
8466                                  ;       JSR     PC,RPTMR        ;CALL RP04/5/6 TIME ROUTINE
8467
8468  037720  005737  033310          RPTMR:  TST     ACTDRV          ;CHECK "ACTDRV & ACTSTR"
8469  037724  001030                          BNE     4$              ;IF NON ZERO EXIT
8470  037726  112737  000001  033311          MOVB    #1,ACTSTR       ;SET "ACTSTR"
8471  037734  104412                          SAVREG                  ;SAVE R0 - R5
8472  037736  005001                          CLR     R1              ;START WITH DRIVE 0
8473  037740  005003                          CLR     R3
8474  037742  005763  033336          1$:     TST     TIMER(R3)       ;IS THE TIMER RUNNING?
8475  037746  002407                          BLT     2$              ;BRANCH IF NO
8476  037750  166663  000002  033336          SUB     2(SP),TIMER(R3) ;COUNT THE INTERVAL
8477  037756  003003                          BGT     2$              ;BR IF NO SOFTWARE TIMEOUT
8478  037760  004737  040012                  JSR     PC,STO          ;CALL SOFTWARE TIMEOUT ROUTINE
8479  037764  000405                          BR      3$              ;GO TO THE EXIT
8480  037766  005201          2$:     INC     R1              ;MOVE TO NEXT DRIVE
8481  037770  005723                          TST     (R3)+
8482  037772  022701  000010                  CMP     #8.,R1          ;OUT OF DRIVES?
8483  037776  003361                          BGT     1$              ;BRANCH IF NO
8484  040000  104413          3$:     RESREG                  ;RESTORE R0 - R5
8485  040002  105037  033311                  CLRB    ACTSTR          ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
8486  040006  012616          4$:     MOV     (SP)+,(SP)      ;ADJUST THE STACK
8487  040010  000207                          RTS     PC              ;RETURN
8488
8489                                  ;SOFTWARE TIMEOUT ROUTINE
8490                                  ;
8491                                  ;NOTE: THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
8492                                  ;       OR GREATER
8493                                  ;
8494                                  ;CALL:  STO
8495                                  ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER
8496                                  ;       JSR     PC,STO          ;CALL
8497                                  ;       RETURN
8498
8499  040012  010146          STO:    MOV     R1,-(SP)        ;SAVE R1
8500  040014  010346                  MOV     R3,-(SP)        ;SAVE R3
8501  040016  013704  033372          MOV     RPADR,R4        ;GET ADDRESS OF "RPCS1"
8502  040022  010164  000010          MOV     R1,RPCS2(R4)    ;SELECT THE DRIVE
8503  040026  004037  040446          JSR     R0,RD.RP        ;READ "DRIVE STATUS REG"
8504  040032  000012                  RPDS!
8505  040034  040334                  STOS
8506  040036  105726                  TSTB    (SP)+           ;IS "DRY"=1?
8507  040040  100477                  BMI     ST02            ;BR IF YES
```

```
8508  040042  105761  033264          ST01:   TSTB    DPINT(R1)         ;TRYING TO INTIALIZE THE DRIVE ?
8509  040046  001074                  BNE     ST02              ;BR IF YES
8510  040050  105761  033274          TSTB    DPRQS(R1)         ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
8511  040054  001071                  BNE     ST02              ;BR IF YES
8512  040056  013702  033304          MOV     TRNSWT,R2         ;PICKUP TRANSFER WAIT QUEUE
8513  040062  020137  033356          CMP     R1,DTUW           ;TRANSFER UNDERWAY ON THIS DRIVE?
8514  040066  001402                  BEQ     1$                ;BRANCH IF YES
8515  040070  004737  041644          JSR     PC,GETREQ         ;GET DPB ADDRESS
8516  040074  052762  101000  000016  1$:     BIS     #BIT15!BIT09,16(R2) ;SET THE ERROR FLAGS
8517  040102  004737  041010          JSR     PC,SVRH11         ;SAVE RH11/RP04/5/6 REGISTERS
8518  040106  012764  000040  000010          MOV     #BIT05,RPCS2(R4) ;"INIT" THE MASS BUS
8519  040114  105061  033234          CLRB    DRVACT(R1)        ;DRIVE IS IDLE
8520  040120  105061  033312          CLRB    ULDFLG(R1)        ;CLEAR THE UNLOAD FLAG
8521  040124  005001                  CLR     R1                ;START WITH DRIVE 0
8522  040126  005003                  CLR     R3
8523  040130  004037  033622          2$:     JSR     R0,DRVINT         ;INIT. THIS DRIVE
8524  040134  000477                  BR      ST05              ;PARITY ERROR RETURN
8525  040136  105761  033234          TSTB    DRVACT(R1)        ;DRIVE IDLE BEFORE THE INIT.?
8526  040142  001414                  BEQ     4$                ;YES--BRANCH
8527  040144  013702  033304          MOV     TRNSWT,R2         ;GET TRANSFER WAIT QUEUE
8528  040150  023701  033356          CMP     DTUW,R1           ;WAS THERE I/O ON THIS DRIVE?
8529  040154  001402                  BEQ     3$                ;YES--BRANCH
8530  040156  004737  041644          JSR     PC,GETREQ         ;GET THE DPB POINTER FROM QUEUE
8531  040162  052762  100400  000016  3$:     BIS     #BIT15!BIT08,16(R2) ;INFORM USER OF INIT.
8532  040170  105061  033234          CLRB    DRVACT(R1)        ;SET DRIVE ACTIVE TO IDLE
8533  040174  105061  033312          4$:     CLRB    ULDFLG(R1)        ;NO UNLOAD
8534  040200  012763  177777  033336          MOV     #-1,TIMER(R3)     ;STOP THE TIMER
8535  040206  005723                  TST     (R3)+             ;UPDATE THE INDEX
8536  040210  005201                  INC     R1                ;INCREMENT THE DRIVE NUMBER
8537  040212  022701  000010          CMP     #8.,R1            ;LAST DRIVE BEEN CHECKED?
8538  040216  003344                  BGT     2$                ;NO--LOOP
8539  040220  012737  177777  033356          MOV     #-1,DTUW          ;NO DATA TRANSFERS UNDERWAY
8540  040226  005037  033304          CLR     TRNSWT            ;CLEAR TRANSFER WAIT QUEUE
8541  040232  004737  041472          JSR     PC,CLRQUE         ;CLEAR ALL REQUEST QUEUES
8542  040236  000500                  BR      ST09              ;EXIT
8543  040240  116405  000016          ST02:   MOVB    RPAS(R4),R5       ;READ ATTENTION REG
8544  040244  136105  033360          BITB    ATABIT(R1),R5     ;IS ATTENTION FOR THIS DRIVE UP ?
8545  040250  001017                  BNE     ST03              ;YES--BRANCH
8546  040252  105761  033264          TSTB    DPINT(R1)         ;TRYING TO INTIALIZE THE DRIVE ?
8547  040256  001031                  BNE     ST06              ;BR IF YES - DRIVE NOT ONLINE
8548  040260  105761  033274          TSTB    DPRQS(R1)         ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
8549  040264  001045                  BNE     ST07              ;BR IF YES - NO RESPONSE TO REQUEST
8550  040266  020137  033356          CMP     R1,DTUW           ;DATA TRANSFER UNDERWAY FOR THIS DRIVE
8551  040272  001263                  BNE     ST01              ;BR IF NO
8552  040274  004037  040446          JSR     R0,RD.RP          ;YES--CHECK "RDY"
8553  040300  000000                  RPCS1
8554  040302  040334                  ST05
8555  040304  105726                  TSTB    (SP)+
8556  040306  100255                  BPL     ST01              ;BR IF "RDY"=0
8557  040310  105761  033264          ST03:   TSTB    DPINT(R1)         ;INITIALIZING THE DRIVE ?
8558  040314  001003                  BNE     1$                ;BR IF INIT PENDING
8559  040316  105761  033274          TSTB    DPRQS(R1)         ;PORT REQUEST PENDING ?
8560  040322  001446                  BEQ     ST09              ;BR IF NOT
8561  040324  012763  177777  033336  1$:     MOV     #-1,TIMER(R3)     ;STOP THE TIMER
8562  040332  000442                  BR      ST09              ;EXIT
8563  040334  004737  035662          ST05:   JSR     PC,CI8            ;GO HANDLE THE PARITY ERROR
```

```
8564  040340  000437                          BR      ST09
8565  040342  105061  033264          ST06:   CLRB    DPINT(R1)       ;CLEAR THE INITIALIZE INDICATOR
8566  040346  105061  033244                  CLRB    DRVSTA(R1)      ;SET UNIT OFFLINE
8567  040352  012763  177777  033336          MOV     #-1,TIMER(R3)   ;STOP THE TIMER
8568  040360  004737  041644                  JSR     PC,GETREQ       ;GET THE DPB ADDRESS
8569  040364  005702                          TST     R2              ;REQUEST IN QUEUE ?
8570  040366  001424                          BEQ     ST09            ;BR IF NOT
8571  040370  052762  140000  000016          BIS     #BIT15!BIT14,16(R2)  ;INFORM THE USER DRIVE NOT AVAILABLE
8572  040376  000414                          BR      ST08            ;FINISH
8573  040400  012763  177777  033336  ST07:   MOV     #-1,TIMER(R3)   ;STOP THE TIMER
8574  040406  105061  033274                  CLRB    DPRQS(R1)       ;CLEAR PORT REQUEST INDICATOR
8575  040412  004737  041644                  JSR     PC,GETREQ       ;GET DPB ADDRESS
8576  040416  005702                          TST     R2              ;QUEUE ENTRY FOR DRIVE ?
8577  040420  001407                          BEQ     ST09            ;BR IF NONE
8578  040422  012762  100004  000016          MOV     #BIT15!BIT2,16(R2)  ;INFORM USER OF PORT REQUEST ERROR
8579  040430  004737  041550          ST08:   JSR     PC,EMPTYQ       ;CLEAR THE QUEUE FOR THE DRIVE
8580  040434  004737  041010                  JSR     PC,SVRH11       ;SAVE THE REGISTERS
8581  040440  012603                  ST09:   MOV     (SP)+,R3        ;RESTORE R3
8582  040442  012601                          MOV     (SP)+,R1        ;RESTORE R1
8583  040444  000207                          RTS     PC              ;RETURN
8584
8585                                  ;ROUTINE TO READ A RH11/RP04/5/6 REGISTER
8586                                  ;
8587                                  ;CALL
8588                                  ;       JSR     R0,RD.RP        ;GO READ A REGISTER
8589                                  ;       INDEX                   ;REG. INDEX FROM BASE
8590                                  ;       ERRADR                  ;ERROR ADDRESS--PROCESS ERROR STARTING
8591                                  ;       RETURN                  ;AT THIS ADDRESS
8592                                  ;                               ;CONTENTS OF REG. IS ON THE STACK
8593                                  ;
8594  040446  013737  033370  040610  RD.RP:   MOV     MCPEMX,RD.RP2   ;MAX. RETRYS ALLOWED
8595  040454  011646                          MOV     (SP),-(SP)      ;SAVE R0 FOR RETURN
8596  040456  013737  033372  040472          MOV     RPADR,RD.ADR    ;FORM THE DESIRED ADDRESS
8597  040464  062037  040472                  ADD     (R0)+,RD.ADR    ;USING THE BASE AND THE INDEX
8598  040470  013727                  RD.RP1: MOV     @(PC)+,(PC)+    ;READ THE DESIRED REGISTER OF THE RP04
8599  040472  000000                  RD.ADR: .WORD   0               ;ADDRESS IS FORMED HERE
8600  040474  000000                  RD.WRD: .WORD   0               ;REG. CONTENTS PUT HERE
8601  040476  013766  040474  000002          MOV     RD.WRD,2(SP)    ;RETURN IT TO THE USER
8602  040504  013746  033372                  MOV     RPADR,-(SP)     ;PUT THE ADDRESS ON THE STACK
8603  040510  062716  000010                  ADD     #RPCS2,(SP)     ;FORM THE ADDRESS OF RPCS2
8604  040514  032736  010000                  BIT     #BIT12,@(SP)+   ;CHECK THE 'NED' BIT
8605  040520  001035                          BNE     RD.RP3          ;BR IF DRIVE NON-EXISTENT
8606  040522  017746  172644                  MOV     @RPADR,-(SP)    ;READ RPCS1
8607  040526  032716  020000                  BIT     #BIT13,(SP)     ;DID MCPE SET?
8608  040532  001002                          BNE     1$              ;BRANCH IF YES
8609  040534  022620                          CMP     (SP)+,(R0)+     ;ADJUST FOR RETURN
8610  040536  000430                          BR      RD.RP4          ;EXIT
8611  040540                          1$:
8612  040540  104003                          ERROR   3               ;REPORT "MCPE" ERROR
8613  040542  005737  033356                  TST     DTUW            ;DATA TRANSFER UNDERWAY?
8614  040546  100405                          BMI     2$              ;NO--BRANCH
8615  040550  032716  040000                  BIT     #BIT14,(SP)     ;NO--"TRE"=1?
8616  040554  001402                          BEQ     2$              ;NO--BRANCH
8617  040556  005726                          TST     (SP)+           ;YES--CLEAN OFF THE STACK AND
8618  040560  000415                          BR      RD.RP3          ;TAKE THE FATAL ERROR EXIT
8619  040562  052716  040000          2$:     BIS     #BIT14,(SP)     ;CLEAR "MCPE" BY SENDING A "1" TO "TRE"
```

```
8620  040566  000316                        SWAB    (SP)                ;POSITION BEFORE WRITING
8621  040570  013737  033372  040604        MOV     RPADR,3$            ;FORM ADDRESS OF HIGH BYTE
8622  040576  005237  040604                INC     3$
8623  040602  112637                        MOVB    (SP)+,@(PC)+        ;WRITE THE HIGH BYTE OF RPCS1
8624  040604  000000                3$:     .WORD   0                   ;ADDRESS STORAGE
8625  040606  005327                        DEC     (PC)+               ;EXCEEDED MAX. RETRYS
8626  040610  000003                RD.RP2: .WORD   3
8627  040612  002326                        BGE     RD.RP1              ;BRANCH IF NO
8628  040614  011000                RD.RP3: MOV     (R0),R0             ;FATAL ERROR EXIT
8629  040616  012616                        MOV     (SP)+,(SP)
8630  040620  000200                RD.RP4: RTS     R0
8631
8632                                ;ROUTINE TO WRITE A REGISTER
8633                                ;
8634                                ;CALL
8635                                ;       MOV     DATA,-(SP)          ;DATA TO BE LOADED ON THE STACK
8636                                ;       JSR     R0,WRT.RP           ;CALL THE ROUTINE TO LOAD(WRITE) THE REG.
8637                                ;       INDEX                       ;INDEX OF THE REGISTER TO BE LOADED
8638                                ;       ERRADR                      ;ADDRESS TO RETURN TO ON AN ERROR
8639                                ;       RETURN                      ;ERROR FREE RETURN
8640
8641  040622  013737  033370  040772  WRT.RP: MOV     MCPEMX,WRT.R2       ;MAX RETRYS ALLOWED
8642  040630  016637  000002  040710          MOV     2(SP),WRT.WD        ;SAVE THE WORD TO WRITE
8643  040636  012616                        MOV     (SP)+,(SP)          ;ADJUST THE STACK
8644  040640  012037  040712                MOV     (R0)+,WRT.AD        ;GET INDEX OF REGISTER TO BE WRITTEN
8645  040644  001015                        BNE     1$                  ;BRANCH IF NOT RPCS1
8646  040646  122737  000150  040710        CMPB    #150,WRT.WD         ;IS THE COMMAND FOR DATA TRANSFERS?
8647  040654  002411                        BLT     1$                  ;YES--DON'T GET THE OLD A16 & A17, & PSEL
8648  040656  004037  040446                JSR     R0,RD.RP            ;NO---COMBINE A16&A17, & PSEL WITH
8649  040662  000000                        RPCS1                       ;THE COMMAND BEFORE SENDING IT TO
8650  040664  041000                        WRT.R3                      ;THE RH11/RP04
8651  040666  000316                        SWAB    (SP)
8652  040670  042716  177770                BIC     #^C7,(SP)
8653  040674  112637  040711                MOVB    (SP)+,WRT.WD+1
8654  040700  063737  033372  040712  1$:   ADD     RPADR,WRT.AD        ;FORM THE ADDRESS OF THE DISK REG.
8655  040706  012737                WRT.R1: MOV     (PC)+,@(PC)+        ;LOAD THE DESIRED REG.
8656  040710  000000                WRT.WD: .WORD   0                   ;WORD TO WRITE GOES HERE
8657  040712  000000                WRT.AD: .WORD   0                   ;ADDRESS IS FORMED HERE
8658  040714  013746  033372                MOV     RPADR,-(SP)         ;PUT THE ADDRESS ON THE STACK
8659  040720  062716  000010                ADD     #RPCS2,(SP)         ;FORM THE ADDRESS OF RPCS2
8660  040724  032736  010000                BIT     #BIT12,@(SP)+       ;CHECK THE 'NED' BIT
8661  040730  001023                        BNE     WRT.R3              ;BR IF DRIVE NON-EXISTENT
8662  040732  004037  040446                JSR     R0,RD.RP            ;CHECK FOR PARITY ERROR ON WRITE
8663  040736  000014                        RPER1
8664  040740  041000                        WRT.R3
8665  040742  032726  000010                BIT     #BIT03,(SP)+
8666  040746  001416                        BEQ     WRT.R4              ;BRANCH IF "PAR=0"
8667  040750  016037  177776  040762        MOV     -2(R0),1$           ;PICKUP THE INDEX
8668  040756  004037  040446                JSR     R0,RD.RP            ;READ THE REG.
8669  040762  000000                1$:     .WORD   0                   ;REG. INDEX
8670  040764  041000                        WRT.R3                      ;RETURN TO THIS ADDRESS ON ERROR
8671  040766  104004                        ERROR   4                   ;REPORT THE PARITY ON WRITE ERROR
8672  040770  005327                        DEC     (PC)+               ;DECREMENT THE ERROR COUNT
8673  040772  000003                WRT.R2: .WORD   3                   ;RETRY COUNTER
8674  040774  002344                        BGE     WRT.R1              ;TRY AGAIN IF NOT FINISHED
8675  040776    5726                        TST     (SP)+               ;CLEAN OFF THE ST
```

```
 8676  041000  011000                  WRT.R3: MOV     (R0),R0         ;TAKE THE "PARITY ON WRITE" ERROR EXIT
 8677  041002  000401                          BR      WRT.R5          ;EXIT
 8678  041004  005720                  WRT.R4: TST     (R0)+           ;ADJUST FOR ERROR FREE EXIT
 8679  041006  000200                  WRT.R5: RTS     R0
 8680
 8681                                  ;ROUTINE TO SAVE THE RH11/RP04/5/6 REGISTERS AS PER DPB+14
 8682                                  ;
 8683                                  ;CALL
 8684                                  ;       MOV     #DPBNUM,R2      ;DPB POINTER TO R2
 8685                                  ;       JSR     PC,SVRH11       ;SAVE THE DRIVES REG'S
 8686
 8687  041010  104412                  SVRH11: SAVREG                  ;SAVE R0 - R5
 8688  041012  005702                          TST     R2              ;QUEUE ENTRY FOR THE DRIVE ?
 8689  041014  001430                          BEQ     4$              ;BR IF NONE
 8690  041016  013704  033372                  MOV     RPADR,R4
 8691  041022  111264  000010                  MOVB    (R2),RPCS2(R4)  ;SELECT DRIVE
 8692  041026  016203  000014                  MOV     14(R2),R3       ;GET THE ERROR TABLE POINTER
 8693  041032  001433                          BEQ     6$              ;EXIT IF NO ADDRESS
 8694  041034  005037  041070                  CLR     3$              ;COUNTER & POINTER
 8695  041040  023727  041070  000022  1$:     CMP     3$,#RPDB        ;REACHED THE BUFFER REGISTER ?
 8696  041046  001006                          BNE     2$              ;BR IF NOT
 8697  041050  032764  000200  000010          BIT     #BIT07,RPCS2(R4)  ;'OR' SET ?
 8698  041056  001002                          BNE     2$              ;BR IF SET
 8699  041060  005023                          CLR     (R3)+           ;STORE RPDB AS ZEROES
 8700  041062  000405                          BR      4$              ;CONTINUE
 8701  041064  004037  040446          2$:     JSR     R0,RD.RP        ;READ THE SELECTED REGISTER
 8702  041070  000000                  3$:     .WORD   0               ;REGISTER INDEX
 8703  041072  041116                          5$                      ;ERROR RETURN ADDRESS
 8704  041074  012623                          MOV     (SP)+,(R3)+     ;STORE THE REGISTER CONTENTS
 8705  041076  023727  041070  000046  4$:     CMP     3$,#RPEC2       ;REACHED THE END ?
 8706  041104  001406                          BEQ     6$              ;BR IF YES
 8707  041106  062737  000002  041070          ADD     #2,3$           ;INCREMENT THE REGISTER INDEX
 8708  041114  000751                          BR      1$              ;CONTINUE READING THE REGISTERS
 8709  041116  004737  035562          5$:     JSR     PC,CI7          ;PROCESS THE UNCORRECTABLE PARITY ERROR
 8710  041122  104413                  6$:     RESREG                  ;RESTORE R0 - R5
 8711  041124  000207                          RTS     PC              ;RETURN
 8712
 8713                                  ;ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A "TRE"
 8714                                  ;CALL
 8715                                  ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
 8716                                  ;       JSR     PC,SET.IE       ;SET "IE"
 8717                                  ;       RETURN
 8718
 8719  041126  010446                  SET.IE: MOV     R4,-(SP)        ;SAVE R4
 8720  041130  013704  033372                  MOV     RPADR,R4        ;PICKUP ADDRESS OF RPCS1
 8721  041134  010164  000010                  MOV     R1,RPCS2(R4)    ;SELECT DRIVE
 8722  041140  011446                          MOV     (R4),-(SP)      ;READ RPCS1
 8723  041142  052716  040000                  BIS     #BIT14,(SP)     ;SET THE "TRE" BIT OF THE WORD READ
 8724  041146  000316                          SWAB    (SP)            ;ADJUST FOR DATO
 8725  041150  112714  000100                  MOVB    #BIT06,(R4)     ;SET "IE"
 8726  041154  032764  010000  000010          BIT     #BIT12,RPCS2(R4) ;IS "NED"=1?
 8727  041162  001002                          BNE     1$              ;YES--CLEAR "TRE"
 8728  041164  005726                          TST     (SP)+           ;CLEAN OFF THE STACK
 8729  041166  000402                          BR      2$
 8730  041170  112664  000001          1$:     MOVB    (SP)+,1(R4)     ;CLEAR "TRE"
 8731  041174  012604                  2$:     MOV     (SP)+,R4        ;RESTORE R4
```

```
8732  041176  000207                  RTS     PC              ;RETURN TO CALLER
8733
8734                                  ;QUEUE COUNT
8735  041200      000                 QCNT:   .BYTE   0       ;DRIVE 0
8736  041201      000                         .BYTE   0       ;DRIVE 1
8737  041202      000                         .BYTE   0       ;DRIVE 2
8738  041203      000                         .BYTE   0       ;DRIVE 3
8739  041204      000                         .BYTE   0       ;DRIVE 4
8740  041205      000                         .BYTE   0       ;DRIVE 5
8741  041206      000                         .BYTE   0       ;DRIVE 6
8742  041207      000                         .BYTE   0       ;DRIVE 7
8743
8744                                  ;QUEUE INPUT POINTERS
8745
8746  041210  041272                  QINPT:  .WORD   QDRV0    ;DRIVE 0
8747  041212  041312                          .WORD   QDRV1    ;DRIVE 1
8748  041214  041332                          .WORD   QDRV2    ;DRIVE 2
8749  041216  041352                          .WORD   QDRV3    ;DRIVE 3
8750  041220  041372                          .WORD   QDRV4    ;DRIVE 4
8751  041222  041412                          .WORD   QDRV5    ;DRIVE 5
8752  041224  041432                          .WORD   QDRV6    ;DRIVE 6
8753  041226  041452                          .WORD   QDRV7    ;DRIVE 7
8754
8755                                  ;QUEUE OUTPUT POINTERS
8756
8757  041230  041272                  QOUTPT: .WORD   QDRV0    ;DRIVE 0
8758  041232  041312                          .WORD   QDRV1    ;DRIVE 1
8759  041234  041332                          .WORD   QDRV2    ;DRIVE 2
8760  041236  041352                          .WORD   QDRV3    ;DRIVE 3
8761  041240  041372                          .WORD   QDRV4    ;DRIVE 4
8762  041242  041412                          .WORD   QDRV5    ;DRIVE 5
8763  041244  041432                          .WORD   QDRV6    ;DRIVE 6
8764  041246  041452                          .WORD   QDRV7    ;DRIVE 7
8765
8766  041250  041272                  QSTART: .WORD   QDRV0    ;DRIVE 0 START ADDRESS
8767  041252  041312                  QSTOP:  .WORD   QDRV1    ;DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
8768  041254  041332                          .WORD   QDRV2    ;STOP DRIVE 1--START DRIVE 2
8769  041256  041352                          .WORD   QDRV3    ;STOP DRIVE 2--START DRIVE 3
8770  041260  041372                          .WORD   QDRV4    ;STOP DRIVE 3--START DRIVE 4
8771  041262  041412                          .WORD   QDRV5    ;STOP DRIVE 4--START DRIVE 5
8772  041264  041432                          .WORD   QDRV6    ;STOP DRIVE 5--START DRIVE 6
8773  041266  041452                          .WORD   QDRV7    ;STOP DRIVE 6--START DRIVE 7
8774  041270  041472                          .WORD   QTERM    ;STOP DRIVE 7
8775
8776                                  ;DRIVE REQUEST QUEUES
8777
8778  041272  000010                  QDRV0:  .BLKW   10
8779  041312  000010                  QDRV1:  .BLKW   10
8780  041332  000010                  QDRV2:  .BLKW   10
8781  041352  000010                  QDRV3:  .BLKW   10
8782  041372  000010                  QDRV4:  .BLKW   10
8783  041412  000010                  QDRV5:  .BLKW   10
8784  041432  000010                  QDRV6:  .BLKW   10
8785  041452  000010                  QDRV7:  .BLKW   10
8786          041472                  QTERM=.
8787
```

```
8788                                    ;ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
8789                                    ;
8790                                    ;CALL
8791                                    ;       JSR     PC,CLRQUE
8792
8793  041472  104412                    CLRQUE: SAVREG                          ;SAVE R0 - R5
8794  041474  012702  041200                    MOV     #QCNT,R2                ;ZERO THE QUEUE COUNTS
8795  041500  005022                            CLR     (R2)+                   ;DRIVES 0 & 1
8796  041502  005022                            CLR     (R2)+                   ;DRIVES 2 & 3
8797  041504  005022                            CLR     (R2)+                   ;DRIVES 4 & 5
8798  041506  005022                            CLR     (R2)+                   ;DRIVES 6 & 7
8799  041510  012703  000010                    MOV     #8.,R3                  ;MOVE THE STARTING
8800  041514  012701  041250                    MOV     #QSTART,R1              ;ADDRESS OF THE QUEUE INTO
8801  041520  012122                    1$:     MOV     (R1)+,(R2)+             ;THE QUEUE INPUT POINTER
8802  041522  005303                            DEC     R3
8803  041524  001375                            BNE     1$
8804  041526  012703  000010                    MOV     #8.,R3                  ;MOVE THE STARTING ADDRESS
8805  041532  012701  041250                    MOV     #QSTART,R1              ;OF THE QUEUE INTO THE
8806  041536  012122                    2$:     MOV     (R1)+,(R2)+             ;QUEUE OUTPUT POINTER
8807  041540  005303                            DEC     R3
8808  041542  001375                            BNE     2$
8809  041544  104413                            RESREG                          ;RESTORE R0 - R5
8810  041546  000207                            RTS     PC
8811
8812                                    ;EMPTY THE QUEUE SPECIFIED BY R1
8813                                    ;
8814                                    ;CALL
8815                                    ;       MOV     DRVNUM,R1               ;DRIVE NUMBER TO R1
8816                                    ;       JSR     PC,EMPTYQ
8817
8818  041550  105061  041200             EMPTYQ: CLRB    QCNT(R1)                ;CLEAR NUMBER OF ITEMS IN QUEUE
8819  041554  006301                            ASL     R1
8820  041556  016161  041210  041230            MOV     QINPT(R1),QOUTPT(R1) ;SET OUTPUT QUEUE POINTER=INPUT POINTER
8821  041564  006201                            ASR     R1
8822  041566  000207                            RTS     PC
8823
8824                                    ;ROUTINE TO PUT A REQUEST IN QUEUE
8825                                    ;
8826                                    ;CALL
8827                                    ;       MOV     #DRVNUM,R1              ;DRIVE NUMBER
8828                                    ;       MOV     #DPB,R2                 ;ADDRESS OF PARAMETER BLOCK
8829                                    ;       JSR     R0,DRVQUE               ;GO PUT REQUEST IN QUEUE
8830                                    ;       RETURN1                         ;RETURN HERE IF QUEUE IS FULL
8831                                    ;       RETURN2                         ;RETURN HERE IF REQUEST IS IN QUEUE
8832
8833  041570  122761  000010  041200  DRVQUE: CMPB    #10,QCNT(R1)            ;IS QUEUE FULL?
8834  041576  001421                            BEQ     2$                      ;BR IF YES-TAKE RETURN1
8835  041600  105261  041200                    INCB    QCNT(R1)                ;INCREMENT QUEUE COUNT
8836  041604  006301                            ASL     R1
8837  041606  010271  041210                    MOV     R2,@QINPT(R1)           ;PUT THIS REQUEST IN QUEUE
8838  041612  062761  000002  041210            ADD     #2,QINPT(R1)            ;UPDATE THE QUEUE POINTER
8839  041620  026161  041210  041252            CMP     QINPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER
8840  041626  001003                            BNE     1$                      ;BRANCH IF NO
8841  041630  016161  041250  041210            MOV     QSTART(R1),QINPT(R1) ;YES--RESET POINTER
8842  041636  006201                    1$:     ASR     R1
8843  041640  005720                            TST     (R0)+                   ;TAKE RETURN 2
```

```
8844  041642  000200                  2$:     RTS     R0              ;RETURN TO USER
8845
8846                                  ;ROUTINE TO GET THE "DPB" ADDRESS OF NEXT REQUEST IN QUEUE
8847                                  ;
8848                                  ;CALL
8849                                  ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
8850                                  ;       JSR     PC,GETREQ       ;GO GET THE REQUEST
8851                                  ;       RETURN                  ;R2="DPB" ADDRESS OF THE REQUEST
8852                                  ;                               ;R2=0 IF NO REQUEST IN QUEUE
8853
8854  041644  005002                  GETREQ: CLR     R2
8855  041646  105761  041200                  TSTB    QCNT(R1)        ;IS THERE ANY REQUEST IN QUEUE?
8856  041652  001404                          BEQ     2$              ;NO---BRANCH
8857  041654  006301                  1$:     ASL     R1
8858  041656  017102  041230                  MOV     @QOUTPT(R1),R2  ;PICKUP "DPB" POINTER FOR THIS DRIVE
8859  041662  006201                          ASR     R1
8860  041664  000207                  2$:     RTS     PC              ;RETURN TO USER
8861
8862                                  ;ROUTINE TO "POP" THE REQUEST FROM QUEUE
8863                                  ;
8864                                  ;CALL
8865                                  ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
8866                                  ;       JSR     PC,POPQUE       ;CALL TO REMOVE REQUEST
8867                                  ;       RETURN                  ;R2=ADDRESS OF DPB REMOVED
8868
8869  041666  105361  041200          POPQUE: DECB    QCNT(R1)        ;DECREMENT QUEUE COUNT
8870  041672  006301                          ASL     R1
8871  041674  017102  041230                  MOV     @QOUTPT(R1),R2  ;GET THE "DPB" POINTER
8872  041700  062761  000002  041230          ADD     #2,QOUTPT(R1)   ;UPDATE THE QUEUE POINTER
8873  041706  026161  041230  041252          CMP     QOUTPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER?
8874  041714  001003                          BNE     1$              ;NO--BRANCH TO EXIT
8875  041716  016161  041250  041230          MOV     QSTART(R1),QOUTPT(R1) ;YES--RESET THE POINTER
8876  041724  006201                  1$:     ASR     R1
8877  041726  000207                          RTS     PC              ;RETURN TO USER
8878
8879
8880                                  ;;*****************************************************************
8881
8882                                  .SBTTL  DATA, CONTROL, & STATUS BLOCKS
8883
8884                                  ;;*****************************************************************
8885
8886                                  ;BLOCK LOCATION EQUATE STATEMENTS
8887
8888          000001                  $FMT    =       1               ;FMT,HCI,ECI OR OFFSET CODE
8889          000002                  $COMND  =       $FMT+1          ;OPERATION CODE
8890          000003                  $PSEL   =       $FMT+2          ;PORT SELECT & BITS A16, A17
8891          000004                  $WRDM   =       $FMT+3          ;WORD COUNT (2'S COMP)
8892          000006                  $BUF    =       $FMT+5          ;BUFFER ADDR OR REGISTER TABLE POINTER
8893          000010                  $SEC    =       $FMT+7          ;SECTOR ADDRESS OR 1ST REG ADDR
8894          000011                  $TRK    =       $FMT+10         ;TRACK ADDRESS OF LAST REG ADDR
8895          000012                  $CYL    =       $FMT+11         ;CYLINDER ADDR
8896          000014                  $REG    =       $FMT+13         ;REGISTER STORAGE (IF ERROR)
8897          000016                  $STATUS =       $FMT+15         ;STATUS WORD (SET BY DRIVER)
8898
8899                                  ;DRIVE'S HISTORY AND CURRENT INDICATOR STORAGE EQUATES
```

```
8900
8901            000020          $WRDL   =       $FMT+17         ;WORD COUNT (NOT 2'S COMP)
8902            000022          $SSEC   =       $WRDL+2         ;SECTOR SIZE FOR CURRENT OPERATION
8903            000024          $CODE   =       $WRDL+4         ;PRESENT COMMAND SELECTION CODE
8904            000026          $PACK   =       $WRDL+6         ;WRITE DATA PACK INDICATOR
8905            000027          $PREVO  =       $WRDL+7         ;PREVIOUS COMMAND SELECTION CODE
8906            000030          $PATTC  =       $WRDL+10        ;PATTERN CODE
8907            000032          $PREVA  =       $WRDL+12        ;PREVIOUS ADDRESS - TRACK, SECTOR, CYLINDER
8908            000036          $OPERC  =       $WRDL+16        ;OPERATION COUNT
8909            000042          $POSIT  =       $WRDL+22        ;SEEK COUNT
8910            000046          $TRANS  =       $WRDL+26        ;TOTAL BITS XFERED COUNT (R & W)
8911            000052          $READ   =       $WRDL+32        ;TOTAL BITS READ COUNT
8912            000056          $TOTAL  =       $WRDL+36        ;TOTAL ERRORS (ALL TYPES) COUNT
8913            000060          $SOFT   =       $WRDL+40        ;'SOFT' ERROR COUNT
8914            000062          $HARD   =       $WRDL+42        ;'HARD' ERROR COUNT
8915            000064          $SKI    =       $WRDL+44        ;'SKI' OR 'OCYL' ERROR COUNT
8916            000066          $MISPO  =       $WRDL+46        ;PROG DETECTED MISPOSITIONING ERROR S COUNT
8917            000070          $PASSC  =       $WRDL+50        ;PASS COUNTER
8918            000072          $FAIR   =       $WRDL+52        ;OPERATION QUEUE 'FAIRNESS' COUNT
8919
8920                            ;INDEX EQUATES TO THE NEXT OPERATION PARAMETERS
8921
8922            000074          $NCODE  =       $WRDL+54        ;NEXT OPERATION CODE
8923            000075          $NPATC  =       $NCODE+1        ;NEXT PATTERN
8924            000076          $NSEC   =       $NCODE+2        ;NEXT SECTOR
8925            000077          $NTRK   =       $NCODE+3        ;NEXT TRACK
8926            000100          $NCYL   =       $NCODE+4        ;NEXT CYLINDER
8927            000102          $NWRDL  =       $NCODE+6        ;NEXT BUFFER SIZE
8928            000104          $NEXT   =       $NCODE+10       ;PARAMETER SELECTION INDICATOR
8929
8930                            ;INDEX EQUATES FOR MAXIMUM/MINIMUM ADDRESSES
8931
8932            000106          MAXCYL  =       $NCODE+12       ;MAXIMUM CYLINDER ADDRESS
8933            000110          MINCYL  =       MAXCYL+2        ;MINIMUM CYLINDER ADDRESS
8934            000112          MAXTRK  =       MAXCYL+4        ;MAXIMUM TRACK ADDRESS
8935            000114          MINTRK  =       MAXCYL+6        ;MINIMUM TRACK ADDRESS
8936            000116          MAXSEC  =       MAXCYL+10       ;MAXIMUM SECTOR ADDRESS
8937            000120          MINSEC  =       MAXCYL+12       ;MINIMUM SECTOR ADDRESS
8938            000122          $FIRST  =       MAXCYL+14       ;FIRST OPERATION INDICATOR
8939
8940                            ;BAD SECTOR/TRACK ADDRESS STORAGE AREA INDEX EQUATE
8941
8942            000124          $BDSEC  =       MAXCYL+16       ;BAD SECTOR STORAGE TABLE
8943
8944                            ;DRIVE ID AREA INDEX EQUATE
8945
8946            000224          $DRVID  =       $BDSEC+100      ;DRIVE ID
8947
8948                            ;RH11/RP04/5/6 REGISTER EQUATES
8949
8950            000234          $RPCS1  =       $DRVID+10       ;RP04 REGISTER STORAGE
8951            000236          $RPWC   =       $RPCS1+2
8952            000240          $RPBA   =       $RPCS1+4
8953            000242          $RPDA   =       $RPCS1+6
8954            000244          $RPCS2  =       $RPCS1+10
8955            000246          $RPDS1  =       $RPCS1+12
```

```
 8956            000250                  $RPER1  =       $RPCS1+14
 8957            000252                  $RPAS   =       $RPCS1+16
 8958            000254                  $RPLA   =       $RPCS1+20
 8959            000256                  $RPDB   =       $RPCS1+22
 8960            000260                  $RPMR   =       $RPCS1+24
 8961            000262                  $RPDT   =       $RPCS1+26
 8962            000264                  $RPSN   =       $RPCS1+30
 8963            000266                  $RPOF   =       $RPCS1+32
 8964            000270                  $RPCA   =       $RPCS1+34
 8965            000272                  $RPCC   =       $RPCS1+36
 8966            000274                  $RPER2  =       $RPCS1+40
 8967            000276                  $RPER3  =       $RPCS1+42
 8968            000300                  $RPEC1  =       $RPCS1+44
 8969            000302                  $RPEC2  =       $RPCS1+46
 8970
 8971
 8972                                    ;BLOCK FOR DRIVE 0
 8973
 8974  041730      000       000         DRIVE0: .BYTE   0,0             ;DRIVE NUMBER
 8975  041732  000005                            .BLKW   5
 8976  041744  042164                            .WORD   .+$RPCS1-$REG
 8977  041746  000266                            .BLKB   $RPEC2-$REG
 8978
 8979
 8980                                    ;BLOCK FOR DRIVE 1
 8991
 8982  042234      001       000         DRIVE1: .BYTE   1,0             ;DRIVE NUMBER
 8983  042236  000005                            .BLKW   5
 8984  042250  042470                            .WORD   .+$RPCS1-$REG
 8985  042252  000266                            .BLKB   $RPEC2-$REG
 8986
 8987
 8988                                    ;BLOCK FOR DRIVE 2
 8989
 8990  042540      002       000         DRIVE2: .BYTE   2,0             ;DRIVE NUMBER
 8991  042542  000005                            .BLKW   5
 8992  042554  042774                            .WORD   .+$RPCS1-$REG
 8993  042556  000266                            .BLKB   $RPEC2-$REG
 8994
 8995
 8996                                    ;BLOCK FOR DRIVE 3
 8997
 8998  043044      003       000         DRIVE3: .BYTE   3,0             ;DRIVE NUMBER
 8999  043046  000005                            .BLKW   5
 9000  043060  043300                            .WORD   .+$RPCS1-$REG
 9001  043062  000266                            .BLKB   $RPEC2-$REG
 9002
 9003
 9004                                    ;BLOCK FOR DRIVE 4
 9005
 9006  043350      004       000         DRIVE4: .BYTE   4,0             ;DRIVE NUMBER
 9007  043352  000005                            .BLKW   5
 9008  043364  043604                            .WORD   .+$RPCS1-$REG
 9009  043366  000266                            .BLKB   $RPEC2-$REG
 9010
 9011
```

```
9012                                    ;BLOCK FOR DRIVE 5
9013
9014  043654     005     000          DRIVE5: .BYTE   5,0             ;DRIVE NUMBER
9015  043656  000005                          .BLKW   5
9016  043670  044110                          .WORD   .+$RPCS1-$REG
9017  043672  000266                          .BLKB   $RPEC2-$REG
9018
9019
9020                                    ;BLOCK FOR DRIVE 6
9021
9022  044160     006     000          DRIVE6: .BYTE   6,0             ;DRIVE NUMBER
9023  044162  000005                          .BLKW   5
9024  044174  044414                          .WORD   .+$RPCS1-$REG
9025  044176  000266                          .BLKB   $RPEC2-$REG
9026
9027
9028                                    ;BLOCK FOR DRIVE 7
9029
9030  044464     007     000          DRIVE7: .BYTE   7,0             ;DRIVE NUMBER
9031  044466  000005                          .BLKW   5
9032  044500  044720                          .WORD   .+$RPCS1-$REG
9033  044502  000266                          .BLKB   $RPEC2-$REG
9034
9035
9036                                    ;GENERAL PURPOSE DPB - USED BY 'READHD', 'RECALT', 'OFFSET',&'RTNCTR'
9037
9038  044770  000000  000000  177774  GENDPB: .WORD   0,0,-4,CYLDER
9039  044776  054516
9040  045000  000000  000000  045010          .WORD   0,0,GENREG,0
9041  045006  000000
9042  045010  000024                  GENREG: .BLKW   24              ;REGISTER STORAGE IF ERROR
9043
9044                                    ;;*****************************************************************
9045
9046                                    .SBTTL  ERROR MESSAGES
9047
9048                                    ;;*****************************************************************
9049
9050  045060  044122  030461  044440  EM1:     .ASCIZ  /RH11 INTERRUPT OCCURRED (RPAS = 0)/
9051  045066  052116  051105  052522
9052  045074  052120  047440  041503
9053  045102  051125  042522  020104
9054  045110  051050  040520  020123
9055  045116  020075  024460     000
9056
9057  045123     125  042516  050130  EM2:     .ASCIZ  /UNEXPECTED ATTENTION OCCURRED/
9058  045130  041505  042524  020104
9059  045136  052101  042524  052116
9060  045144  047511  020116  041517
9061  045152  052503  051122  042105
9062  045160     000
9063
9064  045161     115  051501  041123  EM3:     .ASCIZ  /MASSBUS PARITY ERROR (MCPE=1)/
9065  045166  051525  050040  051101
9066  045174  052111  020131  051105
9067  045202  047522  020122  046450
```

```
9068  045210  050103  036505  024461
9069  045216     000
9070
9071  045217     115  051501  041123  EM4:    .ASCIZ  /MASSBUS PARITY ERROR (PAR=1)/
9072  045224  051525  050040  051101
9073  045232  052111  020131  051105
9074  045240  047522  020122  050050
9075  045246  051101  030475  000051
9076
9077  045254  042101  051104  051505  EM5:    .ASCIZ  /ADDRESS PLUG CHANGE BIT SET/
9078  045262  020123  046120  043525
9079  045270  041440  040510  043516
9080  045276  020105  044502  020124
9081  045304  042523  000124
9082
9083  045310  044122  030461  042040  EM6:    .ASCIZ  /RH11 DIDN'T RESPOND TO ADDRESSING/
9084  045316  042111  023516  020124
9085  045324  042522  050123  047117
9086  045332  020104  047524  040440
9087  045340  042104  042522  051523
9088  045346  047111  000107
9089
9090  045352  047125  047503  051122  EM10:   .ASCIZ  /UNCORRECTABLE MASSBUS PARITY ERROR/
9091  045360  041505  040524  046102
9092  045366  020105  040515  051523
9093  045374  052502  020123  040520
9094  045402  044522  054524  042440
9095  045410  051122  051117     000
9096
9097  045415     106  052101  046101  EM11:   .ASCIZ  /FATAL MASSBUS PARITY ERROR/
9098  045422  046440  051501  041123
9099  045430  051525  050040  051101
9100  045436  052111  020131  051105
9101  045444  047522  000122
9102
9103  045450  042520  051522  051511  EM12:   .ASCIZ  /PERSISTENT DEVICE UNSAFE/
9104  045456  042524  052116  042040
9105  045464  053105  041511  020105
9106  045472  047125  040523  042506
9107  045500     000
9108
9109  045501     117  042520  040522  EM13:   .ASCIZ  /OPERATION NOT COMPLETED WITHIN TIME LIMIT/
9110  045506  044524  047117  047040
9111  045514  052117  041440  046517
9112  045522  046120  052105  042105
9113  045530  053440  052111  044510
9114  045536  020116  044524  042515
9115  045544  046040  046511  052111
9116  045552     000
9117
9118  045553     104  044522  042526  EM14:   .ASCIZ  /DRIVE WENT OFFLINE/
9119  045560  053440  047105  020124
9120  045566  043117  046106  047111
9121  045574  000105
9122
9123  045576  047516  051040  051505  EM15:   .ASCIZ  /NO RESPONSE TO PORT REQUEST/
```

```
9124  045604  047520  051516  020105
9125  045612  047524  050040  051117
9126  045620  020124  042522  052521
9127  045626  051505  000124
9128
9129  045632  042510  042101  051105  EM20:   .ASCIZ  /HEADER CRC ERROR/
9130  045640  041440  041522  042440
9131  045646  051122  051117     000
9132
9133  045653     104  052101  020101  EM21:   .ASCIZ  /DATA CHECK ('DCK') ERROR/
9134  045660  044103  041505  020113
9135  045666  023450  041504  023513
9136  045674  020051  051105  047522
9137  045702  000122
9138
9139  045704  051127  052111  020105  EM22:   .ASCIZ  /WRITE CHECK ERROR - DATA CHECK ('DCK') SET/
9140  045712  044103  041505  020113
9141  045720  051105  047522  020122
9142  045726  020055  040504  040524
9143  045734  041440  042510  045503
9144  045742  024040  042047  045503
9145  045750  024447  051440  052105
9146  045756     000
9147
9148  045757     127  044522  042524  EM23:   .ASCIZ  /WRITE CHECK ERROR - DATA CHECK ('DCK') NOT SET/
9149  045764  041440  042510  045503
9150  045772  042440  051122  051117
9151  046000  026440  042040  052101
9152  046006  020101  044103  041505
9153  046014  020113  023450  041504
9154  046022  023513  020051  047516
9155  046030  020124  042523  000124
9156
9157  046036  042510  042101  051105  EM24:   .ASCIZ  /HEADER READ ERROR - 'FMT' BIT DROPPED
9158  046044  051040  040505  020104
9159  046052  051105  047522  020122
9160  046060  020055  043047  052115
9161  046066  020047  044502  020124
9162  046074  051104  050117  042520
9163  046102  000104
9164
9165  046104  042510  042101  051105  EM25:   .ASCIZ  /HEADER READ ERROR - HEADER COMPARE ('HCE') ERROR/
9166  046112  051040  040505  020104
9167  046120  051105  047522  020122
9168  046126  020055  042510  042101
9169  046134  051105  041440  046517
9170  046142  040520  042522  024040
9171  046150  044047  042503  024447
9172  046156  042440  051122  051117
9173  046164     000
9174
9175  046165     106  051117  040515  EM26:   .ASCIZ  /FORMAT ERROR ('FER')/
9176  046172  020124  051105  047522
9177  046200  020122  023450  042506
9178  046206  023522  000051
9179
```

```
 9180  046212  042510  042101  051105  EM27:   .ASCIZ  /HEADER COMPARE ('HCE') ERROR/
 9181  046220  041440  046517  040520
 9182  046226  042522  024040  044047
 9183  046234  042503  024447  042440
 9184  046242  051122  051117     000
 9185
 9186  046247     115  051511  042503  EM30:   .ASCIZ  /MISCELLANEOUS DRIVE ERROR/
 9187  046254  046114  047101  047505
 9188  046262  051525  042040  044522
 9189  046270  042526  042440  051122
 9190  046276  051117     000
 9191
 9192  046301     117  042520  040522  EM31:   .ASCIZ  /OPERATION INCOMPLETE ('OPI') ERROR/
 9193  046306  044524  047117  044440
 9194  046314  041516  046517  046120
 9195  046322  052105  020105  023450
 9196  046330  050117  023511  020051
 9197  046336  051105  047522  000122
 9198
 9199  046344  051104  053111  020105  EM32:   .ASCIZ  /DRIVE TIMING ('DTE') ERROR/
 9200  046352  044524  044515  043516
 9201  046360  024040  042047  042524
 9202  046366  024447  042440  051122
 9203  046374  051117     000
 9204
 9205  046377     120  051101  052111  EM33:   .ASCIZ  /PARITY ('PAR') ERROR AFTER OPERATION STARTED/
 9206  046404  020131  023450  040520
 9207  046412  023522  020051  051105
 9208  046420  047522  020122  043101
 9209  046426  042524  020122  050117
 9210  046434  051105  052101  047511
 9211  046442  020116  052123  051101
 9212  046450  042524  000104
 9213
 9214  046454  051127  052111  020105  EM34:   .ASCIZ  /WRITE CLOCK FAILURE ('WCF') ERROR/
 9215  046462  046103  041517  020113
 9216  046470  040506  046111  051125
 9217  046476  020105  023450  041527
 9218  046504  023506  020051  051105
 9219  046512  047522  000122
 9220
 9221  046516  047111  040526  044514  EM35:   .ASCIZ  /INVALID ADDRESS ('IAE') ERROR/
 9222  046524  020104  042101  051104
 9223  046532  051505  020123  023450
 9224  046540  040511  023505  020051
 9225  046546  051105  047522  000122
 9226
 9227  046554  051127  052111  020105  EM36:   .ASCIZ  /WRITE LOCK ('WLE') ERROR/
 9228  046562  047514  045503  024040
 9229  046570  053447  042514  024447
 9230  046576  042440  051122  051117
 9231  046604     000
 9232
 9233  046605     104  052101  020101  EM37:   .ASCIZ  /DATA CHECK ('DCK') SET DURING WRITE CHECK COMMAND/
 9234  046612  044103  041505  020113
 9235  046620  023450  041504  023513
```

```
 9236  046626  020051  042523  020124
 9237  046634  052504  044522  043516
 9238  046642  053440  044522  042524
 9239  046650  041440  042510  045503
 9240  046656  041440  046517  040515
 9241  046664  042116     000
 9242
 9243  046667     122  030510  020061  EM40:   .ASCIZ  /RH11 OR UNIBUS TRANSFER ERROR/
 9244  046674  051117  052440  044516
 9245  046702  052502  020123  051124
 9246  046710  047101  043123  051105
 9247  046716  042440  051122  051117
 9248  046724     000
 9249
 9250  046725     102  051525  040440  EM41:   .ASCIZ  /BUS ADDRESS OR WORD COUNT INCORRECT/
 9251  046732  042104  042522  051523
 9252  046740  047440  020122  047527
 9253  046746  042122  041440  052517
 9254  046754  052116  044440  041516
 9255  046762  051117  042522  052103
 9256  046770     000
 9257
 9258  046771     104  052101  020101  EM42:   .ASCIZ  /DATA COMPARE ERRORS - NO OTHER ERROR(S) DETECTED/
 9259  046776  047503  050115  051101
 9260  047004  020105  051105  047522
 9261  047012  051522  026440  047040
 9262  047020  020117  052117  042510
 9263  047026  020122  051105  047522
 9264  047034  024122  024523  042040
 9265  047042  052105  041505  042524
 9266  047050  000104
 9267
 9268  047052  040503  023516  020124  EM43:   .ASCIZ  /CAN'T MATCH DATA READ WITH A PATTERN/
 9269  047060  040515  041524  020110
 9270  047066  040504  040524  051040
 9271  047074  040505  020104  044527
 9272  047102  044124  040440  050040
 9273  047110  052101  042524  047122
 9274  047116     000
 9275
 9276  047117     105  051122  051117  EM44:   .ASCIZ  /ERROR BIT(S) SET, BUT NO ERROR SIGNALED BY THE RH11/
 9277  047124  041040  052111  051450
 9278  047132  020051  042523  026124
 9279  047140  041040  052125  047040
 9280  047146  020117  051105  047522
 9281  047154  020122  044523  047107
 9282  047162  046101  042105  041040
 9283  047170  020131  044124  020105
 9284  047176  044122  030461     000
 9285
 9286  047203     105  041503  046040  EM45:   .ASCIZ  /ECC LOGIC FAILURE - POSITION REGISTER VALUE TOO LARGE.
 9287  047210  043517  041511  043040
 9288  047216  044501  052514  042522
 9289  047224  026440  050040  051517
 9290  047232  052111  047511  020116
 9291  047240  042522  044507  052123
```

```
 9292  047246  051105  053040  046101
 9293  047254  042525  052040  047517
 9294  047262  046040  051101  042507
 9295  047270     000
 9296
 9297  047271     102  051525  040440  EM46:   .ASCIZ  /BUS ADDRESS AND WORD COUNT NOT CONSISTENT/
 9298  047276  042104  042522  051523
 9299  047304  040440  042116  053440
 9300  047312  051117  020104  047503
 9301  047320  047125  020124  047516
 9302  047326  020124  047503  051516
 9303  047334  051511  042524  052116
 9304  047342     000
 9305
 9306  047343     123  042505  020113  EM50:   .ASCIZ  /SEEK INCOMPLETE ('SKI') OR OFF CYLINDER ('OCYL') ERROR/
 9307  047350  047111  047503  050115
 9308  047356  042514  042524  024040
 9309  047364  051447  044513  024447
 9310  047372  047440  020122  043117
 9311  047400  020106  054503  044514
 9312  047406  042116  051105  024040
 9313  047414  047447  054503  023514
 9314  047422  020051  051105  047522
 9315  047430  000122
 9316
 9317  047432  051120  043517  040522  EM51:   .ASCIZ  /PROGRAM DETECTED POSITIONING ERROR/
 9318  047440  020115  042504  042524
 9319  047446  052103  042105  050040
 9320  047454  051517  052111  047511
 9321  047462  044516  043516  042440
 9322  047470  051122  051117     000
 9323
 9324  047475     104  044522  042526  EM60:   .ASCIZ  /DRIVE UNSAFE ERROR/
 9325  047502  052440  051516  043101
 9326  047510  020105  051105  047522
 9327  047516  000122
 9328
 9329  047520  050122  051501     000  DH1:    .ASCIZ  /RPAS/
 9330
 9331  047525     104  044522  042526  DH2:    .ASCIZ  /DRIVE   RPDS1   RPER1   RPER2   RPER3   RPAS/
 9332  047532  020040  051040  042120
 9333  047540  030523  020040  051040
 9334  047546  042520  030522  020040
 9335  047554  051040  042520  031122
 9336  047562  020040  051040  042520
 9337  047570  031522  020040  051040
 9338  047576  040520  000123
 9339
 9340  047602  051104  053111  020105  DH3:    .ASCIZ  /DRIVE   REG ADR  DATA/
 9341  047610  020040  042522  020107
 9342  047616  042101  020122  042040
 9343  047624  052101  000101
 9344
 9345  047630  051104  053111  020105  DH4:    .ASCIZ  /DRIVE   REG ADR    GOOD    BAD/
 9346  047636  020040  042522  020107
 9347  047644  042101  020122  020040
```

```
9348  047652  043440  047517  020104
9349  047660  020040  041040  042101
9350  047666     000
9351
9352  047667     044  050122  042101  DH6:    .ASCIZ  /$RPADR/
9353  047674  000122
9354
9355  047676  051104  020126  050122  DH14:   .ASCII  /DRV RPCS1   RPCS2   RPDS1   RPER1   /
9356  047704  051503  020061  020040
9357  047712  050122  051503  020062
9358  047720  020040  050122  051504
9359  047726  020061  020040  050122
9360  047734  051105  020061  020040
9361  047742  050122  051105  020062          .ASCIZ  /RPER2   RPER3   RPEC1   RPEC2/<CR><LF>
9362  047750  020040  050122  051105
9363  047756  020063  020040  050122
9364  047764  041505  020061  020040
9365  047772  050122  041505  006462
9366  050000  000012
9367
9368  050002  050122  041527  020040  DH15:   .ASCII  /RPWC    RPBA    RPDA    RPAS    RPLA    /
9369  050010  020040  050122  040502
9370  050016  020040  020040  050122
9371  050024  040504  020040  020040
9372  050032  050122  051501  020040
9373  050040  020040  050122  040514
9374  050046  020040  020040
9375  050052  050122  041104  020040          .ASCIZ  /RPDB    RPMR    RPDT/<CR><LF>
9376  050060  020040  050122  051115
9377  050066  020040  020040  050122
9378  050074  052104  005015     000
9379
9380  050101     122  051520  020116  DH16:   .ASCIZ  /RPSN    RPOF    RPCA    RPCC    STATUS/<CR><LF>
9381  050106  020040  051040  047520
9382  050114  020106  020040  051040
9383  050122  041520  020101  020040
9384  050130  051040  041520  020103
9385  050136  020040  051440  040524
9386  050144  052524  006523  000012
9387
9388                                    .EVEN
9389
9390  050152  001244  000000          DT1:    .WORD   ATTN,0
9391
9392  050156  001242  033224  033226  DT2:    .WORD   DRIVE,RPERRS,RPERRS+2,RPERRS+4,RPERRS+6,ATTN,0
9393  050164  033230  033232  001244
9394  050172  000000
9395
9396  050174  001242  040472  040474  DT3:    .WORD   DRIVE,RD.ADR,RD.WRD,0
9397  050202  000000
9398
9399  050204  001242  040712  040710  DT4:    .WORD   DRIVE,WRT.AD,WRT.WD,RD.WRD,0
9400  050212  040474  000000
9401
9402  050216  001170  000000          DT6:    .WORD   $RPADR,0
9403
```

```
9404  050222  000234  000244  000246  DT14:   .WORD   $RPCS1,$RPCS2,$RPDS1,$RPER1,$RPER2,$RPER3,$RPEC1,$RPEC2,0
9405  050230  000250  000274  000276
9406  050236  000300  000302  000000
9407
9408  050244  000236  000240  000242  DT15:   .WORD   $RPWC,$RPBA,$RPDA,$RPAS,$RPLA,$RPDB,$RPMR,$RPDT,0
9409  050252  000252  000254  000256
9410  050260  000260  000262  000000
9411
9412  050266  000264  000266  000270  DT16:   .WORD   $RPSN,$RPOF,$RPCA,$RPCC,$STATUS,0
9413  050274  000272  000016  000000
9414
9415  050302  051120  051505  047105  LIN2C:  .ASCIZ  /PRESENT ORDER = /
9416  050310  020124  051117  042504
9417  050316  020122  020075     000
9418  050323     040  050040  042522  LIN2P:  .ASCIZ  /  PREVIOUS ORDER = /
9419  050330  044526  052517  020123
9420  050336  051117  042504  020122
9421  050344  020075     000
9422  050347     052  042440  051122  LIN2S:  .ASCIZ  @* ERROR AT BAD TRACK/SECTOR@
9423  050354  051117  040440  020124
9424  050362  040502  020104  051124
9425  050370  041501  027513  042523
9426  050376  052103  051117     000
9427  050403     105  051122  051117  LINM3:  .ASCIZ  /ERROR AT C/
9428  050410  040440  020124  000103
9429  050416  052040     000          T:      .ASCIZ  / T/
9430  050421     120  042522  042523  LINN3:  .ASCIZ  /PRESENT ADDR = C/
9431  050426  052116  040440  042104
9432  050434  020122  020075  000103
9433  050442  051440     000          S:      .ASCIZ  / S/
9434  050445     040  020040  051120  LINP3:  .ASCIZ  /   PREV ADDR = C/
9435  050452  053105  040440  042104
9436  050460  020122  020075  000103
9437  050466  052123  051101  020124  LINS3:  .ASCIZ  /START CYL = /
9438  050474  054503  020114  020075
9439  050502     000
9440  050503     040  042440  042116  LINEN3: .ASCIZ  /  END CYL = /
9441  050510  041440  046131  036440
9442  050516  000040
9443  050520  020040  041501  052524  LINA3:  .ASCIZ  /  ACTUAL CYL = /
9444  050526  046101  041440  046131
9445  050534  036440  000040
9446  050540  020040  051124  020113  LINT3:  .ASCIZ  /  TRK = /
9447  050546  020075     000
9448  050551     040  050122  040503  LINCA3: .ASCIZ  / RPCA = /
9449  050556  036440  000040
9450  050562  050122  040504  036440  LINDA3: .ASCIZ  /RPDA = /
9451  050570  000040
9452  050572  050122  040502  036440  LINB3:  .ASCIZ  /RPBA = /
9453  050600  000040
9454  050602  020040  050122  041527  LINW3:  .ASCIZ  /  RPWC = /
9455  050610  036440  000040
9456  050614  052123  051101  020124  LINST3: .ASCIZ  /START TRK = /
9457  050622  051124  020113  020075
9458  050630     000
9459  050631     123  040524  052122  LINSS3: .ASCIZ  /START SEC = /
```

```
9460  050636  051440  041505  036440
9461  050644  000040
9462  050646  052502  043106  051105  LINM4:  .ASCIZ  /BUFFER ADDR = /
9463  050654  040440  042104  020122
9464  050662  020075     000
9465  050665     040  051440  055111  LINS4:  .ASCIZ  /  SIZE = /
9466  050672  020105  020075     000
9467  050677     040  040440  052103  LINX4:  .ASCIZ  /  ACTUAL NMBR WRDS XFRD = /
9468  050704  040525  020114  046516
9469  050712  051102  053440  042122
9470  050720  020123  043130  042122
9471  050726  036440  000040
9472  050732  047507  042117  042040  LIND5:  .ASCIZ  /GOOD DATA = /
9473  050740  052101  020101  020075
9474  050746     000
9475  050747     040  041040  042101  LINB5:  .ASCIZ  /  BAD DATA = /
9476  050754  042040  052101  020101
9477  050762  020075     000
9478  050765     040  051440  041505  LINP5:  .ASCIZ  /  SECT POS = /
9479  050772  020124  047520  020123
9480  051000  020075     000
9481  051003     110  040505  042504  LINS5:  .ASCIZ  /HEADER FROM ERROR SECTOR =
9482  051010  020122  051106  046517
9483  051016  042440  051122  051117
9484  051024  051440  041505  047524
9485  051032  020122  020075     000
9486  051037     122  042520  030503  LINEP5: .ASCIZ  /RPEC1 = /
9487  051044  036440  000040
9488  051050  051040  042520  031103  LINE05: .ASCIZ  / RPEC2 = /
9489  051056  036440  000040
9490  051062  042523  052103  051117  LINB6:  .ASCIZ  /SECTOR IS ECC CORRECTABLE /
9491  051070  044440  020123  041505
9492  051076  020103  047503  051122
9493  051104  041505  040524  046102
9494  051112  020105     000
9495  051115     123  041505  047524  LINC6:  .ASCIZ  /SECTOR READ CORRECTLY /
9496  051122  020122  042522  042101
9497  051130  041440  051117  042522
9498  051136  052103  054514  000040
9499  051144  047503  051122  04.505  LING6:  .ASCIZ  /CORRECTED ON /
9500  051152  042524  020104  047117
9501  051160  00C040
9502  051162  051040  052105  044522  LINR6:  .ASCIZ  / RETRIES/
9503  051170  051505     000
9504  051173     125  041516  051117  LINU06: .ASCIZ  /UNCORRECTABLE AFTER /
9505  051200  042522  052103  041101
9506  051206  042514  040440  052106
9507  051214  051105  000040
9508  051220  020040  047524  040524  LIN7M:  .ASCIZ  /  TOTAL MISPOS ERR = /
9509  051226  020114  044515  050123
9510  051234  051517  042440  051122
9511  051242  036440  000040
9512  051246  051117  042504  051522  LIN7O:  .ASCIZ  /ORDERS:/
9513  051254  000072
9514  051256  052040  052117  046101  LIN7P:  .ASCIZ  / TOTAL SEEkS = /
9515  051264  051440  042505  051513
```

```
9516  051272  036440  000040
9517  051276  052040  052117  046101  LIN7S:  .ASCIZ  / TOTAL SKI,OCYL ERR = /
9518  051304  051440  044513  047454
9519  051312  054503  020114  051105
9520  051320  020122  020075     000
9521  051325     040  042440  051122  LIN7T:  .ASCIZ  /  ERRORS:/
9522  051332  051117  035123     000
9523  051337     040  053440  042122  LIN7X:  .ASCIZ  /  WRDS XFR:/
9524  051344  020123  043130  035122
9525  051352     000
9526  051353     040  053440  042122  LIN7R:  .ASCIZ  /  WRDS READ:/
9527  051360  020123  042522  042101
9528  051366  000072
9529  051370  044504  043106  051105  LIN8M:  .ASCIZ  /DIFFERENT ERROR DURING RETRY/
9530  051376  047105  020124  051105
9531  051404  047522  020122  052504
9532  051412  044522  043516  051040
9533  051420  052105  054522     000
9534  051425     104  052101  020101  LIN9B:  .ASCIZ  /DATA COMPARISON ERRORS/
9535  051432  047503  050115  051101
9536  051440  051511  047117  042440
9537  051446  051122  051117  000123
9538  051454  020040  020040  020040  LIN9H:  .ASCII  /          GOOD    BAD/<CR><LF>
9539  051462  020040  043440  047517
9540  051470  020104  020040  041040
9541  051476  042101  005015
9542  051502  047514  020103  020040          .ASCIZ  /LOC       DATA    DATA/<CR><LF>
9543  051510  020040  042040  052101
9544  051516  020101  020040  042040
9545  051524  052101  006501  000012
9546  051532  047514  020103  020040  LIN9I:  .ASCIZ  /LOC       DATA/<CR><LF>
9547  051540  020040  042040  052101
9548  051546  006501  000012
9549  051552  047524  040524  020114  LIN9E:  .ASCIZ  /TOTAL COMPARE ERRORS = /
9550  051560  047503  050115  051101
9551  051566  020105  051105  047522
9552  051574  051522  036440  000040
9553  051602  044124  020105  040504  LIN9G:  .ASCIZ  /THE DATA COMPARED OK/<CR><LF>
9554  051610  040524  041440  046517
9555  051616  040520  042522  020104
9556  051624  045517  005015     000
9557  051631     105  051122  051117  LIN10A: .ASCIZ  /ERROR BURST BEGINS AT WORD /
9558  051636  041040  051125  052123
9559  051644  041040  043505  047111
9560  051652  020123  052101  053440
9561  051660  051117  020104     000
9562  051665     040  047111  042040  LIN10B: .ASCIZ  / IN DATA FIELD OF ERROR SECTOR/<CR><LF>
9563  051672  052101  020101  044506
9564  051700  046105  020104  043117
9565  051706  042440  051122  051117
9566  051714  051440  041505  047524
9567  051722  006522  000012
9568  051726  051105  047522  020122  LIN10C: .ASCII  /ERROR WAS NOT IN THE DATA READ - /<CR><LF>
9569  051734  040527  020123  047516
9570  051742  020124  047111  052040
9571  051750  042510  042040  052101
```

```
9572  051756  020101  042522  042:01
9573  051764  026440  006440     012
9574  051771     105  041503  041440          .ASCIZ  /ECC CORRECTION CAN'T BE PERFORMED/
9575  051776  051117  042522  052103
9576  052004  047511  020116  040503
9577  052012  023516  020124  042502
9578  052020  050040  051105  047506
9579  052026  046522  042105     000
9580  052033     105  041503  041440  LIN10H: .ASCII  /ECC CORRECTION RESULTS/<CR><LF>
9581  052040  051117  042522  052103
9582  052046  047511  020116  042522
9583  052054  052523  052114  006523
9584  052062     012
9585  052063     101  042104  020122          .ASCIZ  /ADDR    BAD     CORRECTED /<CR><LF>
9586  052070  020040  041040  042101
9587  052076  020040  020040  041440
9588  052104  051117  042522  052103
9589  052112  042105  006440  000012
9590  052120  047503  052116  047105  LIN11H: .ASCIZ  /CONTENTS OF ERROR SECTOR (REPORTED ABOVE)/<CR><LF>
9591  052126  051524  047440  020106
9592  052134  051105  047522  020122
9593  052142  042523  052103  051117
9594  052150  024040  042522  047520
9595  052156  052122  042105  040440
9596  052164  047502  042526  006451
9597  052172  000012
9598  052174  042101  051104  020040          .ASCIZ  /ADDR    DATA/<CR><LF>
9599  052202  020040  040504  040524
9600  052210  005015     000
9601  052213     040     040          LIN4SP: .ASCII  /  /
9602  052215     040                  LINSP:  .ASCII  / /
9603  052216  000040                  LINSPO: .ASCIZ  / /
9604
9605                                  .SBTTL  TELETYPE MESSAGES
9606
9607  052220  051104  053111  000105  UNTMSG: .ASCIZ  /DRIVE/
9608  052226  047440  043106  044514  UNTOFF: .ASCIZ  / OFFLINE/
9609  052234  042516     000
9610  052237     040  047117  044514  UNTON:  .ASCIZ  / ONLINE/
9611  052244  042516     000
9612  052247     040  047516  020124  UNTNOT: .ASCIZ  / NOT BEING TESTED/
9613  052254  042502  047111  020107
9614  052262  042524  052123  042105
9615  052270     000
9616  052271     040  046101  042522  UNTASN: .ASCIZ  / ALREADY BEING TESTED/
9617  052276  042101  020131  042502
9618  052304  047111  020107  042524
9619  052312  052123  042105     000
9620  052317     040  047516  020124  NOTRP:  .ASCIZ  @ NOT AN RP04/5/6@
9621  052324  047101  051040  030120
9622  052332  027464  027465  000066
9623  052340  047040  052117  050040  NOTPRS: .ASCIZ  / NOT PRESENT/
9624  052346  042522  042523  052116
9625  052354     000
9626  052355     040  047516  020124  NOTAVL: .ASCIZ  / NOT AVAILABLE/
9627  052362  053101  044501  040514
```

```
 9628  052370  046102  000105
 9629  052374  052440  051516  043101  NOTSAF: .ASCIZ  / UNSAFE/
 9630  052402  000105
 9631  052404  047125  052111  051440  SYST^T: .ASCIZ  /UNIT STATUS:/<CR><LF><LF>
 9632  052412  040524  052524  035123
 9633  052420  005015  000012
 9634  052424  050122  032060     000  RP04B:  .ASCIZ  /RP04/
 9635  052431     122  030120  000065  RP05:   .ASCIZ  /RP05/
 9636  052436  050122  033060     000  RP06:   .ASCIZ  /RP06/
 9637  052443     104  044522  042526  STATHD: .ASCII  /DRIVE PERFORMANCE SUMMARY/<CR><LF>
 9638  052450  050040  051105  047506
 9639  052456  046522  047101  042503
 9640  052464  051440  046525  040515
 9641  052472  054522  005015
 9642  052476  051104  020126  040520          .ASCII  /DRV PASS ORDERS   SEEKS   WRDS XFER   WRDS READ /
 9643  052504  051523  047440  042122
 9644  052512  051105  020123  020040
 9645  052520  042523  045505  020123
 9646  052526  020040  05'127  051504
 9647  052534  054040  042506  020122
 9648  052542  020040  051127  051504
 9649  052550  051040  040505  020104
 9650  052556  047523  052106  044040          .ASCIZ  /SOFT HARD  SKI MISP OTHER/<CR><LF>
 9651  052564  051101  020104  051440
 9652  052572  044513  046440  051511
 9653  052600  020120  052117  042510
 9654  052606  006522  000012
 9655  052612  047504  042516  005015  PDONE:  .ASCIZ  /DONE/<CR><LF><LF>
 9656  052620  000012
 9657  052622  037407  040506  040524  DROPNG: .ASCIZ  <07>/?FATAL OR EXCESSIVE ERRORS/
 9658  052630  020114  051117  042440
 9659  052636  041530  051505  044523
 9660  052644  042526  042440  051122
 9661  052652  051117  000123
 9662  052656  047105  020104  043117  ENDPAS: .ASCIZ  /END OF PASS/
 9663  052664  050040  051501  000123
 9664  052672  005015  047105  020104  ENDTST: .ASCIZ  <CR><LF>/END OF TEST/
 9665  052700  043117  052040  051505
 9666  052706  000124
 9667  052710  005015  051104  053111  DEASSG: .ASCIZ  <CR><LF>/DRIVE DEASSIGNED/
 9668  052716  020105  042504  051501
 9669  052724  044523  047107  042105
 9670  052732     000
 9671  052733     015  025012  025052  DRNUM:  .ASCIZ  <CR><LF>/********** DRIVE #/
 9672  052740  025052  025052  025052
 9673  052746  020052  051104  053111
 9674  052754  020105  000043
 9675  052760  051440  040524  052122  ASGND:  .ASCIZ  / STARTED/<CR><LF>
 9676  052766  042105  005015     000  NEDCLK: .ASCIZ  <CR><LF>/? 'L' OR 'P' CLOCK REQUIRED ON SYSTEM/<CR><LF>
 9677  052773     015  037412  023440
 9678  053000  023514  047440  020122
 9679  053006  050047  020047  046103
 9680  053014  041517  020113  042522
 9681  053022  052521  051111  042105
 9682  053030  047440  020116  054523
 9683  053036  052123  046505  005015
```

```
 9684  053044     000
 9685  053045     056     000          PERIOD: .ASCIZ  /./
 9686  053047     077     000          QUES:   .ASCIZ  /?/
 9687  053051     111  053116  046101  INVLD:  .ASCIZ  /INVALID COMMAND/<CR><LF>
 9688  053056  042111  041440  046517
 9689  053064  040515  042116  005015
 9690  053072     000
 9691  053073     015  042412  052116  ENTCOM: .ASCIZ  <CR><LF>/ENTER COMMANDS: /<CR><LF>
 9692  053100  051105  041440  046517
 9693  053106  040515  042116  035123
 9694  053114  006440  000012
 9695  053120  047105  042524  020122  ENTDRV: .ASCIZ  /ENTER I.D. FOR DRV #/
 9696  053126  027111  027104  043040
 9697  053134  051117  042040  053122
 9698  053142  021440     000
 9699  053145     015  042412  052116  ENTLMT: .ASCIZ  <CR><LF>/ENTER ADDRESS LIMITS FOR DRV #/
 9700  053152  051105  040440  042104
 9701  053160  042522  051523  046040
 9702  053166  046511  052111  020123
 9703  053174  047506  020122  051104
 9704  053202  020126  000043
 9705  053206  047105  042524  020122  ENTADR: .ASCIZ  @ENTER BAD TRK/SEC ADRS FOR DRV #@
 9706  053214  040502  020104  051124
 9707  053222  027513  042523  020103
 9708  053230  042101  051522  043040
 9709  053236  051117  042040  053122
 9710  053244  021440     000
 9711  053247     072     000          COLON:  .ASCIZ  /:/
 9712  053251     015  042012  052101  DATEIS: .ASCIZ  <CR><LF>/DATE: /
 9713  053256  035105  000040
 9714  053262  005015  050117  051105  IDIS:   .ASCIZ  <CR><LF>/OPERATOR I.D.: /
 9715  053270  052101  051117  044440
 9716  053276  042056  035056  000040
 9717  053304  005015  042012  053122  HEDLIN: .ASCIZ  <CR><LF><LF>/DRV  DRV I.D./<CR><LF>
 9718  053312  020040  051104  020126
 9719  053320  027111  027104  005015
 9720  053326     000
 9721  053327     116  047117  006505  NONE:   .ASCIZ  /NONE/<CR><LF>
 9722  053334  000012
 9723  053336  020077  047111  040526  BADENT: .ASCIZ  /? INVALID ENTRY/<CR><LF>
 9724  053344  044514  020104  047105
 9725  053352  051124  006531  000012
 9726  053360  054523  052123  046505  BUSY:   .ASCIZ  /SYSTEM BUSY.../<CR><LF>
 9727  053366  041040  051525  027131
 9728  053374  027056  005015     000  INTDON: .ASCII  <CR><LF>/PROGRAM INITIALIZATION COMPLETE/
 9729  053401     015  050012  047522
 9730  053406  051107  046501  044440
 9731  053414  044516  044524  046101
 9732  053422  055111  052101  047511
 9733  053430  020116  047503  050115
 9734  053436  042514  042524
 9735  053442  005015  054524  042520          .ASCIZ  <CR><LF>/TYPE A 'CONTROL C' TO ENTER COMMANDS/<CR><LF><LF>
 9736  053450  040440  023440  047503
 9737  053456  052116  047522  020114
 9738  053464  023503  052040  020117
 9739  053472  047105  042524  020122
```

```
 9740  053500  047503  046515  047101
 9741  053506  051504  005015  000012
 9742
 9743                                  .EVEN
 9744
 9745                                  ;PARAMETER ENTRY TABLE
 9746
 9747  053514  053642  000000  001404  PARLST: .WORD   PAR1,0,MAXDL
 9748  053522  053650  177777  001410          .WORD   PAR2,-1,INTRVL
 9749  053530  054000  177777  001402          .WORD   PAR19,-1,PASCNT
 9750  053536  053657  177777  001414          .WORD   PAR3,-1,CMPLMT
 9751  053544  053747  000001  001420          .WORD   PAR11,1,WCSEL
 9752  053552  053755  000007  001422          .WORD   PAR14,7,RATIO
 9753  053560  053772  000001  001430          .WORD   PAR16,1,ENDET
 9754  053566  053740  000001  001416          .WORD   PAR10,1,FORMAT
 9755  053574  053763  000001  001424          .WORD   PAR15,1,AUTOCK
 9756  053602  054007  000001  001426          .WORD   PAR20,1,NOTPRT
 9757  053610  000000                          .WORD   0                ;TABLE TERMINATOR
 9758
 9759  053612  047105  042524  020122  ASKPAR: .ASCIZ  /ENTER PARAMETERS:  /
 9760  053620  040520  040522  042515
 9761  053626  042524  051522  020072
 9762  053634  000040
 9763  053636  027440  000040          SLASH:  .ASCIZ  @ / @
 9764
 9765  053642  040515  042130  000114  PAR1:   .ASCIZ  /MAXDL/
 9766  053650  047111  051124  046126  PAR2:   .ASCIZ  /INTRVL/
 9767  053656     000
 9768  053657     103  050115  046514  PAR3:   .ASCIZ  /CMPLMT/
 9769  053664  000124
 9770  053666  040515  041530  046131  PAR4:   .ASCIZ  /MAXCYL/
 9771  053674     000
 9772  053675     115  047111  054503  PAR5:   .ASCIZ  /MINCYL/
 9773  053702  000114
 9774  053704  040515  052130  045522  PAR6:   .ASCIZ  /MAXTRK/
 9775  053712     000
 9776  053713     115  047111  051124  PAR7:   .ASCIZ  /MINTRK/
 9777  053720  000113
 9778  053722  040515  051530  041505  PAR8:   .ASCIZ  /MAXSEC/
 9779  053730     000
 9780  053731     115  047111  042523  PAR9:   .ASCIZ  /MINSEC/
 9781  053736  000103
 9782  053740  047506  046522  052101  PAR10:  .ASCIZ  /FORMAT/
 9783  053746     000
 9784  053747     127  051503  046105  PAR11:  .ASCIZ  /WCSEL/
 9785  053754     000
 9786  053755     122  052101  047511  PAR14:  .ASCIZ  /RATIO/
 9787  053762     000
 9788  053763     101  052125  041517  PAR15:  .ASCIZ  /AUTOCK/
 9789  053770  000113
 9790  053772  047105  042504  000124  PAR16:  .ASCIZ  /ENDET/
 9791  054000  040520  041523  052116  PAR19:  .ASCIZ  /PASCNT/
 9792  054006     000
 9793  054007     116  052117  051120  PAR20:  .ASCIZ  /NOTPRT/
 9794  054014  000124
 9795
```

```
 9796                                           .EVEN
 9797
 9798                                           ;PARAMETER TABLE POINTERS FOR ADDRESS LIMITS
 9799
 9800  054016  054036                          TABLE:  .WORD   TABLE0          ;PARAMETER TABLE FOR DRIVE 0
 9801  054020  054104                                  .WORD   TABLE1          ;PARAMETER TABLE FOR DRIVE 1
 9802  054022  054152                                  .WORD   TABLE2          ;PARAMETER TABLE FOR DRIVE 2
 9803  054024  054220                                  .WORD   TABLE3          ;PARAMETER TABLE FOR DRIVE 3
 9804  054026  054266                                  .WORD   TABLE4          ;PARAMETER TABLE FOR DRIVE 4
 9805  054030  054334                                  .WORD   TABLE5          ;PARAMETER TABLE FOR DRIVE 5
 9806  054032  054402                                  .WORD   TABLE6          ;PARAMETER TABLE FOR DRIVE 6
 9807  054034  054450                                  .WORD   TABLE7          ;PARAMETER TABLE FOR DRIVE 7
 9808
 9809                                           ;PARAMETER TABLE FOR ADDRESS LIMITS
 9810
 9811  054036  053675  000000  042040  TABLE0: .WORD   PAR5,0,MINCYL+DRIVE0
 9812  054044  053666  000000  042036          .WORD   PAR4,0,MAXCYL+DRIVE0
 9813  054052  053713  000022  042044          .WORD   PAR7,18.,MINTRK+DRIVE0
 9814  054060  053704  000022  042042          .WORD   PAR6,18.,MAXTRK+DRIVE0
 9815  054066  053731  000025  042050          .WORD   PAR9,21.,MINSEC+DRIVE0
 9816  054074  053722  000025  042046          .WORD   PAR8,21.,MAXSEC+DRIVE0,0
 9817  054102  000000
 9818
 9819  054104  053675  000000  042344  TABLE1: .WORD   PAR5,0,MINCYL+DRIVE1
 9820  054112  053666  000000  042342          .WORD   PAR4,0,MAXCYL+DRIVE1
 9821  054120  053713  000022  042350          .WORD   PAR7,18.,MINTRK+DRIVE1
 9822  054126  053704  000022  042346          .WORD   PAR6,18.,MAXTRK+DRIVE1
 9823  054134  053731  000025  042354          .WORD   PAR9,21.,MINSEC+DRIVE1
 9824  054142  053722  000025  042352          .WORD   PAR8,21.,MAXSEC+DRIVE1,0
 9825  054150  000000
 9826
 9827  054152  053675  000000  042650  TABLE2: .WORD   PAR5,0,MINCYL+DRIVE2
 9828  054160  053666  000000  042646          .WORD   PAR4,0,MAXCYL+DRIVE2
 9829  054166  053713  000022  042654          .WORD   PAR7,18.,MINTRK+DRIVE2
 9830  054174  053704  000022  042652          .WORD   PAR6,18.,MAXTRK+DRIVE2
 9831  054202  053731  000025  042660          .WORD   PAR9,21.,MINSEC+DRIVE2
 9832  054210  053722  000025  042656          .WORD   PAR8,21.,MAXSEC+DRIVE2,0
 9833  054216  000000
 9834
 9835  054220  053675  000000  043154  TABLE3: .WORD   PAR5,0,MINCYL+DRIVE3
 9836  054226  053666  000000  043152          .WORD   PAR4,0,MAXCYL+DRIVE3
 9837  054234  053713  000022  043160          .WORD   PAR7,18.,MINTRK+DRIVE3
 9838  054242  053704  000022  043156          .WORD   PAR6,18.,MAXTRK+DRIVE3
 9839  054250  053731  000025  043164          .WORD   PAR9,21.,MINSEC+DRIVE3
 9840  054256  053722  000025  043162          .WORD   PAR8,21.,MAXSEC+DRIVE3,0
 9841  054264  000000
 9842
 9843  054266  053675  000000  043460  TABLE4: .WORD   PAR5,0,MINCYL+DRIVE4
 9844  054274  053666  000000  043456          .WORD   PAR4,0,MAXCYL+DRIVE4
 9845  054302  053713  000022  043464          .WORD   PAR7,18.,MINTRK+DRIVE4
 9846  054310  053704  000022  043462          .WORD   PAR6,18.,MAXTRK+DRIVE4
 9847  054316  053731  000025  043470          .WORD   PAR9,21.,MINSEC+DRIVE4
 9848  054324  053722  000025  043466          .WORD   PAR8,21.,MAXSEC+DRIVE4,0
 9849  054332  000000
 9850
 9851  054334  053675  000000  043764  TABLE5: .WORD   PAR5,0,MINCYL+DRIVE5
```

```
9852  054342  053666  000000  043762          .WORD   PAR4,0,MAXCYL+DRIVE5
9853  054350  053713  000022  043770          .WORD   PAR7,18.,MINTRK+DRIVE5
9854  054356  053704  000022  043766          .WORD   PAR6,18.,MAXTRK+DRIVE5
9855  054364  053731  000025  043774          .WORD   PAR9,21.,MINSEC+DRIVE5
9856  054372  053722  000025  043772          .WORD   PAR8,21.,MAXSEC+DRIVE5,0
9857  054400  000000
9858
9859  054402  053675  000000  044270  TABLE6: .WORD   PAR5,0,MINCYL+DRIVE6
9860  054410  053666  000000  044266          .WORD   PAR4,0,MAXCYL+DRIVE6
9861  054416  053713  000022  044274          .WORD   PAR7,18.,MINTRK+DRIVE6
9862  054424  053704  000022  044272          .WORD   PAR6,18.,MAXTRK+DRIVE6
9863  054432  053731  000025  044300          .WORD   PAR9,21.,MINSEC+DRIVE6
9864  054440  053722  000025  044276          .WORD   PAR8,21.,MAXSEC+DRIVE6,0
9865  054446  000000
9866
9867  054450  053675  000000  044574  TABLE7: .WORD   PAR5,0,MINCYL+DRIVE7
9868  054456  053666  000000  044572          .WORD   PAR4,0,MAXCYL+DRIVE7
9869  054464  053713  000022  044600          .WORD   PAR7,18.,MINTRK+DRIVE7
9870  054472  053704  000022  044576          .WORD   PAR6,18.,MAXTRK+DRIVE7
9871  054500  053731  000025  044604          .WORD   PAR9,21.,MINSEC+DRIVE7
9872  054506  053722  000025  044602          .WORD   PAR8,21.,MAXSEC+DRIVE7,0
9873  054514  000000
9874
9875
9876  054516  000000  000000  000000  CYLDER: .WORD   0,0,0,0         ;HEADER BUFFER FOR 'READHD' ROUTINE
9877  054524  000000
9878
9879          054526                  ENDPGM  =       .                       ;LAST LOCATION OF PROG + 2
9880
9881                                  ;;*****************************************************************
9882
9883                                  ;ROUTINE TO GET THE DATE AND THE OPERATOR FROM THE OPERATOR
9884                                  ;CALL:
9885                                  ;       JSR     PC,OPRDAT
9886                                  ;       RETURN
9887                                  ;
9888                                  ;NOTE: THIS ROUTINE IS ENTERED ONLY AT INITIAL START
9889
9890  054526  104401  054640          OPRDAT: TYPE    ,ENTDAT         ;'ENTER DATE'
9891  054532  104411                          RDLIN                   ;READ THE ENTRY
9892  054534  012605                          MOV     (SP)+,R5        ;PUT THE ENTRY ADDRESS INTO R5
9893  054536  112537  001220                  MOVB    (R5)+,DATE      ;STORE THE DATE
9894  054542  112537  001221                  MOVB    (R5)+,DATE+1    ;STORE THE DATE
9895  054546  112537  001222                  MOVB    (R5)+,DATE+2    ;STORE THE DATE
9896  054552  112537  001223                  MOVB    (R5)+,DATE+3    ;STORE THE DATE
9897  054556  112537  001224                  MOVB    (R5)+,DATE+4    ;STORE THE DATE
9898  054562  112537  001225                  MOVB    (R5)+,DATE+5    ;STORE THE DATE
9899  054566  112537  001226                  MOVB    (R5)+,DATE+6    ;STORE THE DATE
9900  054572  112537  001227                  MOVB    (R5)+,DATE+7    ;STORE THE DATE
9901  054576  104401  054657                  TYPE    ,ENTID          ;'ENTER OPERATOR I.D.'
9902  054602  104411                          RDLIN                   ;READ THE ENTRY
9903  054604  012605                          MOV     (SP)+,R5        ;ENTRY ADDRESS
9904  054606  112537  001232                  MOVB    (R5)+,OPERID    ;STORE THE I.D.
9905  054612  112537  001233                  MOVB    (R5)+,OPERID+1  ;STORE THE I.D.
9906  054616  112537  001234                  MOVB    (R5)+,OPERID+2  ;STORE THE I.D.
9907  054622  112537  001235                  MOVB    (R5)+,OPERID+3  ;STORE THE I.D.
```

```
 9908  054626  112537  001236              MOVB    (R5)+,OPERID+4  ;STORE THE I.D.
 9909  054632  112537  001237              MOVB    (R5)+,OPERID+5  ;STORE THE I.D.
 9910  054636  000207                      RTS     PC              ;RETURN
 9911
 9912  054640  005015  047105  042524  ENTDAT: .ASCIZ  <CR><LF>/ENTER DATE: /
 9913  054646  020122  040504  042524
 9914  054654  020072     000
 9915  054657     105  052116  051105  ENTID:  .ASCIZ  /ENTER OPERATOR I.D.: /
 9916  054664  047440  042520  040522
 9917  054672  047524  020122  027111
 9918  054700  027104  020072     000
 9919          054706                  .EVEN
 9920
 9921                                  .SBTTL  ROUTINE TO SIZE MEMORY
 9922
 9923                                  ;;******************************************************************
 9924                                  ;*CALL:
 9925                                  ;*      JSR     PC,$SIZE
 9926                                  ;*      RETURN
 9927                                  ;*$LSTAD WILL CONTAIN THE LAST AVAILABLE MEMORY LOCATION
 9928
 9929  054706  010046          $SIZE:  MOV     R0,-(SP)        ;;SAVE R0 ON THE STACK
 9930  054710  010146                  MOV     R1,-(SP)        ;;SAVE R1 ON THE STACK
 9931  054712  013746  000004          MOV     @#ERRVEC,-(SP)  ;;SAVE PRESENT ERROR VECTOR PS & PC
 9932  054716  013746  000006          MOV     @#ERRVEC+2,-(SP)
 9933  054722  010600                  MOV     SP,R0           ;;SAVE THE STACK POINTER
 9934                                  ;;SET THE ERRVEC PS TO THE PRESENT PS
 9935  054724  104400                  TRAP                    ;;PUSH OLD PSW AND PC ON STACK
 9936  054726  012637  000006          MOV     (SP)+,@#ERRVEC+2        ;;SAVE THE PSW IN @#ERRVEC+2
 9937  054732  012737  054752  000004  MOV     #2$,@#ERRVEC    ;;SET FOR TIMEOUT
 9938  054740  012701  020000          MOV     #20000,R1       ;;FIRST ADDRESS
 9939  054744  005711          1$:     TST     (R1)            ;;TEST THIS ADDRESS
 9940  054746  005721                  TST     (R1)+           ;;STEP TO NEXT ADDRESS
 9941  054750  000775                  BR      1$              ;;TRY ANOTHER
 9942  054752  162701  000002  2$:     SUB     #2,R1           ;;DROP BACK
 9943  054756  010006                  MOV     R0,SP           ;;RESTORE THE STACK
 9944  054760  012637  000006          MOV     (SP)+,@#ERRVEC+2 ;;RESTORE ERROR VECTOR
 9945  054764  012637  000004          MOV     (SP)+,@#ERRVEC
 9946  054770  010137  055002          MOV     R1,$LSTAD       ;;LAST ADDRESS
 9947  054774  012601                  MOV     (SP)+,R1        ;;RESTORE R1
 9948  054776  012600                  MOV     (SP)+,R0        ;;RESTORE R0
 9949  055000  000207                  RTS     PC
 9950  055002  000000          $LSTAD: .WORD   0               ;;CONTAINS THE LAST ADDRESS
 9951
 9952                                  .SBTTL  BUSADR - GET BUS ADDRESS AND VECTOR ADDRESS FOR RH11
 9953                                  ;THIS ROUTINE IS USED TO INSURE THE BUS ADDRESS
 9954                                  ;OF THE RH11 IS SETUP FOR THE PROPER ADDRESS.
 9955                                  ;IT WILL ALSO READ THE ADDRESS FROM THE TTY IF
 9956                                  ;REQUIRED.
 9957                                  ;NOTE: THIS ROUTINE DESTROYS R0-R4
 9958                                  ;CALL
 9959                                  ;
 9960                                  ;       JSR     PC,BUSADR
 9961                                  ;       RETURN
 9962
 9963  055004  005737  001260  BUSADR: TST     CHGADR          ;INPUT FROM TTY REQUESTED?
```

```
 9964  055010  001450                        BEQ     7$              ;NO--BRANCH
 9965  055012  005037  001260                CLR     CHGADR          ;YES--CLEAR THE REQUEST FLAG
 9966  055016  012700  001170        1$:     MOV     #$RPADR,R0      ;FIRST ADDRESS
 9967  055022  104401  055214                TYPE    ,MRPCS1         ;"RPCS1="
 9968  055026  012046                        MOV     (R0)+,-(SP)     ;PRESENT RPCS1 ADDRESS
 9969  055030  104402                        TYPOC                   ;TYPE IT
 9970  055032  104401  052215                TYPE    ,LINSP          ;2 SPACES
 9971  055036  104411                        RDLIN                   ;GET THE ENTRY
 9972  055040  012601                        MOV     (SP)+,R1        ;ADDRESS OF ASCII TEXT
 9973  055042  004537  055236                JSR     R5,CK.NUM       ;CHECK THE NUMBER
 9974  055046  055066                        3$                      ;CARRIAGE RETURN ONLY ENTERED
 9975  055050  055132                        7$                      ;PERIOD ONLY ENTERED
 9976  055052  055016                        1$                      ;ILLEGAL INPUT
 9977  055054  055062                        2$                      ;TERMINATED WITH A CARRIAGE RETURN
 9978  055056  055016                        1$                      ;TERMINATED WITH A ","
 9979  055060  055126                        4$                      ;TERMINATED WITH A "'"
 9980  055062  010260  177776        2$:     MOV     R2,-2(R0)       ;SAVE NEW RPCS1
 9981  055066  104401  055225        3$:     TYPE    ,MRHVEC         ;"RHVEC="
 9982  055072  012046                        MOV     (R0)+,-(SP)     ;PRESENT RH11 VECTOR ADDRESS ON THE STACK
 9983  055074  104402                        TYPOC                   ;TYPE IT
 9984  055076  104401  052215                TYPE    ,LINSP          ;2 SPACES
 9985  055102  104411                        RDLIN                   ;READ THE ENTRY
 9986  055104  012601                        MOV     (SP)+,R1        ;ASCII TEXT ADDRESS
 9987  055106  004537  055236                JSR     R5,CK.NUM       ;CHECK THE NUMBER
 9988  055112  055132                        7$                      ;CARRIAGE RETURN ONLY ENTERED
 9989  055114  055132                        7$                      ;PERIOD ONLY ENTERED
 9990  055116  055066                        3$                      ;ILLEGAL INPUT
 9991  055120  055126                        4$                      ;TERMINATED WITH A CARRIAGE RETURN
 9992  055122  055066                        3$                      ;TERMINATED WITH A ","
 9993  055124  055126                        4$                      ;TERMINATED WITH A "'"
 9994  055126  010260  177776        4$:     MOV     R2,-2(R0)       ;SAVE INPUT
 9995  055132  013701  000004        7$:     MOV     @#ERRVEC,R1     ;SAVE THE ERROR VECTOR
 9996  055136  012737  055172  000004        MOV     #8$,@#ERRVEC    ;SETUP FOR TRAP
 9997  055144  005777  124020                TST     @$RPADR         ;CHECK FOR RH11
 9998  055150  010137  000004                MOV     R1,@#ERRVEC     ;RESTORE ERROR VECTOR
 9999  055154  012700  001170                MOV     #$RPADR,R0      ;FIRST ADDRESS OF NEW PARAMETERS
10000  055160  012701  033372                MOV     #RPADR,R1       ;FIRST ADDRESS OF WHERE TO PUT THEM
10001  055164  012021                        MOV     (R0)+,(R1)+     ;BUS ADDRESS
10002  055166  012021                        MOV     (R0)+,(R1)+     ;VECTOR ADDRESS
10003  055170  000207                        RTS     PC              ;RETURN
10004  055172  010137  000004        8$:     MOV     R1,@#ERRVEC     ;RESTORE ERROR VECTOR
10005  055176  022626                        CMP     (SP)+,(SP)+     ;CLEAN OFF THE STACK
10006  055200  104006                        ERROR   6               ;REPORT THE ERROR
10007  055202  005737  000042                TST     @#42            ;IS THERE A MONITOR?
10008  055206  001703                        BEQ     1$              ;NO--GO ASK FOR ADDRESS
10009  055210  000137  005404                JMP     $GET42          ;GO TO END OF PROGRAM
10010
10011  055214  050122  051503  020061  MRPCS1: .ASCIZ  @RPCS1 = @
10012  055222  020075     000
10013  055225     122  053110  041505  MRHVEC: .ASCIZ  @RHVEC = @
10014  055232  036440  000040
10015
10016                                  .SBTTL  CK.NUM - CHECK NUMBER (OCTAL)
10017                                  ;THIS ROUTINE CHECKS AN ASCIZ STRING FOR LEGAL CHARACTERS
10018                                  ;AND FORMS AN OCTAL NUMBER IN R2
10019                                  ;CALL:
```

```
10020                                   ;       MOV     #ADR,R1         ;ADDRESS OF ASCIZ STRING
10021                                   ;       MOV     #NUM,R2         ;MAX SIZE OF INPUT NUMBER
10022                                   ;       JSR     R5,CK.NUM       ;GO FORM THE NUMBER
10023                                   ;       RETURN  ADR1            ;"CR" ONLY ENTERED -- R2 = 0
10024                                   ;       RETURN  ADR2            ;"PERIOD" ONLY ENTERED -- R2 = 0
10025                                   ;       RETURN  ADR3            ;ILLEGAL CHARACTER IN THE INPUT STRING
10026                                   ;       RETURN  ADR4            ;"CR" ENTERED -- R2 = NUMBER
10027                                   ;       RETURN  ADR5            ;"COMMA" -- R2 = NUMBER
10028                                   ;       RETURN  ADR6            ;"PERIOD" -- R2 = NUMBER
10029
10030  055236  010446                  CK.NUM: MOV     R4,-(SP)        ;SAVE R4
10031  055240  010346                          MOV     R3,-(SP)        ;SAVE R3
10032  055242  010246                          MOV     R2,-(SP)        ;SAVE R2
10033  055244  005004                          CLR     R4              ;RETURN POINTER
10034  055246  005003                          CLR     R3              ;START NUMBER AT ZERO
10035  055250  005002                          CLR     R2              ;STORE RESULT
10036  055252  004537  027620                  JSR     R5,CK.CHR       ;CHECK ONE CHARACTER
10037  055256  055354                          6$                      ;ILLEGAL CHARACTER
10038  055260  055360                          8$                      ;CARRIAGE RETURN
10039  055262  055354                          6$                      ;" "
10040  055264  055356                          7$                      ;"'"
10041  055266  055272                          1$                      ;DIGIT 0-7
10042  055270  055354                          6$                      ;DIGIT 8-9
10043  055272  062705  000004          1$:     ADD     #4,R5           ;INCREMENT RETURN PAST "CR" AND "PERIOD" ONLY RETURNS
10044  055276  006303                  2$:     ASL     R3              ;FOR THE OCTAL NUMBER IN R3
10045  055300  103425                          BCS     6$              ;DON'T LET IT GET TO BIG
10046  055302  006303                          ASL     R3
10047  055304  103423                          BCS     6$
10048  055306  006303                          ASL     R3
10049  055310  103421                          BCS     6$
10050  055312  060203                          ADD     R2,R3
10051  055314  004537  027620                  JSR     R5,CK.CHR       ;CHECK ONE CHARACTER
10052  055320  055360                          8$                      ;ILLEGAL CHARACTER
10053  055322  055344                          5$                      ;CARRIAGE RETURN
10054  055324  055342                          4$                      ;" "
10055  055326  055334                          3$                      ;"'"
10056  055330  055276                          2$                      ;DIGIT 0-7
10057  055332  055360                          8$                      ;DIGIT 8-9
10058  055334  105711                  3$:     TSTB    (R1)            ;DOES A "CR" FOLLOW THE "PERIOD"
10059  055336  001010                          BNE     8$              ;BR IF NOT
10060  055340  005724                          TST     (R4)+           ;INCREMENT THE RETURN
10061  055342  005724                  4$:     TST     (R4)+           ;INCREMENT THE RETURN INDEX
10062  055344  005724                  5$:     TST     (R4)+           ;INCREMENT THE RETURN INDEX
10063  055346  020316                          CMP     R3,(SP)         ;INPUT VALUE TOO LARGE ?
10064  055350  101003                          BHI     8$              ;BR IF IT IS
10065  055352  000401                          BR      7$              ;BR IF NOT
10066  055354  005725                  6$:     TST     (R5)+           ;INCREMENT THE RETURN ADDRESS
10067  055356  005725                  7$:     TST     (R5)+           ;INCREMENT THE RETURN ADDRESS
10068  055360  060405                  8$:     ADD     R4,R5           ;SETUP FOR PROPER RETURN
10069  055362  010302                          MOV     R3,R2           ;LOAD ENTERED VALUE
10070  055364  005726                          TST     (SP)+           ;CLEAN OFF THE STACK
10071  055366  012603                          MOV     (SP)+,R3        ;RESTORE R3
10072  055370  012604                          MOV     (SP)+,R4        ;RESTORE R4
10073  055372  011505                          MOV     (R5),R5         ;GET RETURN ADDRESS
10074  055374  000205                          RTS     R5              ;RETURN
10075
```

```
10076
10077  055376  005015  040515  047111  TITLE:  .ASCII  <CR><LF>/MAINDEC-11-DZRJD-B/<CR><LF>
10078  055404  042504  026503  030461
10079  055412  042055  051132  042112
10080  055420  041055  005015
10081  055424  050122  032060  032457          .ASCIZ  /RP04/5/6 MULTI-DRIVE EXERCISER PROGRAM/<CR><LF><LF>
10082  055432  033057  046440  046125
10083  055440  044524  042055  044522
10084  055446  042526  042440  042530
10085  055454  041522  051511  051105
10086  055462  050040  047522  051107
10087  055470  046501  005015  000012
10088  055476  005015  047524  052040  LOADRV: .ASCII  <CR><LF>/TO TEST DRIVE 0, REPLACE THE 'XXDP' PACK ON DRIVE 0/<CR><LF>
10089  055504  051505  020124  051104
10090  055512  053111  020105  026060
10091  055520  051040  050105  040514
10092  055526  042503  052040  042510
10093  055534  023440  054130  050104
10094  055542  020047  040520  045503
10095  055550  047440  020116  051104
10096  055556  053111  020105  006460
10097  055564     012
10098  055565     127  052111  020110          .ASCII  /WITH ANOTHER PACK, CLEAR MEMORY LOCATION 40,/<CR><LF>
10099  055572  047101  052117  042510
10100  055600  020122  040520  045503
10101  055606  020054  046103  040505
10102  055614  020122  042515  047515
10103  055622  054522  046040  041517
10104  055630  052101  047511  020116
10105  055636  030064  006454     012
10106  055643     101  042116  051040          .ASCIZ  /AND RESTART THE PROGRAM/<CR><LF>
10107  055650  051505  040524  052122
10108  055656  052040  042510  050040
10109  055664  047522  051107  046501
10110  055672  005015     000
10111  055675     015  051412  051531  NOLOAD: .ASCIZ  <CR><LF>/SYSTEM HAS 16K MEMORY, 'XXDP' LOADER WILL BE OVERWRITTEN/<CR><L
10112  055702  042524  020115  040510
10113  055710  020123  033061  020113
10114  055716  042515  047515  054522
10115  055724  020054  054047  042130
10116  055732  023520  046040  040517
10117  055740  042504  020122  044527
10118  055746  046114  041040  020105
10119  055754  053117  051105  051127
10120  055762  052111  042524  006516
10121  055770  000012
10122
10123          000001                          .END
```

```
ABNRML  026564
ABS   = 000200
ACK   = 000123
ACL   = 000040
ACTDRV  033310
ACTSTR  033311
ACU   = 100000
ADE   = 001000
ASGN0   052760
ASGN1   024254
ASGN2   024334
ASGN3   024406
ASGN4   024524
ASGN5   024534
ASGN6   024542
ASGN7   024552
ASKPAR  053612
ASNERR  026456
ASNLST  001462
ASNMSG  026476
ASSIGN  024246
ATA   = 100000
ATABIT  033360
ATTN    001244
AT0   = 000001
AT1   = 000002
AT2   = 000004
AT3   = 000010
AT4   = 000020
AT5   = 000040
AT6   = 000100
AT7   = 000200
AUTOCK  001424
AVAIL   001530
A16   = 000400
A17   = 001000
BADENT  053336
BADSEC  001264
BAI   = 000010
BEGCOD  001434
BEGPAT  001432
BEGSIZ  001436
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BLKADR  001740
BPTVEC= 000014
BUFTBL  001616
BUSADR  055004
BUSY    053360
CFLAG   001262
CHGADR  001260
CI1     034732
CI3     035040
CI4     035146
CI5     035524
CI6     035546
CI7     035562
CI7B    035610
CI8     035662
CKBUS   013130
CKCLK   022452
CKCLK1  022546
CKCLK2  022614
CKCLK3  022644
CKERR   013030
CKFMT   011132
CKHCE   011326
CKSWR = 104407
CK.CHR  027620
CK.DEC  027572
CK.DIG  027672
CK.NUM  055236
CK.OCT  027544
CLKFLG  001210
CLOCK   023606
CLR   = 000040
CLRDPB  025224
CLRQUE  041472
CMCNT   001312
CMCYL   001314
CMDAT   013476
CMHED   013402
CMPAR   013214
CMPARQ  013232
CMPLMT  001414
CMPRES  017570
CMPRT   013742
CMPRX   013734
CMSEC   001316
CMSTR   013324
CMTRK   001317
COLON   053247
COMTBL  001760
CR    = 000015
CRLF  = 000200
CSF   = 000002
CSU   = 000010
CYLDER  054516
CYLIMT  001350
DATAPK  025200
DATAO   003026
DATAI   003066
DATE    001220
DATEIS  053251
DCK   = 100000
DCKER   007630
DCKER1  007762
DCL   = 000100
DCU   = 000001
DDISP = 177570
DEASGN  024622
DEASSG  052710
DE1   = 000040
DFF20 = 000002
DH1     047520
DH14    047676
DH15    050002
DH16    050101
DH2     047525
DH3     047602
DH4     047630
DH6     047667
DIGB  = 000004
DISPLA  001142
DISPLY= 104414
DISPRE  000174
DLT   = 100000
DL64  = 000020
DMD   = 000001
DONE    007324
DPINT   033264
DPR   = 000400
DPRQS   033274
DRIVE   001242
DRIVE0  041730
DRIVE1  042234
DRIVE2  042540
DRIVE3  043044
DRIVE4  043350
DRIVE5  043654
DRIVE6  044160
DRIVE7  044464
DRNUM   052733
DROP    026506
DROPNG  052622
DRQ   = 004000
DRVACT  033234
DRVCLR= 000111
DRVER   011100
DRVINT  033622
DRVPRM  025420
DRVQUE  041570
DRVSTA  033244
DRVTYP  033254
DRY   = 000200
DSWR  = 177570
DTE   = 010000
DTEER   011736
DTSY  = 000200
DTUM    033356
DT00  = 000001
DT01  = 000002
DT02  = 000004
DT03  = 000010
DT04  = 000020
DT05  = 000040
DT06  = 000100
DT07  = 000200
DT08  = 000400
DT1     050152
DT14    050222
DT15    050244
DT16    050266
DT2     050156
DT3     050174
DT4     050204
DT6     050216
DUNIT   001464
DVA   = 004000
ECBAD0  001334
ECBAD1  001342
ECBIT   001320
ECC     014342
ECCX    015074
ECC1    014740
ECC2    015070
ECGD    001332
ECGD1   001340
ECH   = 000130
ECI   = 004000
ECMSK0  001324
ECMSK1  001326
ECSEC   001322
ECWRD0  001330
ECWRD1  001336
EMPTYQ  041550
EMTVEC= 000030
EM1     045060
EM10    045352
EM11    045415
EM12    045450
EM13    045501
EM14    045553
EM15    045576
EM2     045123
EM20    045632
EM21    045653
EM22    045704
EM23    045757
EM24    046036
EM25    046104
EM26    046165
EM27    046212
EM3     045161
EM30    046247
EM31    046301
EM32    046344
EM33    046377
EM34    046454
EM35    046516
EM36    046554
EM37    046605
EM4     045217
EM40    046667
EM41    046725
EM42    046771
EM43    047052
EM44    047117
EM45    047203
EM46    047271
EM5     045254
EM50    047343
EM51    047432
EM6     045310
EM60    047475
ENDCMP  014216
ENDCN   001352
ENDCON  001372
ENDET   001430
ENDPAS  052656
ENDPGM= 054526
```

```
ENDSEK  001376
ENDSK   001356
ENDTST  052672
ENTADR  053206
ENTCOM  053073
ENTDAT  054640
ENTDRV  053120
ENTID   054657
ENTLMT  053145
ENTPR   005032
EOP     026612
EOPX    027064
EOP1    026644
EOP2    026666
ERCTR   001306
ERPRC1  006704
ERPROC  006670
ERR   = 040000
ERRVEC= 000004
EXT1  = 000001
EXT10 = 000010
EXT2  = 000002
EXT20 = 000020
EXT4  = 000004
EXT40 = 000040
FACTOR  015730
FAIRNS  001254
FALPAR  007070
FALPR1  007100
FEN   = 000200
FER   = 000020
FILBUF  016302
FMTER   012132
FMT22 = 010000
FORMAT  001416
FRSTER  001300
F1    = 000002
F2    = 000004
F3    = 000010
F4    = 000020
F5    = 000040
GENDPB  044770
GENREG  045010
GETADR  026012
GETBUF  015732
GETID   025702
GETPAR  017362
GETPAT  017334
GETREG= 000141
GETREM  027066
GETREQ  041644
GO    = 000001
GODRIV  016430
GRV   = 000010
GTSWR = 104406
HCE   = 000200
HCEER   012210
HCI   = 002000
HCRC  = 000400
HCRCER  010750
HEDLIN  053304
HOUR    001266
HT    = 000011
HZ      001212
IAE   = 002000
IAEER   012052
IDIS    053262
IDLE    006242
IDLE1   006340
IE    = 000100
ILF   = 000001
ILR   = 000002
INCHRD  023370
INCMIS  023440
INCSKI  023414
INCSOF  023344
INCTOT  023464
INTDON  053401
INTRVL  001410
INVLD   053051
IOTVEC= 000020
IR    = 000100
ISR     036254
IXE   = 004000
KSR     023750
KSR1    024004
LA      036116
LACNT   033322
LF    = 000012
LIMIT   001310
LINA3   050520
LINB3   050572
LINB5   050747
LINB6   051062
LINCA3  050551
LINC6   051115
LINDA3  050562
LINDEC  022334
LIND5   050732
LINEN3  050503
LINEO5  051050
LINEP5  051037
LINE1   020240
LINE2   020304
LINE2A  020454
LINE2B  020472
LINE3   020712
LINE3A  020720
LINE3B  020726
LINE3C  020740
LINE3D  020750
LINE3E  021016
LINE3F  021104
LINE4   021366
LINE5   021456
LINE5A  021556
LINE5B  021650
LINE6   021712
LINE6A  021724
LINE6B  021732
LINE6C  021740
LINE6D  021746
LINE7   022022
LINE7A  022150
LINE8   022270
LING6   051144
LINKDV  027114
LINM3   050403
LINM4   050646
LINN3   050421
LINOCT  022302
LINP3   050445
LINP5   050765
LINR6   051162
LINSP   052215
LINSPO  052216
LINSS3  050631
LINST3  050614
LINS3   050466
LINS4   050665
LINS5   051003
LINT3   050540
LINU06  051173
LINW3   050602
LINX4   050677
LIN10A  051631
LIN10B  051665
LIN10C  051726
LIN10H  052033
LIN11H  052120
LIN2C   050302
LIN2P   050323
LIN2S   050347
LIN3.1  021156
LIN3.3  021270
LIN3.4  021322
LIN4SP  052213
LIN6.1  021754
LIN6.2  021776
LIN7M   051220
LIN7O   051246
LIN7P   051256
LIN7R   051353
LIN7S   051276
LIN7T   051325
LIN7X   051337
LIN8M   051370
LIN9B   051425
LIN9E   051552
LIN9G   051602
LIN9H   051454
LIN9I   051532
LKPAR   005006
LOADRV  055476
LST   = 002000
LSTAD   001256
MAIN    005434
MAIN1   005562
MAIN2   005702
MAIN3   006136
MASK    001250
MATCH   014264
MAXCYL= 000106
MAXDL   001404
MAXER   001406
MAXSEC= 000116
MAXTRK= 000112
MCLK  = 000002
MCPE  = 020000
MCPEMX  033370
MHS   = 001000
MINCYL= 000110
MINSEC= 000120
MINTRK= 000114
MINUTE  001270
MINX  = 000004
MNDLTA  033404
MNTBL   002010
MOH   = 020000
MOL   = 010000
MONTR   005326
MPE   = 000400
MRD   = 000020
MRHVEC  055225
MRPCS1  055214
MSE   = 000020
MSTCK = 000010
MWR   = 000040
MXDLTA  033402
MXF   = 001000
MXLACT  033400
MXWNDW  033406
M.DPID  027162
M.DP40  027220
M.DP41  027254
M.DP42  027264
M.DP44  027316
M.DP50  027330
NBA   = 100000
NED   = 010000
NEDCLK  052773
NEM   = 004000
NEWASN  024612
NEWUNT  001506
NHS   = 002000
NOLOAD  055675
NOMTCH  012606
NONE    053327
NOTAVL  052355
NOTPRS  052340
NOTPRT  001426
NOTRP   052317
NOTSAF  052374
OCYL  = 100000
OFFCOD  002220
OFFSET= 000115
OFFST   015332
OFLIN   007176
OFMSGA  002724
OFMSG0  002274
OFMSG1  002327
OFMSG2  002363
OFMSG3  002417
OFMSG4  002453
OFMSG5  002507
OFMSG6  002543
OFMSG7  002577
OFMSG8  002633
OFMSG9  002667
OFMTBL  002240
OFREV = 000200
OF100 = 000004
OF200 = 000010
OF25  = 000001
OF400 = 000020
OF50  = 000002
OF800 = 000040
OPE   = 020000
OPERID  001232
OPI   = 020000
OPIER   011626
OPIER1  011672
OPRDAT  054526
OPT     034452
OPTBL   001766
```

```
OR     = 000200
ORDERQ  001440
PACK    001216
PAR    = 000010
PARENT  026332
PARER   011760
PARLST  053614
PARQ    001574
PAR1    053642
PAR10   053740
PAR11   053747
PAR14   053755
PAR15   053763
PAR16   053772
PAR19   054000
PAR2    053650
PAR20   054007
PAR3    053657
PAR4    053666
PAR5    053675
PAR6    053704
PAR7    053713
PAR8    053722
PAR9    053731
PASCNT  001402
PAT    = 000020
PCLOCK  001206
PDONE   052612
PERIOD  053045
PGE    = 002000
PGM    = 001000
PIP    = 020000
PIRQ   = 177772
PIRQVE= 000240
PLU    = 020000
POPQUE  041666
POSER   011552
PROCES  006600
PRTBAD  015102
PRTIM   007266
PR0    = 000000
PR1    = 000040
PR2    = 000100
PR3    = 000140
PR4    = 000200
PR5    = 000240
PR6    = 000300
PR7    = 000340
PS     = 177776
PSEL   = 002000
PSU    = 000001
PSW    = 177776
PUNSAF  006766
PWRVEC= 000024
QCNT    041200
QDRV0   041272
QDRV1   041312
QDRV2   041332
QDRV3   041352
QDRV4   041372
QDRV5   041412
QDRV6   041432
QDRV7   041452
QINPT   041210
QOUTPT  041220
QSTART  041230
QSTOP   041252
QTERM  = 041472
QUES    053047
QVCON   001362
QVSEK   001366
RANCYL  016742
RANPAT  017210
RANSEC  016566
RANSIZ  017054
RANTRK  016642
RANWRT  017230
RANXIT  017220
RATIO   001422
RAW    = 000020
RDCHR  = 104410
RDDAT  = 000171
RDHD   = 000173
RDLIN  = 104411
RDY    = 000200
RD.ADR  040472
RD.RP   040446
RD.RP1  040470
RD.RP2  040610
RD.RP3  040614
RD.RP4  040620
RD.WRD  040474
READDR  022354
READHD  015356
READIN= 000121
RECAL  = 000107
RECALT  015300
RECALD  015304
REDAPK  025212
REFMT   006434
REFMTX  006576
REGPAS  004164
RELBUF  016066
RELSE  = 000113
REPLZ   027340
RESREG= 104413
RESVEC= 000010
RETRY   001252
RMR    = 000004
RNOP   = 000101
RPADR   033372
RPAS   = 000016
RP6A   = 000004
RPCN   = 000034
RPLC   = 000036
RPOS1  = 000000
RPOS2  = 000010
RPOA   = 000006
RPOB   = 000022
RPOS1  = 000012
RPOT   = 000026
RPEC1  = 000044
RPEC2  = 000046
RPERRS  033224
RPER1  = 000014
RPER2  = 000040
RPER3  = 000042
RPINIT  033410
RPLA   = 000020
RPMR   = 000024
RPOF   = 000032
RPSN   = 000030
RPTMR   037720
RPVEC   033374
RPWC   = 000002
RP04    034160
RP04B   052424
RP05    052431
RP06    052436
RTC    = 000117
RTNCTR  015254
R6     =%000006
R7     =%000007
S       050442
SAVEFG  033332
SAVER1  001302
SAVER5  001304
SAVREG= 104412
SC      036602
SCMND   024730
SC1    = 000100
SC10   = 001000
SC11    037420
SC12    037510
SC13    037560
SC2    = 000200
SC20   = 002000
SC3     036646
SC4     036652
SC5     036664
SC6     037054
SC6A    037164
SC7     037312
SC8     037370
SDETAL  023006
SEARCH= 000131
SECLMT  001344
SECOND  001272
SEEK   = 000105
SEEKFG  033334
SELDRV= 000145
SELPAR  016506
SETFMT= 000143
SETUP   004214
SETVEC  005060
SET.IE  041126
SHDTYP  022764
SIXTEE  001274
SIZMEM  004636
SKI    = 040000
SKIER   012372
SLASH   053636
SPOTCK  020102
SRCHWT  033306
STACK  = 001100
START   004106
START1  004116
START2  004122
STATHD  052443
STATIN  001214
STATIS  015566
STATPR  022654
STKLMT= 177774
STNDAT  002762
ST0     040012
ST01    040042
ST02    040240
ST03    040310
ST05    040334
ST06    040342
ST07    040400
ST08    040430
ST09    040440
SVRH11  041010
SWR     001140
SWREG   000176
SWTIM   007140
SW0    = 000001
SW00   = 000001
SW01   = 000002
SW02   = 000004
SW03   = 000010
SW04   = 000020
SW05   = 000040
SW06   = 000100
SW07   = 000200
SW08   = 000400
SW09   = 001000
SW1    = 000002
SW10   = 002000
SW11   = 004000
SW12   = 010000
SW13   = 020000
SW14   = 040000
SW15   = 100000
SW2    = 000004
SW3    = 000010
SW4    = 000020
SW5    = 000040
SW6    = 000100
SW7    = 000200
SW8    = 000400
SW9    = 001000
SYSTAT  052404
T       050416
TABLE   054016
TABLE0  054036
TABLE1  054104
TABLE2  054152
TABLE3  054220
TABLE4  054266
TABLE5  054334
TABLE6  054402
TABLE7  054450
TAP    = 040000
TBITVE= 000014
TD      036320
TDF    = 000040
TIMER   033336
TITLE   055376
TKVEC  = 000060
TPVEC  = 000064
TRAPVE= 000034
TRE    = 040000
TRFER   012272
TRKLMT  001346
TRK1   = 004000
TRK10  = 040000
TRK2   = 010000
TRK20  = 100000
TRK4   = 020000
TRNSWT  033304
TRTVEC= 000014
TUF    = 000100
TYPDS  = 104405
```

```
TYPE  = 104401        WRTPK2  020016        $ERRPC  001116        $OPERC= 000036        $SEC   = 000010
TYPEST  022736        WRTPK3  020042        $ERRTB  004026        $PACK  = 000026        $SETUP= 000146
TYPOC = 104402        WRTPK4  020074        $ERRTY  030740        $PASS    001100        $SIZE    054706
TYPON = 104404        WRT.AD  040712        $ERTTL  001112        $PASSC= 000070        $SKI   = 000064
TYPOS = 104403        WRT.RP  040622        $FAIR = 000072        $PATTC= 000030        $SOFT  = 000060
TYPRI4  027470        WRT.R1  040706        $FILLC  001156        $POSIT= 000042        $SSEC  = 000022
UCPAR   007056        WRT.R2  040772        $FILLS  001155        $PREVA= 000032        $STUP  = 177777
ULDFLG  033312        WRT.R3  041000        $FIRST= 000122        $PREVO= 000027        $SUPRS  027430
UNIT    001246        WRT.R4  041004        $FMT   = 000001        $PSEL  = 000003        $SVPC  = 000210
UNLOAD= 000103        WRT.R5  041006        $GDADR  001120        $QUES    001164        $SWR   = 122000
UNS   = 040000        WRT.WD  040710        $GDDAT  001124        $RAND    032424        $TATUS= 000016
UNSAF   012526        WRU   = 000400        $GET42  005404        $RDCHR  032250        $TERM  = 000032
UNTASN  052271        WSU   = 000004        $GTSWR  032036        $RDLIN  030272        $TIME    023510
UNTMSG  052220        ZROIND  001276        $HARD = 000062        $RDSZ  = 000001        $TKB     001146
UNTNOT  052247        $AUTOB  001134        $HD    = 000000        $READ  = 000052        $TKINT  030110
UNTOFF  052226        $BDADR  001122        $HINUM  032522        $REG   = 000014        $TKS     001144
UNTON   052237        $BDDAT  001126        $ICNT   001104        $RESRE  032564        $TKSRV  030140
UPE   = 020000        $BDSEC= 000124        $INTAG  001135        $RETRY  015440        $TN    = 000001
US1   = 000001        $BELL   001160        $ITEMB  001114        $RPADR  001170        $TNPWR  032732
US2   = 000002        $BUF   = 000006        $LF     001166        $RPAS  = 000252        $TOTAL= 000056
US4   = 000004        $CHARC  031310        $LKCSB  001176        $RPBA  = 000240        $TPB     001152
UWR   = 000010        $CKSWR  031766        $LKCSR  001174        $RPCA  = 000270        $TPFLG  001157
VUF   = 000002        $CMTAG  001100        $LKS    001202        $RPCC  = 000272        $TPS     001150
VU30  = 010000        $CM3   = 000000        $LLVEC  001204        $RPCS1= 000234        $TRANS= 000046
VV    = 000100        $CNTLC  030562        $LONUM  032524        $RPCS2= 000244        $TRAP    033136
WAIT    001552        $CNTLG  032375        $LPADR  001106        $RPDA  = 000242        $TRAP2  033160
WAO   = 000002        $CNTLU  032370        $LPERR  001110        $RPDB  = 000256        $TRK   = 000011
WC      036474        $CODE  = 000024        $LPVEC  001200        $RPDS1= 000246        $TRP   = 000015
WCE   = 040000        $COMND= 000002        $LSTAD  055002        $RPDT  = 000262        $TRPAD  033172
WCF   = 000040        $CRLF   001165        $MISPO= 000066        $RPEC1= 000300        $TSTNM  001102
WCFER   012430        $CYL   = 000012        $MNEW   032413        $RPEC2= 000302        $TTYIN  030550
WCKD  = 000151        $DBLK   031756        $MSWR   032402        $RPER1= 000250        $TYPDS  031542
WCKER   010422        $DB2D   032622        $NCODE= 000074        $RPER2= 000274        $TYPE   031074
WCKHD = 000153        $DB2O   033016        $NCYL  = 000100        $RPER3= 000276        $TYPEC  031244
WCSEL   001420        $DECVL  033002        $NEXT  = 000104        $RPLA  = 000254        $TYPEX  031312
WCU   = 000001        $DOAGN  005430        $NPATC= 000075        $RPMR  = 000260        $TYPOC  031340
WLE   = 004000        $DRVID= 000224        $NSEC  = 000076        $RPOF  = 000266        $TYPON  031354
WLEER   012104        $DSPLY  027516        $NTRK  = 000077        $RPSN  = 000264        $TYPOS  031314
WRL   = 004000        $DTBL   031746        $NULL   001154        $RPVEC  001172        $WRDL  = 000020
WRTDAT= 000161        $ENDAD  005420        $NWRDL= 000102        $RPWC  = 000236        $WRDM  = 000004
WRTHD = 000163        $ERFLG  001103        $OCNT   031536        $SAVRE  032526        $OFILL  031537
WRTPK   017605        $ERMAX  001115        $OCTVL  033120        $SB2D   030030        .      = 055772
WRTPK1  017636        $ERROR  030570        $OMODE  031540        $SB2O   030060

. ABS.  055772     000

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DSKM:DZRJDB,DSKZ:DZRJDB/SOL/NL:MD:MC:CND=DSKZ:SYSMAC.SML,DSKM:RP0456.010,DSKM:DZRJDB.P11
 RUN-TIME: 21 31 1 SECONDS
 RUN-TIME RATIO: 486/54=9.0
 CORE USED:  50K  (100 PAGES)


EOF1DZRJDBSEQ              00010000      770720            PDP10 411
```