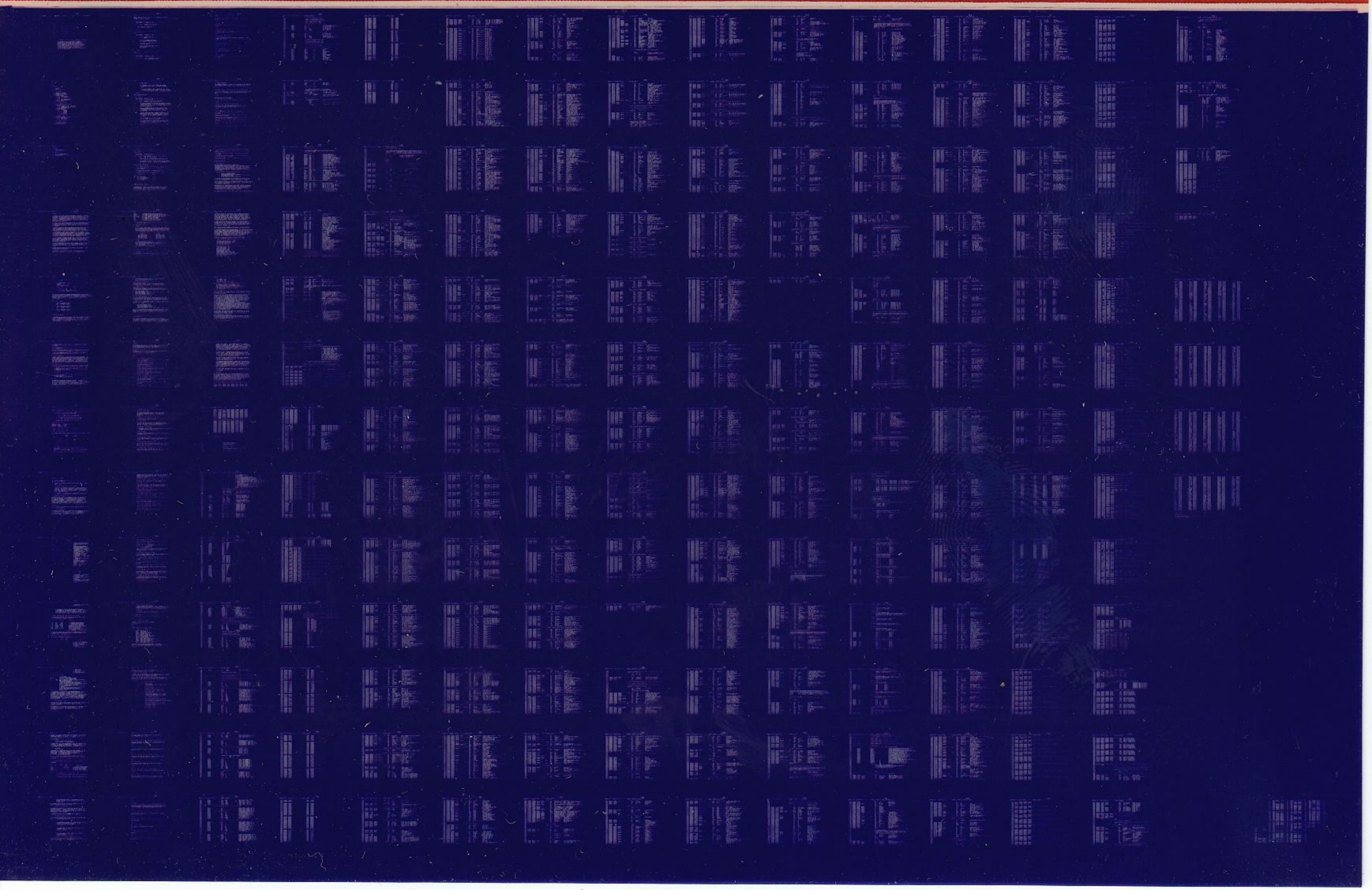
RP04/5/6

MULTIDRIVE LOGIC
CZRJDC0

AH-9197C-MC
COPYRIGHT © 74-77
FICHE 1 OF 1

JAN 1978
digital
MADE IN USA



.REM @

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

PRODUCT CODE: AC-9195C-MC

PRODUCT NAME: CZRJDC0 MLT-DR LGC

DATE CREATED: APRIL, 1977

MAINTAINER: DIAGNOSTIC ENGINEERING

AUTHOR: C. HESS

COPYRIGHT (C) 1974,1977 DIGITAL EQUIPMENT CORP., MAYNARD, MASS.

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE
WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A
COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.
DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY
FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.
THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED
UNDER A LICENSE AND MAY ONLY BE USED OR COPIED IN
ACCORDANCE WITH THE TERMS OF SUCH LICENSE.
DIGITAL EQIUPMENT CORPORATION ASSUMES NO RESPONSIBILITY
FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT
THAT IS NOT SUPPLIED BY DIGITAL.

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
CMPLMT	10	3	0 - 'MAXDL'	ERRORS PRINTED OUT IF SW<07>=0 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 'WCF' 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.
R0<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 HAS OCCURRED. THE OPERATION WILL BE RETRIED 3 TIMES IF THE ERROR WAS CAUSED BY 'DLT', 'UPE', '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.

B.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 MANTAIN 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

```
1770
1771                    .TITLE  CZRJDCO, RP04/5/6 MLT-DR LGC
1772                    ;*COPYRIGHT (C) 1975,1977
1773                    ;*DIGITAL EQUIPMENT CORP.
1774                    ;*MAYNARD, MASS. 01754
1775                    ;*
1776                    ;*PROGRAM BY C. HESS
1777                    ;*
1778                    ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
1779                    ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
1780                    ;*
1781                    .SBTTL  OPERATIONAL SWITCH SETTINGS
1782                    ;*
1783                    ;*      SWITCH                  USE
1784                    ;*      ------          --------------------
```

```
                        ;*          15          HALT ON ERROR
                        ;*          13          INHIBIT ERROR TYPEOUTS
                        ;*          10          BELL ON ERROR
                        ;*           7          DISPLAY ALL DATA COMPARE ERRORS
                        ;*           6          DON'T CHANGE PARAMETERS (LOOP ON PRESENT VALUES)
                        ;*           5          A. PARTIAL REGISTER DISPLAY IF ERROR
                        ;*                      B. NO ECC CORRECTION RESULTS DISPLAYED IF ERROR
                        ;*           4          A. DO NOT CHECK FOR MAXIMUM ERROR COUNTS
                        ;*                      B. DO NOT DROP DRIVE 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 STATUS 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
                        002000          SW10=   2000
                        001000          SW09=   1000
                        000400          SW08=   400
                        000200          SW07=   200
                        000100          SW06=   100
                        000040          SW05=   40
                        000020          SW04=   20
                        000010          SW03=   10
                        000004          SW02=   4
                        000002          SW01=   2
                        000001          SW00=   1
                                        .EQUIV  SW09,SW9
                                        .EQUIV  SW08,SW8
                                        .EQUIV  SW07,SW7
                                        .EQUIV  SW06,SW6
                                        .EQUIV  SW05,SW5
                                        .EQUIV  SW04,SW4
                                        .EQUIV  SW03,SW3
                                        .EQUIV  SW02,SW2
                                        .EQUIV  SW01,SW1
                                        .EQUIV  SW00,SW0

                                        ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
                100000                  BIT15=  100000
                040000                  BIT14=  40000
                020000                  BIT13=  20000
                010000                  BIT12=  10000
                004000                  BIT11=  4000
                002000                  BIT10=  2000
                001000                  BIT09=  1000
                000400                  BIT08=  400
                000200                  BIT07=  200
                000100                  BIT06=  100
                000040                  BIT05=  40
                000020                  BIT04=  20
                000010                  BIT03=  10
                000004                  BIT02=  4
                000002                  BIT01=  2
                000001                  BIT00=  1
                                        .EQUIV  BIT09,BIT9
                                        .EQUIV  BIT08,BIT8
                                        .EQUIV  BIT07,BIT7
                                        .EQUIV  BIT06,BIT6
                                        .EQUIV  BIT05,BIT5
                                        .EQUIV  BIT04,BIT4
                                        .EQUIV  BIT03,BIT3
                                        .EQUIV  BIT02,BIT2
```

```
                        .EQUIV  BIT01,BIT1
                        .EQUIV  BIT00,BIT0

                        ;*BASIC "CPU" TRAP VECTOR ADDRESSES
000004                  ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
000010                  RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
000014                  TBITVEC=14              ;;"T" BIT
000014                  TRTVEC= 14              ;;TRACE TRAP
000014                  BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
000020                  IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
000024                  PWRVEC= 24              ;;POWER FAIL
000030                  EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
000034                  TRAPVEC=34              ;;"TRAP" TRAP
000060                  TKVEC=  60              ;;TTY KEYBOARD VECTOR
000064                  TPVEC=  64              ;;TTY PRINTER VECTOR
000240                  PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR

                        ;;*****************************************************************

                        .SBTTL  RH11 REGISTERS

                        ;;*****************************************************************

                        ;CONTROL AND STATUS REGISTER 1 (RPCS1)

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

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

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

                        ;CONTROL AND STATUS REGISTER 2 (RPCS2)

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

```
                        WCE=    40000           ;WRITE CHECK ERROR (BIT #14)
                        DLT=    100000          ;DATA LATE (BIT #15)

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


                        ;;*****************************************************************

                        .SBTTL  RP04/5/6 REGISTERS

                        ;;*****************************************************************

                        ;CONTROL AND STATUS 1 REGISTER. (#00)

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)
```

```
                        DTE=    10000           ;DRIVE TIMING ERROR (BIT #12)
                        OPI=    20000           ;OPERATION INCOMPLETE (BIT #13)
                        UNS=    40000           ;DRIVE UNSAFE (BIT #14)
                        DCK=    100000          ;DATA CHECK ERROR (BIT 15)

                        ;MAINTAINABILITY REGISTER (RPMR)(#03)

                        DMD=    1               ;DIAGINOSTIC MODE (BIT #0)
                        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)
```

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

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

                                ;RP05/6 ERROR REGISTER #02 (RPER2) (#10)

000001                  WCU=    1               ;WRITE CURRENT UNSAFE (BIT #0)
000002                  CSF=    2               ;CURRENT SINK FAILURE (BIT #1)
000004                  WSU=    4               ;WRITE SELECT UNSAFE (BIT #2)
000010                  CSU=    10              ;CURRENT SWITCH UNSAFE (BIT #3)
000020                  RAW=    20              ;READ AND WRITE (BIT #4)
000040                  TDF=    40              ;TRANSITIONS DETECTOR FAILURE (BIT #5)
000100                  TUF=    100             ;TRANSITIONS UNSAFE (BIT #6)
000200                  ABS=    200             ;ABNORMAL STOP (BIT #7)
000400                  WRU=    400             ;WRITE READY UNSAFE (BIT #8)
001000                  MHS=    1000            ;MUTLTIPLE HEAD SELECT (BIT #9)
002000                  NHS=    2000            ;NO HEAD SELECTION (BIT #10)
004000                  IXE=    4000            ;INDEX ERROR (BIT #11)
020000                  PLU=    20000           ;PLO UNSAFE (BIT #12)

                                ;OFFSET REGISTER (RPOF) (#11)

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

                                ;DESIRED CYLINDER ADDRESS (RPCA) (#12)
```

```
                        ;(EACH BIT IS CALLED BY BIT NUMBER)

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

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

                        ;RP04 ERROR REGISTER #03 (RPER3) (#15)

000001                  PSU=    1               ;PACK SPEED UNSAFE (BIT #0)
000002                  VUF=    2               ;VELOCITY UNSAFE (BIT #1)
000010                  UWR=    10              ;ANY UNSAFE EXCEPT READ/WRITE (BIT #3)
000040                  ACL=    40              ;AC LOW (BIT #5)
000100                  DCL=    100             ;DC LOW (BIT #6)
040000                  SKI=    40000           ;SEEK INCOMPLETE (BIT #14)
100000                  OCYL=   100000          ;OFF CYLINDER (BIT #15)

                        ;RP05/6 ERROR REGISTER #03 (RPER3) (#15)

000001                  DCU=    1               ;DC UNSAFE (BIT #0)
000002                  WAO=    2               ;WRITE AND OFFSET (BIT #1)
000040                  ACL=    40              ;AC LOW (BIT #5)
000100                  DCL=    100             ;DC LOW (BIT #6)
020000                  OPE=    20000           ;OPERATOR PLUG ERROR (BIT #13)
040000                  SKI=    40000           ;SEEK INCOMPLETE (BIT #14)
100000                  OCYL=   100000          ;OFF CYLINDER ERROR (BIT #15)

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

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

                        ;;*****************************************************************

                        .SBTTL  RP04/5/6 DRIVER COMMANDS

                        ;;*****************************************************************

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

```
                    000161                  WRTDAT  =       161             ;WRITE DATA
                    000163                  WRTHD   =       163             ;WRITE HEADER & DATA
                    000171                  RDDAT   =       171             ;READ DATA
                    000173                  RDHD    =       173             ;READ HEADER & DATA

                                            .SBTTL  TRAP CATCHER

                    000000                          .=0
                                            ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
                                            ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                                            ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
                    000174                          .=174
000174  000000                             DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
000176  000000                             SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
                                            .SBTTL  STARTING ADDRESS(ES)
000200  000137  004116                              JMP     @#START1        ;;JUMP TO STARTING ADDRESS OF PROGRAM
000204  000137  004106                              JMP     @#START         ;CHANGE THE RH11 UNIBUS ADDRESS
                                                                            ;AFTER INITIAL START

                                            .SBTTL  ACT11 HOOKS

                                            ;;******************************************************************
                                            ;HOOKS REQUIRED BY ACT11
                    000210                          $SVPC=.                 ;SAVE PC
                    000046                          .=46
000046  005360                                      $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
                    000052                          .=52
000052  040000                                      .WORD   40000           ;;2)SET LOC.52 TO 40000
                    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 REG. 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
```

```
001220  000000  000000  000000  DATE:   .WORD   0,0,0,0,0       ;OPERATOR ENTERED DATE
001226  000000  000000
001232  000000  000000  000000  OPERID: .WORD   0,0,0,0         ;OPERATOR ID
001240  000000
001242  000000                  DRIVE:  .WORD   0               ;DRIVE # STORAGE: ERRORS 1-5 & 10
001244  000000                  ATTN:   .WORD   0               ;ATTN REG STORAGE: ERRORS 1-5 & 10
001246  000000                  UNIT:   .WORD   0               ;DRIVE # STORAGE FOR PRINTOUT
001250  000000                  MASK:   .WORD   0               ;ERROR RETRY REGISTER MASK
001252     000     000          RETRY:  .BYTE   0,0             ;ERROR RETRY LIMIT IN THE LOWER BYTE
                                                                ;RETRY COUNT IN THE UPPER BYTE
001254  000003                  FAIRNS: .WORD   3               ;MAXIMUM TIME IN QUEUE VALUE
001256  000000                  LSTAD:  .WORD   0               ;STORE LAST MEMORY ADDRESS HERE
001260  000000                  CHGADR: .WORD   0               ;CHANGE RH11 UNIBUS ADDRESS FLAG
001262  000000                  CFLAG:  .WORD   0               ;'CONTROL C' FLAG
001264  000000                  BADSEC: .WORD   0               ;BAD SECTOR/TRACK FLAG
001266  000000                  HOUR:   .WORD   0               ;HOUR COUNT STORED HERE (MAXIMUM - 999.)
001270  000000                  MINUTE: .WORD   0               ;MINUTE'S COUNT STORED HERE
001272  000000                  SECOND: .WORD   0               ;SECOND'S COUNT STORED HERE
001274  000000                  SIXTEE: .WORD   0               ;TIMER ROUTINE COUNTER (FOR ONE SECOND)
001276  177777                  ZROIND: .WORD   -1              ;ZERO INDICATOR FOR THE DATA COMPARE ROUTINE
001300     000                  FRSTER: .BYTE   0               ;DATA COMPARE ERROR FLAG
                                                                ;IF > 0, PROCESSING 'DCKER' OR CAN'T MATCH PATTERN
                                                                ;IF < 0, MISCOMPARSION FOUND
001301     000                          .BYTE   0               ;MISCOMPARSION OR CAN'T MATCH PATTERN FLAG
                                                                ;IF < 0, ERROR IN BUFFER
001302  000000                  SAVER1: .WORD   0               ;SAVE R1 HERE
001304  000000                  SAVER5: .WORD   0               ;SAVE R5 HERE
001306  000000                  ERCTR:  .WORD   0               ;NUMBER OF ERRORS
001310  000000                  LIMIT:  .WORD   0               ;DISPLAY LIMIT
001312  000000                  CMCNT:  .WORD   0               ;WORD COUNT
001314  000000                  CMCYL:  .WORD   0               ;CYLINDER ADDRESS
001316     000                  CMSEC:  .BYTE   0               ;SECTOR ADDRESS
001317     000                  CMTRK:  .BYTE   0               ;TRACK ADDRESS
001320  000000                  ECBIT:  .WORD   0               ;ERROR BURST BIT OFFSET
001322  000000                  ECSEC:  .WORD   0               ;ERROR BURST WORD OFFSET (RELATIVE TO SECTOR)
001324  000000                  ECMSK0: .WORD   0               ;CORRECTION MASK FOR FIRST ERROR WORD
001326  000000                  ECMSK1: .WORD   0               ;CORRECTION MASK FOR SECOND ERROR WORD
001330  000000                  ECWRD:  .WORD   0               ;LOCATION OF FIRST ERROR WORD
001332  000000                  ECGD:   .WORD   0               ;GOOD DATA, FIRST WORD
001334  000000                  ECBAD0: .WORD   0               ;BAD DATA, FIRST WORD
001336  000000                  ECWRD1: .WORD   0               ;LOCATION OF SECOND ERROR WORD
001340  000000                  ECGD1:  .WORD   0               ;GOOD DATA, SECOND WORD
001342  000000                  ECBAD1: .WORD   0               ;BAD DATA, SECOND WORD
001344  000025                  SECLMT: .WORD   21.             ;SECTOR ADDRESS LIMIT
001346  000022                  TRKLMT: .WORD   18.             ;TRACK ADDRESS LIMIT
001350  000632                  CYLIMT: .WORD   410.            ;CYLINDER ADDRESS LIMIT FOR RP04/5'S
                                                                ;(CHANGED TO 814. FOR RP06)

                                ;;*******************************************************************

                                .SBTTL  COMMON PARAMETERS

                                ;;*******************************************************************

                                ;PROGRAM USES THESE PARAMETERS TO DETERMINE REGULAR END OF PASS
001352  002740                  ENDCN:  .WORD   002740          ;1.875X10↑8 WORDS (10) [3X10↑9 BITS]
```

```
2318  001354  005455                          .WORD   005455          ;MSW
2319  001356  143300                ENDSK:    .WORD   143300          ;3 X 10↑6 SEEKS  (LSW)
2320  001360  000055                          .WORD   55              ;MSW
2321                                ;PROGRAM USES THESE PARAMETERS TO DETERMINE Q.V. END OF PASS
2322  001362  120274                QVCON:    .WORD   120274          ;2.3437X10↑7 WORDS (10)
2323  001364  000005                          .WORD   5               ;MSW
2324  001366  134330                QVSEK:    .WORD   134330          ;3.75 X 10↑5 SEEKS (10)
2325  001370  000005                          .WORD   5               ;MSW
2326                                ;THE NUMBERS TO DETERMINE END OF PASS ARE LOADED IN HERE BY THE PROGRAM.
2327                                ;THE FIRST TIME THROUGH, THE QV PARAMETERS ARE USED, AFTER THAT THE
2328                                ;REGULAR PARAMETERS ARE USED.
2329  001372  000000                ENDCON:   .WORD   0
2330  001374  000000                          .WORD   0
2331  001376  000000                ENDSEK:   .WORD   0
2332  001400  000000                          .WORD   0
2333
2334  001402  000001                PASCNT:   .WORD   1               ;NUMBER OF PASSES TO END OF TEST
2335  001404  000000                MAXDL:    .WORD   0               ;MAXIMUM DATA TRANFER SIZE IN WORDS
2336                                                                  ;(FILLED BY PROGRAM AT STARTUP OR BY OPERATOR
2337                                                                  ;DURING PARAMETER ENTRY DIALOG.)
2338  001406  000144                MAXER:    .WORD   100.            ;MAXIMUM ERRORS - 100(10)
2339  001410  000005  000000        INTRVL:   .WORD   5,0             ;FIRST WORD IS THE PERFORMANCE TYPEOUT INTERVAL
2340                                                                  ;(IN MINUTES).  SECOND WORD IS THE INTERVAL
2341                                                                  ;COUNTER
2342                                                                  ;COUNTER.  UPPER BYTE IS VALUE.
2343  001414  000004                CMPLMT:   .WORD   4               ;NUMBER OF COMPARE ERRORS TYPED OUT
2344  001416  000001                FORMAT:   .WORD   1               ;IF NOT EQ 0, ALLOW WRITE HEADER & DATA ORDERS
2345                                                                  ;IF EQ 0, DO NOT ALLOW WRITE HEADER & DATA ORDERS
2346  001420  000000                WCSEL:    .WORD   0               ;IF EQ TO 0, GENERATE A RANDOM WORD COUNT
2347                                                                  ;  FOR THE OPERATION.
2348                                                                  ;IF NOT EQ TO 0, USE THE VALUE IN 'MAXDL' FOR
2349                                                                  ;  THE WORD COUNT
2350  001422  000003                RATIO:    .WORD   3               ;READ/WRITE RATIO [RANGE 0 - 7]
2351                                                                  ;0 - 0/8          (READ/WRITE)
2352                                                                  ;1 - 7/1
2353                                                                  ;2 - 6/2
2354                                                                  ;3 - 5/3
2355                                                                  ;4 - 4/4
2356                                                                  ;5 - 3/5
2357                                                                  ;6 - 2/6
2358                                                                  ;7 - 1/7
2359  001424  000001                AUTOCK:   .WORD   1               ;IF NOT EQ 0, DO AN APPROPRITE WRITE
2360                                                                  ;  CHECK AFTER EACH WRITE ORDER.
2361                                                                  ;IF EQ 0, SELECT WRITE CHECK ORDERS
2362                                                                  ;  RANDOMLY.
2363  001426  000001                NOTPRT:   .WORD   1               ;IF EQ 1, DO NOT PRINT DATA ERROR MESSAGES
2364                                                                  ;  ASSOCIATED WITH OPERATOR SPECIFIED
2365                                                                  ;  BAD PACK AREAS.
2366                                                                  ;IF NOT EQ 0, PRINT ERROR MESSAGES RELATING TO
2367                                                                  ;  THESE AREAS.
2368  001430  000001                ENDET:    .WORD   1               ;IF NOT EQ 0, END OF PASS DETERMINED
2369                                                                  ;  BY THE 'WORDS READ' COUNT.
2370                                                                  ;IF EQ 0, END OF PASS DETERMINED
2371                                                                  ;  BY THE SEEK COUNT.
2372
2373                                ;;*****************************************************************
```

```
                                .SBTTL  VALUES FOR FIRST OPERATION

                                ;;*****************************************************************

001432  000010                BEGPAT: .WORD   10              ;STARTING PATTERN CODE [RANGE 1 - 17 (OCTAL)]
001434  000005                BEGCOD: .WORD   5               ;STARTING COMMAND CODE [RANGE 0 - 5]
                                                                ;0 = WRITE CHECK DATA ('WCKD')
                                                                ;1 = WRITE CHECK HEADER & DATA ('WCHKHD')
                                                                ;2 = WRITE DATA ('WRTDAT')
                                                                ;3 = WRITE HEADER & DATA ('WRTHD')
                                                                ;4 = READ DATA ('RDDAT')
                                                                ;5 = READ HEADER & DATA ('RDHD')
001436  000404                BEGSIZ: .WORD   404             ;STARTING RECORD SIZE [RANGE 4 - MAXMEM]
                                                                ;NOTE:  THE SIZE MUST BE AT LEAST 4 IF
                                                                ;WRITE DATA OR READ DATA; THE SIZE MUST
                                                                ;BE AT LEAST 8 IF WRITE HEADER AND
                                                                ;DATA OR READ HEADER AND DATA.
                                                                ;IF THE SIZE IS GREATER THAN 1 SECTOR, THE
                                                                ;SIZE MUST ALLOW FOR OVERLAPPING 4 OR 8
                                                                ;WORDS INTO THE LAST SECTOR USED.

                                ;;*****************************************************************

                                .SBTTL  TABLES, CONSTANTS, AND VARIABLE LOCATIONS

                                ;;*****************************************************************

001440  000000  000000  000000  ORDERQ: .WORD   0,0,0,0,0,0,0,0,0  ;LIST OF DRIVES PERFORMING COMMANDS
001446  000000  000000  000000
001454  000000  000000  000000

001462  000000                  ASNLST: .WORD   0               ;A BIT SET IS AN ASSIGNED DRIVE

001464  000000  000000  000000  DUNIT:  .WORD   0,0,0,0,0,0,0,0,0       ;ADDRESSES OF DRIVES TO BE DEASSIGNED
001472  000000  000000  000000
001500  000000  000000  000000

001506  000000  000000  000000  NEWUNT: .WORD   0,0,0,0,0,0,0,0,0       ;ADDRESSES OF NEWLY ASSIGNED DRIVES
001514  000000  000000  000000
001522  000000  000000  000000

001530  000000  000000  000000  AVAIL:  .WORD   0,0,0,0,0,0,0,0,0       ;LIST OF DRIVES WAITING FOR BUFFERS/PARAMETERS
001536  000000  000000  000000
001544  000000  000000  000000

001552  000000  000000  000000  WAIT:   .WORD   0,0,0,0,0,0,0,0,0       ;LIST OF DRIVES WAITING FOR BUFFERS
001560  000000  000000  000000
001566  000000  000000  000000

001574  000000  000000  000000  PARQ:   .WORD   0,0,0,0,0,0,0,0,0       ;LIST OF DRIVES WAITING FOR NEXT PARAMETERS
001602  000000  000000  000000
001610  000000  000000  000000

001616  000000                  BUFTBL: .WORD   0               ;BUFFER ALLOCATION TABLE ENTRY COUNT
001620  000000  000000                  .WORD   0,0
```

```
                                                .WORD   0,0
                                                .WORD   0,0
                                                .WORD   0,0
                                                .WORD   0,0
                                                .WORD   0,0
                                                .WORD   0,0
                                                .WORD   0,0
                                                .WORD   0,0
                                                .WORD   0,0
                                                .WORD   0,0
                                                .WORD   0,0
                                                .WORD   0,0
                                                .WORD   0,0
                                                .WORD   0,0
                                                .WORD   0,0
                                                .WORD   0,0

001740  041670                        BLKADR: .WORD   DRIVE0           ;ADDRESS OF THE BLOCK FOR DRIVE 0
001742  042174                                .WORD   DRIVE1           ;ADDRESS OF THE BLOCK FOR DRIVE 1
001744  042500                                .WORD   DRIVE2           ;ADDRESS OF THE BLOCK FOR DRIVE 2
001746  043004                                .WORD   DRIVE3           ;ADDRESS OF THE BLOCK FOR DRIVE 3
001750  043310                                .WORD   DRIVE4           ;ADDRESS OF THE BLOCK FOR DRIVE 4
001752  043614                                .WORD   DRIVE5           ;ADDRESS OF THE BLOCK FOR DRIVE 5
001754  044120                                .WORD   DRIVE6           ;ADDRESS OF THE BLOCK FOR DRIVE 6
001756  044424                                .WORD   DRIVE7           ;ADDRESS OF THE BLOCK FOR DRIVE 7

001760      151                     COMTBL: .BYTE   WCKD             ;WRITE CHECK DATA
001761      153                             .BYTE   WCKHD            ;WRITE CHECK HEADER AND DATA
001762      161                             .BYTE   WRTDAT           ;WRITE DATA
001763      163                             .BYTE   WRTHD            ;WRITE HEADER AND DATA
001764      171                             .BYTE   RDDAT            ;READ DATA
001765      173                             .BYTE   RDHD             ;READ HEADER AND DATA

001766      002                     OPTBL:  .BYTE   2                ;UNLOAD
001767      004                             .BYTE   4                ;SEEK
001770      006                             .BYTE   6                ;RECAL
001771      010                             .BYTE   10               ;DRIVE CLEAR
001772      012                             .BYTE   12               ;RELEASE
001773      014                             .BYTE   14               ;OFFSET
001774      016                             .BYTE   16               ;RETURN TO CENTERLINE
001775      020                             .BYTE   20               ;READIN PRESET
001776      022                             .BYTE   22               ;PACK ACKNOWLEDGE
001777      030                             .BYTE   30               ;SEARCH
002000      050                             .BYTE   50               ;WRITE CHECK DATA
002001      052                             .BYTE   52               ;WRITE CHECK HEADER AND DATA
002002      060                             .BYTE   60               ;WRITE DATA
002003      062                             .BYTE   62               ;WRITE HEADER AND DATA
002004      070                             .BYTE   70               ;READ DATA
002005      072                             .BYTE   72               ;READ HEADER AND DATA
002006      377                             .BYTE   -1               ;TERMINATOR

            002010                  .EVEN
```

```
2486  002010  047125  047514  042101  MNTBL:  .ASCIZ  /UNLOAD /
2487  002016  000040
2488  002020  042523  045505  020040          .ASCIZ  /SEEK   /
2489  002026  000040
2490  002030  042522  040503  020114          .ASCIZ  /RECAL  /
2491  002036  000040
2492  002040  051104  041526  051114          .ASCIZ  /DRVCLR /
2493  002046  000040
2494  002050  042522  051514  020105          .ASCIZ  /RELSE  /
2495  002056  000040
2496  002060  043117  051506  052105          .ASCIZ  /OFFSET /
2497  002066  000040
2498  002070  052122  020103  020040          .ASCIZ  /RTC    /
2499  002076  000040
2500  002100  042522  042101  047111          .ASCIZ  /READIN /
2501  002106  000040
2502  002110  040520  045503  020040          .ASCIZ  /PACK   /
2503  002116  000040
2504  002120  042523  051101  044103          .ASCIZ  /SEARCH /
2505  002126  000040
2506  002130  041527  042113  020040          .ASCIZ  /WCKD   /
2507  002136  000040
2508  002140  041527  044113  020104          .ASCIZ  /WCKHD  /
2509  002146  000040
2510  002150  051127  042124  052101          .ASCIZ  /WRTDAT /
2511  002156  000040
2512  002160  051127  044124  020104          .ASCIZ  /WRTHD  /
2513  002166  000040
2514  002170  042122  040504  020124          .ASCIZ  /RDDAT  /
2515  002176  000040
2516  002200  042122  042110  020040          .ASCIZ  /RDHD   /
2517  002206  000040
2518  002210  047516  042516  020040          .ASCIZ  /NONE   /
2519  002216  000040
2520
2521  002220     000                  OFFCOD: .BYTE   0                       ;OFFSET CODE TABLE
2522  002221     010                          .BYTE   10                      ;+200 U INCHES
2523  002222     210                          .BYTE   210                     ;-200 U INCHES
2524  002223     020                          .BYTE   20                      ;+400 U INCHES
2525  002224     220                          .BYTE   220                     ;-400 U INCHES
2526  002225     030                          .BYTE   30                      ;+600 U INCHES
2527  002226     230     000                  .BYTE   230,0                   ;-600 U INCHES, TERMINATOR
2528  002230     020                          .BYTE   20                      ;+400 U INCHES
2529  002231     220                          .BYTE   220                     ;-400 U INCHES
2530  002232     040                          .BYTE   40                      ;+800 U INCHES
2531  002233     240                          .BYTE   240                     ;-800 U INCHES
2532  002234     060                          .BYTE   60                      ;+1200 U INCHES
2533  002235     260     000                  .BYTE   260,0                   ;-1200 U INCHES, TERMINATOR
2534          002240                  .EVEN
2535
2536  002240  002274                  OFMTBL: .WORD   OFMSG0                  ;1ST OFFSET MESSAGE
2537  002242  002327                          .WORD   OFMSG1                  ;2ND OFFSET MESSAGE
2538  002244  002363                          .WORD   OFMSG2                  ;3RD OFFSET MESSAGE
2539  002246  002417                          .WORD   OFMSG3                  ;4TH OFFSET MESSAGE
2540  002250  002453                          .WORD   OFMSG4                  ;5TH OFFSET MESSAGE
2541  002252  002507                          .WORD   OFMSG5                  ;6TH OFFSET MESSAGE
```

```
                                                .WORD   OFMSG6          ;7TH OFFSET MESSAGE
                                                .WORD   OFMSG0          ;1ST OFFSET MESSAGE
                                                .WORD   OFMSG3          ;4TH OFFSET MESSAGE
                                                .WORD   OFMSG4          ;5TH OFFSET MESSAGE
                                                .WORD   OFMSG7          ;8TH OFFSET MESSAGE
                                                .WORD   OFMSG8          ;9TH OFFSET MESSAGE
                                                .WORD   OFMSG9          ;10TH OFFSET MESSAGE
                                                .WORD   OFMSGA          ;11TH OFFSET MESSAGE

002274  043101  042524  020122  OFMSG0: .ASCIZ  /AFTER RETRY WITHOUT OFFSET/
002302  042522  051124  020131
002310  044527  044124  052517
002316  020124  043117  051506
002324  052105     000
002327     101  020124  043117  OFMSG1: .ASCIZ  /AT OFFSET +200 MICRO-INCHES/
002334  051506  052105  025440
002342  030062  020060  044515
002350  051103  026517  047111
002356  044103  051505     000
002363     101  020124  043117  OFMSG2: .ASCIZ  /AT OFFSET -200 MICRO-INCHES/
002370  051506  052105  026440
002376  030062  020060  044515
002404  051103  026517  047111
002412  044103  051505     000
002417     101  020124  043117  OFMSG3: .ASCIZ  /AT OFFSET +400 MICRO-INCHES/
002424  051506  052105  025440
002432  030064  020060  044515
002440  051103  026517  047111
002446  044103  051505     000
002453     101  020124  043117  OFMSG4: .ASCIZ  /AT OFFSET -400 MICRO-INCHES/
002460  051506  052105  026440
002466  030064  020060  044515
002474  051103  026517  047111
002502  044103  051505     000
002507     101  020124  043117  OFMSG5: .ASCIZ  /AT OFFSET +600 MICRO-INCHES/
002514  051506  052105  025440
002522  030066  020060  044515
002530  051103  026517  047111
002536  044103  051505     000
002543     101  020124  043117  OFMSG6: .ASCIZ  /AT OFFSET -600 MICRO-INCHES/
002550  051506  052105  026440
002556  030066  020060  044515
002564  051103  026517  047111
002572  044103  051505     000
002577     101  020124  043117  OFMSG7: .ASCIZ  /AT OFFSET +800 MICRO-INCHES/
002604  051506  052105  025440
002612  030070  020060  044515
002620  051103  026517  047111
002626  044103  051505     000
002633     101  020124  043117  OFMSG8: .ASCIZ  /AT OFFSET -800 MICRO-INCHES/
002640  051506  052105  026440
002646  030070  020060  044515
002654  051103  026517  047111
002662  044103  051505     000
002667     101  020124  043117  OFMSG9: .ASCIZ  /AT OFFSET +1200 MICRO-INCHES/
002674  051506  052105  025440
```

```
 2598   002702   031061   030060   046440
 2599   002710   041511   047522   044455
 2600   002716   041516   042510   000123
 2601   002724   052101   047440   043106  OFMSGA: .ASCIZ  /AT OFFSET -1200 MICRO-INCHES/
 2602   002732   042523   020124   030455
 2603   002740   030062   020060   044515
 2604   002746   051103   026517   047111
 2605   002754   044103   051505      000
 2606
 2607            002762                           .EVEN
 2608
 2609                                             ;;*****************************************************************
 2610
 2611                                             .SBTTL  DATA PATTERNS
 2612
 2613                                             ;;*****************************************************************
 2614
 2615   002762   000000                           STNDAT: .WORD   0               ;STANDARD DATA PATTERN POINTER TABLE
 2616   002764   003066                                   .WORD   DATA1
 2617   002766   003126                                   .WORD   DATA1+40
 2618   002770   003166                                   .WORD   DATA1+100
 2619   002772   003226                                   .WORD   DATA1+140
 2620   002774   003266                                   .WORD   DATA1+200
 2621   002776   003326                                   .WORD   DATA1+240
 2622   003000   003366                                   .WORD   DATA1+300
 2623   003002   003426                                   .WORD   DATA1+340
 2624   003004   003466                                   .WORD   DATA1+400
 2625   003006   003526                                   .WORD   DATA1+440
 2626   003010   003566                                   .WORD   DATA1+500
 2627   003012   003626                                   .WORD   DATA1+540
 2628   003014   003666                                   .WORD   DATA1+600
 2629   003016   003726                                   .WORD   DATA1+640
 2630   003020   003766                                   .WORD   DATA1+700
 2631   003022   003026                                   .WORD   DATA0           ;ZEROES
 2632   003024   003730                                   .WORD   DATA1+642       ;ONES
 2633
 2634   003026   000000                           DATA0:  .WORD   0               ;DUMMY DATA PATTERN
 2635   003030   000000                                   .WORD   0
 2636   003032   000000                                   .WORD   0
 2637   003034   000000                                   .WORD   0
 2638   003036   000000                                   .WORD   0
 2639   003040   000000                                   .WORD   0
 2640   003042   000000                                   .WORD   0
 2641   003044   000000                                   .WORD   0
 2642   003046   000000                                   .WORD   0
 2643   003050   000000                                   .WORD   0
 2644   003052   000000                                   .WORD   0
 2645   003054   000000                                   .WORD   0
 2646   003056   000000                                   .WORD   0
 2647   003060   000000                                   .WORD   0
 2648   003062   000000                                   .WORD   0
 2649   003064   000000                                   .WORD   0
 2650
 2651   003066   000001                           DATA1:  .WORD   000001  ;STANDARD PATTERN 1
 2652   003070   000003                                   .WORD   000003
 2653   003072   000007                                   .WORD   000007
```

```
2654  003074  000017                    .WORD   000017
2655  003076  000037                    .WORD   000037
2656  003100  000077                    .WORD   000077
2657  003102  000177                    .WORD   000177
2658  003104  000377                    .WORD   000377
2659  003106  000777                    .WORD   000777
2660  003110  001777                    .WORD   001777
2661  003112  003777                    .WORD   003777
2662  003114  007777                    .WORD   007777
2663  003116  017777                    .WORD   017777
2664  003120  037777                    .WORD   037777
2665  003122  077777                    .WORD   077777
2666  003124  177777                    .WORD   177777
2667
2668  003126  177776                    .WORD   177776  ;STANDARD PATTERN 2
2669  003130  177774                    .WORD   177774
2670  003132  177770                    .WORD   177770
2671  003134  177760                    .WORD   177760
2672  003136  177740                    .WORD   177740
2673  003140  177700                    .WORD   177700
2674  003142  177600                    .WORD   177600
2675  003144  177400                    .WORD   177400
2676  003146  177000                    .WORD   177000
2677  003150  176000                    .WORD   176000
2678  003152  174000                    .WORD   174000
2679  003154  170000                    .WORD   170000
2680  003156  160000                    .WORD   160000
2681  003160  140000                    .WORD   140000
2682  003162  100000                    .WORD   100000
2683  003164  000000                    .WORD   000000
2684
2685  003166  000000                    .WORD   000000  ;STANDARD PATTERN 3
2686  003170  000000                    .WORD   000000
2687  003172  000000                    .WORD   000000
2688  003174  177777                    .WORD   177777
2689  003176  177777                    .WORD   177777
2690  003200  177777                    .WORD   177777
2691  003202  000000                    .WORD   000000
2692  003204  000000                    .WORD   000000
2693  003206  177777                    .WORD   177777
2694  003210  177777                    .WORD   177777
2695  003212  000000                    .WORD   000000
2696  003214  177777                    .WORD   177777
2697  003216  000000                    .WORD   000000
2698  003220  177777                    .WORD   177777
2699  003222  000000                    .WORD   000000
2700  003224  177777                    .WORD   177777
2701
2702  003226  000000                    .WORD   000000  ;STANDARD PATTERN 4
2703  003230  010421                    .WORD   010421
2704  003232  021042                    .WORD   021042
2705  003234  031463                    .WORD   031463
2706  003236  042104                    .WORD   042104
2707  003240  052525                    .WORD   052525
2708  003242  063146                    .WORD   063146
2709  003244  073567                    .WORD   073567
```

```
2710  003246  104210                          .WORD   104210
2711  003250  114631                          .WORD   114631
2712  003252  125252                          .WORD   125252
2713  003254  135673                          .WORD   135673
2714  003256  146314                          .WORD   146314
2715  003260  156735                          .WORD   156735
2716  003262  167356                          .WORD   167356
2717  003264  177777                          .WORD   177777
2718
2719  003266  052525                          .WORD   052525  ;STANDARD PATTERN 5
2720  003270  052525                          .WORD   052525
2721  003272  052525                          .WORD   052525
2722  003274  125252                          .WORD   125252
2723  003276  125252                          .WORD   125252
2724  003300  125252                          .WORD   125252
2725  003302  052525                          .WORD   052525
2726  003304  052525                          .WORD   052525
2727  003306  125252                          .WORD   125252
2728  003310  125252                          .WORD   125252
2729  003312  052525                          .WORD   052525
2730  003314  125252                          .WORD   125252
2731  003316  052525                          .WORD   052525
2732  003320  125252                          .WORD   125252
2733  003322  052525                          .WORD   052525
2734  003324  125252                          .WORD   125252
2735
2736  003326  007417                          .WORD   007417  ;STANDARD PATTERN 6
2737  003330  007417                          .WORD   007417
2738  003332  007417                          .WORD   007417
2739  003334  170360                          .WORD   170360
2740  003336  170360                          .WORD   170360
2741  003340  170360                          .WORD   170360
2742  003342  007417                          .WORD   007417
2743  003344  007417                          .WORD   007417
2744  003346  170360                          .WORD   170360
2745  003350  170360                          .WORD   170360
2746  003352  007417                          .WORD   007417
2747  003354  170360                          .WORD   170360
2748  003356  007417                          .WORD   007417
2749  003360  170360                          .WORD   170360
2750  003362  007417                          .WORD   007417
2751  003364  170360                          .WORD   170360
2752
2753  003366  026455                          .WORD   026455  ;STANDARD PATTERN 7
2754  003370  026455                          .WORD   026455
2755  003372  026455                          .WORD   026455
2756  003374  151322                          .WORD   151322
2757  003376  151322                          .WORD   151322
2758  003400  151322                          .WORD   151322
2759  003402  026455                          .WORD   026455
2760  003404  026455                          .WORD   026455
2761  003406  151322                          .WORD   151322
2762  003410  151322                          .WORD   151322
2763  003412  026455                          .WORD   026455
2764  003414  151322                          .WORD   151322
2765  003416  026455                          .WORD   026455
```

```
2766   003420   151322                          .WORD   151322
2767   003422   026455                          .WORD   026455
2768   003424   151322                          .WORD   151322
2769
2770   003426   165555                          .WORD   165555  ;STANDARD PATTERN 8
2771   003430   133333                          .WORD   133333
2772   003432   165555                          .WORD   165555
2773   003434   133333                          .WORD   133333
2774   003436   165555                          .WORD   165555
2775   003440   133333                          .WORD   133333
2776   003442   165555                          .WORD   165555
2777   003444   133333                          .WORD   133333
2778   003446   165555                          .WORD   165555
2779   003450   133333                          .WORD   133333
2780   003452   165555                          .WORD   165555
2781   003454   133333                          .WORD   133333
2782   003456   165555                          .WORD   165555
2783   003460   133333                          .WORD   133333
2784   003462   165555                          .WORD   165555
2785   003464   133333                          .WORD   133333
2786
2787   003466   000001                          .WORD   000001  ;STANDARD PATTERN 9
2788   003470   000002                          .WORD   000002
2789   003472   000004                          .WORD   000004
2790   003474   000010                          .WORD   000010
2791   003476   000020                          .WORD   000020
2792   003500   000040                          .WORD   000040
2793   003502   000100                          .WORD   000100
2794   003504   000200                          .WORD   000200
2795   003506   000400                          .WORD   000400
2796   003510   001000                          .WORD   001000
2797   003512   002000                          .WORD   002000
2798   003514   004000                          .WORD   004000
2799   003516   010000                          .WORD   010000
2800   003520   020000                          .WORD   020000
2801   003522   040000                          .WORD   040000
2802   003524   100000                          .WORD   100000
2803
2804   003526   177776                          .WORD   177776  ;STANDARD PATTERN 10
2805   003530   177775                          .WORD   177775
2806   003532   177773                          .WORD   177773
2807   003534   177767                          .WORD   177767
2808   003536   177757                          .WORD   177757
2809   003540   177737                          .WORD   177737
2810   003542   177677                          .WORD   177677
2811   003544   177577                          .WORD   177577
2812   003546   177377                          .WORD   177377
2813   003550   176777                          .WORD   176777
2814   003552   175777                          .WORD   175777
2815   003554   173777                          .WORD   173777
2816   003556   167777                          .WORD   167777
2817   003560   157777                          .WORD   157777
2818   003562   137777                          .WORD   137777
2819   003564   077777                          .WORD   077777
2820
2821   003566   172666                          .WORD   172666  ;STANDARD PATTERN 11
```

```
2822  003570  155555                  .WORD   155555
2823  003572  172666                  .WORD   172666
2824  003574  155555                  .WORD   155555
2825  003576  172666                  .WORD   172666
2826  003600  155555                  .WORD   155555
2827  003602  172666                  .WORD   172666
2828  003604  155555                  .WORD   155555
2829  003606  172666                  .WORD   172666
2830  003610  155555                  .WORD   155555
2831  003612  172666                  .WORD   172666
2832  003614  155555                  .WORD   155555
2833  003616  172666                  .WORD   172666
2834  003620  155555                  .WORD   155555
2835  003622  172666                  .WORD   172666
2836  003624  155555                  .WORD   155555
2837
2838  003626  077777                  .WORD   077777  ;STANDARD PATTERN 12
2839  003630  137777                  .WORD   137777
2840  003632  157777                  .WORD   157777
2841  003634  167777                  .WORD   167777
2842  003636  173777                  .WORD   173777
2843  003640  175777                  .WORD   175777
2844  003642  176777                  .WORD   176777
2845  003644  177377                  .WORD   177377
2846  003646  177577                  .WORD   177577
2847  003650  177677                  .WORD   177677
2848  003652  177737                  .WORD   177737
2849  003654  177757                  .WORD   177757
2850  003656  177767                  .WORD   177767
2851  003660  177773                  .WORD   177773
2852  003662  177775                  .WORD   177775
2853  003664  177776                  .WORD   177776
2854
2855  003666  153333                  .WORD   153333  ;STANDARD PATTERN 13
2856  003670  066667                  .WORD   066667
2857  003672  153333                  .WORD   153333
2858  003674  066667                  .WORD   066667
2859  003676  153333                  .WORD   153333
2860  003700  066667                  .WORD   066667
2861  003702  153333                  .WORD   153333
2862  003704  066667                  .WORD   066667
2863  003706  153333                  .WORD   153333
2864  003710  066667                  .WORD   066667
2865  003712  153333                  .WORD   153333
2866  003714  066667                  .WORD   066667
2867  003716  153333                  .WORD   153333
2868  003720  066667                  .WORD   066667
2869  003722  153333                  .WORD   153333
2870  003724  066667                  .WORD   066667
2871
2872  003726  000000                  .WORD   000000  ;STANDARD PATTERN 14
2873  003730  177777                  .WORD   177777
2874  003732  177777                  .WORD   177777
2875  003734  177777                  .WORD   177777
2876  003736  177777                  .WORD   177777
2877  003740  177777                  .WORD   177777
```

```
2878  003742  177777                          .WORD   177777
2879  003744  177777                          .WORD   177777
2880  003746  177777                          .WORD   177777
2881  003750  177777                          .WORD   177777
2882  003752  177777                          .WORD   177777
2883  003754  177777                          .WORD   177777
2884  003756  177777                          .WORD   177777
2885  003760  177777                          .WORD   177777
2886  003762  177777                          .WORD   177777
2887  003764  177777                          .WORD   177777
2888
2889  003766  177777                          .WORD   177777  ;STANDARD PATTERN 15
2890  003770  000000                          .WORD   000000
2891  003772  000000                          .WORD   000000
2892  003774  000000                          .WORD   000000
2893  003776  000000                          .WORD   000000
2894  004000  000000                          .WORD   000000
2895  004002  000000                          .WORD   000000
2896  004004  000000                          .WORD   000000
2897  004006  000000                          .WORD   000000
2898  004010  000000                          .WORD   000000
2899  004012  000000                          .WORD   000000
2900  004014  000000                          .WORD   000000
2901  004016  000000                          .WORD   000000
2902  004020  000000                          .WORD   000000
2903  004022  000000                          .WORD   000000
2904  004024  000000                          .WORD   000000
2905
```

```
                                .SBTTL  ERROR POINTER TABLE

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

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

004026                          $ERRTB:
                                ;ERROR 1

                                ;ERROR 1

004026  045020                          EM1                     ;RH11 INTERRUPT OCCURRED (RPAS = 0)
004030  047460                          DH1
004032  050112                          DT1
004034  000000                          0

                                ;ERROR 2

004036  045063                          EM2                     ;UNEXPECTED ATTENTION OCCURRED
004040  047465                          DH2
004042  050116                          DT2
004044  000000                          0

                                ;ERROR 3

004046  045121                          EM3                     ;MASSBUS PARITY ERROR (MCPE=1)
004050  047542                          DH3
004052  050134                          DT3
004054  000000                          0

                                ;ERROR 4

004056  045157                          EM4                     ;MASSBUS PARITY ERROR (PAR=1)
004060  047570                          DH4
004062  050144                          DT4
004064  000000                          0

                                ;ERROR 5

004066  045214                          EM5                     ;ADDRESS PLUG BIT CHANGED
004070  047465                          DH2
004072  050116                          DT2
004074  000000                          0

                                ;ERROR 6

004076  045250                          EM6                     ;RH11 DIDN'T RESPOND TO ADDRESSING
004100  047627                          DH6
```

```
2962  004102  050156                        DT6
2963  004104  000000                        0
2964
2965                                  ;;*****************************************************************
2966
2967                                  .SBTTL  SETUP AND INITIALIZATION ROUTINE
2968
2969                                  ;       START ADDRESS = 200
2970                                  ;       ADDRESS TO CHANGE RH11 UNIBUS ADDRESS = 204
2971
2972                                  ;;*****************************************************************
2973
2974  004106  012737  177777  001260  START:  MOV     #-1,CHGADR      ;SET RH11 ADDRESS CHANGE FLAG
2975  004114  000402                        BR      START2          ;START THE PROGRAM
2976  004116  005037  001260        START1: CLR     CHGADR          ;CLEAR THE RH11 ADDRESS CHANGE FLAG
2977  004122  000005                START2: RESET                   ;CLEAR THE BUS
2978  004124  013737  001352  001372        MOV     ENDCN,ENDCON            ;SET UP FOR NORMAL PASS
2979  004132  013737  001354  001374        MOV     ENDCN+2,ENDCON+2
2980  004140  013737  001356  001376        MOV     ENDSK,ENDSEK
2981  004146  013737  001360  001400        MOV     ENDSK+2,ENDSEK+2
2982  004154                        SETUP:
2983                                  .SBTTL  INITIALIZE THE COMMON TAGS
2984                                  ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
2985  004154  012706  001100                MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
2986  004160  005026                        CLR     (R6)+           ;;CLEAR MEMORY LOCATION
2987  004162  022706  001140                CMP     #SWR,R6 ;;DONE?
2988  004166  001374                        BNE     .-6             ;;LOOP BACK IF NO
2989  004170  012706  001100                MOV     #STACK,SP       ;;SETUP THE STACK POINTER
2990                                  ;;INITIALIZE A FEW VECTORS
2991  004174  012737  030530  000030        MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
2992  004202  012737  000340  000032        MOV     #340,@#EMTVEC+2 ;;LEVEL 7
2993  004210  012737  033076  000034        MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
2994  004216  012737  000340  000036        MOV     #340,@#TRAPVEC+2;LEVEL 7
2995  004224  012737  176543  032462        MOV     #176543,$HINUM  ;;PRIME THE RANDOM NUMBER GENERATOR
2996  004232  012737  123456  032464        MOV     #123456,$LONUM  ;;BOTH HIGH AND LOW WORDS
2997                                  ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
2998                                  ;;EQUAL TO A "-1"  SETUP FOR A SOFTWARE SWITCH REGISTER.
2999  004240  013746  000004                MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
3000  004244  012737  004300  000004        MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
3001  004252  012737  177570  001140        MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
3002  004260  012737  177570  001142        MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
3003  004266  022777  177777  174644        CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
3004  004274  001012                        BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
3005                                                                ;;AND  THE HARDWARE SWR IS NOT = -1
3006  004276  000403                        BR      65$             ;;BRANCH IF NO TIMEOUT
3007  004300  012716  004306        64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
3008  004304  000002                        RTI
3009  004306  012737  000176  001140  65$:  MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
3010  004314  012737  000174  001142        MOV     #DISPREG,DISPLAY
3011  004322  012637  000004        66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
3012
3013  004326  012737  000240  000032        MOV     #240,@#EMTVEC+2   ;CHANGE EMT PRIORITY TO 5
3014  004334  012737  000240  000036        MOV     #240,@#TRAPVEC+2  ;CHANGE TRAP PRIORITY TO 5
3015  004342  005227  177777                INC     #-1             ;FIRST START ?
3016  004346  001017                        BNE     1$              ;BR IF NOT
3017  004350  023737  000042  000046        CMP     @#42,@#46       ;ARE WE IN ACT-11 AUTO MODE?
```

```
3018  004356  001413                        BEQ     1$                      ;BRANCH IF YES
3019  004360  104401  055362                TYPE    ,TITLE                  ;NAME AND MAINDEC NUMBER
3020  004364  005737  000042                TST     42                      ;AUTO ACCEPT OR CHAIN MODE ?
3021  004370  001006                        BNE     1$                      ;BR IF EITHER
3022  004372  122737  000011  000041        CMPB    #11,41                  ;LOADED FROM AN RP04/5/6 ?
3023  004400  001002                        BNE     1$                      ;BR IF NOT
3024  004402  104401  055464                TYPE    ,LOADRV                 ;INSTRUCT THE OPERATOR ON HOW TO TEST DRIVE 0
3025  004406  004737  030050        1$:     JSR     PC,$TKINT               ;TURN ON THE KEYBOARD INTERRUPT
3026                                .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
3027  004412  005737  000042                TST     @#42                    ;;ARE WE RUNNING UNDER XXDP/ACT?
3028  004416  001006                        BNE     67$                     ;;BRANCH IF YES
3029  004420  023727  001140  000176        CMP     SWR,#SWREG              ;;SOFTWARE SWITCH REG SELECTED?
3030  004426  001005                        BNE     68$                     ;;BRANCH IF NO
3031  004430  104406                        GTSWR                           ;;GET SOFT-SWR SETTINGS
3032  004432  000403                        BR      68$
3033  004434  112737  000001  001134  67$:  MOVB    #1,$AUTOB               ;;SET AUTO-MODE INDICATOR
3034  004442                        68$:
3035  004442  005227  177777                INC     #-1                     ;FIRST START ?
3036  004446  001015                        BNE     2$                      ;BR IF NOT
3037  004450  004737  054752                JSR     PC,BUSADR               ;CHECK RH11 BUS ADDRESS
3038  004454  013737  001170  033332        MOV     $RPADR,RPADR            ;RH11 ADDRESS
3039  004462  013737  001172  033334        MOV     $RPVEC,RPVEC            ;RH11 VECTOR ADDRESS
3040  004470  005737  000042                TST     @#42                    ;ACT-11 AUTO OR CHAIN MODE?
3041  004474  001002                        BNE     2$                      ;BRANCH IF EITHER, SKIP
3042                                                                        ;DATE & OPERATOR ID INPUT
3043  004476  004737  054466                JSR     PC,OPRDAT               ;GET THE DATE AND OPERATOR ID
3044  004502  005037  001214        2$:     CLR     STATIN                  ;CLEAR PERFORMANCE SUMMARY TYPEOUT FLAG
3045  004506  012705  001440                MOV     #ORDERQ,R5              ;START OF AREA TO CLEAR
3046  004512  005025                3$:     CLR     (R5)+
3047  004514  022705  001740                CMP     #BLKADR,R5              ;LOOK FOR END OF CLEAR AREA
3048  004520  001374                        BNE     3$                      ;BR IF NOT FINISHED
3049  004522  012706  001100                MOV     #STACK,SP               ;SETUP THE STACK POINTER
3050  004526  005037  177776                CLR     PS                      ;CLEAR THE PROCESSOR STATUS WORD
3051  004532  013737  001212  001274        MOV     HZ,SIXTEE               ;1/60 TH OR 1/50 TH SECOND COUNTER VALUE
3052  004540  005037  001266                CLR     HOUR                    ;CLEAR THE HOUR'S COUNTER
3053  004544  005037  001270                CLR     MINUTE                  ;CLEAR THE MINUTE'S COUNTER
3054  004550  005037  001272                CLR     SECOND                  ;CLEAR THE SECOND'S COUNTER
3055  004554  005037  001412                CLR     INTRVL+2                ;CLEAR INTERVAL COUNTER
3056  004560  005037  001216                CLR     PACK                    ;CLEAR THE 'R' OR 'W' COMMAND FLAG
3057  004564  005037  001262                CLR     CFLAG                   ;CLEAR THE 'CONTROL C' FLAG
3058  004570  042737  170000  001406        BIC     #170000,MAXER           ;MAKE SURE ERROR LIMITS ARE NOT TOO HIGH
3059
3060                                ;ROUTINE TO DETERMINE BUFFER AREA SIZE
3061
3062  004576  005227  177777        SIZMEM: INC     #-1                     ;SEE IF TIME TO SIZE MEMORY
3063  004602  001005                        BNE     1$                      ;BR IF NOT
3064  004604  004737  054646                JSR     PC,$SIZE                ;SEE HOW MUCH MEMORY ON SYSTEM
3065  004610  013737  054742  001256        MOV     $LSTAD,LSTAD            ;SAVE THE LAST ADDRESS
3066  004616  012737  000001  001616  1$:   MOV     #1,BUFTBL               ;LOAD NUMBER OF BUFFERS
3067  004624  012737  054466  001620        MOV     #ENDPGM,BUFTBL+2        ;STARTING ADDRESS OF BUFFER
3068  004632  013737  001256  001622        MOV     LSTAD,BUFTBL+4          ;LAST ADDR TO BUFFER ALLOCATION TABLE
3069  004640  162737  054466  001622        SUB     #ENDPGM,BUFTBL+4        ;SUBTRACT PROGRAM SPACE
3070  004646  000241                        CLC                             ;CLEAR THE 'C' BIT
3071  004650  006037  001622                ROR     BUFTBL+4                ;CONVERT TO WORD COUNT
3072  004654  162737  000144  001622        SUB     #100.,BUFTBL+4          ;SAVE ROOM FOR THE 'ABS' LOADER
3073  004662  023727  001256  100000        CMP     LSTAD,#100000           ;16K ON THE SYSTEM ?
```

```
3074  004670  103406                        BLO     3$              ;BR IF YES
3075  004672  105737  000041                TSTB    41              ;SEE WHO LOADED THE PROGRAM
3076  004676  001403                        BEQ     3$              ;BR IF LOADED BY PAPER TAPE
3077  004700  162737  002570  001622        SUB     #1400.,BUFTBL+4 ;SUBTRACT 'XXDP' LOADER SIZE
3078  004706  005737  001404          3$:   TST     MAXDL           ;VALUE IN 'MAXDL' ?
3079  004712  001012                        BNE     4$              ;BR IF VALUE IS
3080  004714  012737  013534  001404        MOV     #5980.,MAXDL    ;ASSUME FULL TRACK + 1 SEC MAXIMUM
3081  004722  023737  001404  001622        CMP     MAXDL,BUFTBL+4  ;IS THAT TOO LARGE ?
3082  004730  103403                        BLO     4$              ;BR IF NOT
3083  004732  013737  001622  001404        MOV     BUFTBL+4,MAXDL  ;USE MAX AVAIL MEMORY AS MAX BUFFER SIZE
3084  004740  013737  001622  053456  4$:   MOV     BUFTBL+4,PARLST+2  ;VALUE FOR THE PARAMETER TABLE
3085
3086                                        ;SEE IF THE OPERATOR WANTS TO CHANGE ANY PARAMETERS
3087
3088  004746  005737  000042          LKPAR:  TST     @#42            ;'XXDP' CHAIN MODE OR 'ACT11' OPERATION ?
3089  004752  001022                        BNE     SETVEC          ;BR IF YES
3090  004754  104401  053552                TYPE    ,ASKPAR         ;ASK FOR PARMETERS
3091  004760  104411                        RDLIN                   ;READ THE ENTRY
3092  004762  012605                        MOV     (SP)+,R5        ;ADDRESS OF ENTRY TO R5
3093  004764  122715  000131                CMPB    #'Y,(R5)        ;WAS ENTRY A 'Y' (YES)
3094  004770  001013                        BNE     SETVEC          ;BR IF NOT 'Y'
3095
3096  004772  012703  053454          ENTPR:  MOV     #PARLST,R3      ;PARAMETER TABLE ADDRESS
3097  004776  004737  026272                JSR     PC,PARENT       ;GET THE PARAMETER ENTRY
3098  005002  023727  001404  000004        CMP     MAXDL,#4        ;IS THE 'MAXDL' VALUE OK ?
3099  005010  103003                        BHIS    SETVEC          ;BR IF IT IS
3100  005012  012737  000004  001404        MOV     #4,MAXDL        ;SET 'MAXDL' TO THE MINIMUM VALUE
3101
3102                                        ;DISPLAY DRIVE STATUS AND SET UP THE OTHER SYSTEM DEVICES THAT
3103                                        ;       THE PROGRAM WILL USE
3104
3105  005020  004737  022412          SETVEC: JSR     PC,CKCLK        ;START THE CLOCK
3106  005024  004737  033350                JSR     PC,RPINIT       ;INITIALIZE THE RP04/5/6 DRIVER
3107  005030  012737  177777  033272        MOV     #-1,SAVEFG      ;SET THE SAVE REGISTERS FLAG
3108  005036  062727  177777  000000        ADD     #-1,#0          ;CHECK FOR FIRST START
3109  005044  103004                        BCC     11$             ;BR IF FIRST START
3110  005046  032777  000004  174064        BIT     #SW02,@SWR      ;TYPEOUT THE DRIVE STATUS TABLE ?
3111  005054  001076                        BNE     10$             ;BR IF NOT
3112  005056  012737  000340  177776  11$:  MOV     #PR7,PS         ;SET PRIORITY TO 7
3113  005064  005004                        CLR     R4              ;DRIVE TABLE POINTER
3114  005066  104401  001165                TYPE    ,$CRLF          ;CR-LF
3115  005072  104401  052344                TYPE    ,SYSTAT         ;TYPE STATUS HEADING
3116  005076                          1$:
3117  005076  010446                        MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
3118                                                                 ;;TYPE DRIVE NUMBER
3119  005100  104403                        TYPOS                   ;;GO TYPE--OCTAL ASCII
3120  005102     002                        .BYTE   2               ;;TYPE 2 DIGIT(S)
3121  005103     000                        .BYTE   0               ;;SUPPRESS LEADING ZEROS
3122  005104  104401  052153                TYPE    ,LIN4SP         ;SPACES
3123  005110  105764  033204                TSTB    DRVSTA(R4)      ;CHECK DRIVE'S STATUS
3124  005114  100416                        BMI     4$              ;BR IF UNSAFE
3125  005116  001020                        BNE     5$              ;BR IF ONLINE
3126  005120  105764  033214                TSTB    DRVTYP(R4)      ;SEE IF OFFLINE OR NONEXISTENT
3127  005124  001404                        BEQ     2$              ;BR IF NONEXISTENT
3128  005126  100006                        BPL     3$              ;BR IF OFFLINE
3129  005130  104401  052257                TYPE    ,NOTRP          ;DRIVE NOT AN RP04/5/6
```

```
3130  005134  000440                        BR      9$              ;CHECK NEXT DRIVE
3131  005136  104401  052300        2$:     TYPE    ,NOTPRS         ;DRIVE NOT PRESENT
3132  005142  000435                        BR      9$              ;CHECK NEXT DRIVE
3133  005144  104401  052166        3$:     TYPE    ,UNTOFF         ;DRIVE OFFLINE
3134  005150  000405                        BR      6$              ;PRINT DRIVE TYPE
3135  005152  104401  052334        4$:     TYPE    ,NOTSAF         ;DRIVE UNSAFE
3136  005156  000402                        BR      6$              ;PRINT DRIVE TYPE
3137  005160  104401  052177        5$:     TYPE    ,UNTON          ;DRIVE ONLINE
3138  005164  104401  052155        6$:     TYPE    ,LINSP          ;SPACES
3139  005170  012737  052364  005234        MOV     #RP04B,8$       ;ADDRESS OF RP04 MESSAGE
3140  005176  132764  000001  033214        BITB    #BIT00,DRVTYP(R4)  ;RP04 ?
3141  005204  001012                        BNE     7$              ;BR IF YES
3142  005206  012737  052371  005234        MOV     #RP05,8$        ;ADDRESS OF RP05 MESSAGE
3143  005214  132764  000002  033214        BITB    #BIT01,DRVTYP(R4)  ;RP05 ?
3144  005222  001003                        BNE     7$              ;BR IF YES
3145  005224  012737  052376  005234        MOV     #RP06,8$        ;ADDRESS OF RP06 MESSAGE
3146  005232  104401                7$:     TYPE                    ;TYPE THE DRIVE TYPE MESSAGE
3147  005234  000000                8$:     .WORD   0               ;MESSAGE ADDRESS HERE
3148  005236  104401  001165        9$:     TYPE    ,$CRLF          ;CR-LF
3149  005242  005204                        INC     R4              ;INCREMENT DRIVE NUMBER/TABLE POINTER
3150  005244  020427  000010                CMP     R4,#8.          ;FINISHED ?
3151  005250  001312                        BNE     1$              ;BR IF NOT
3152  005252  104401  001165        10$:    TYPE    ,$CRLF          ;CR-LF
3153  005256  005037  177776                CLR     @#PS            ;SET PRIORITY BACK TO '0'
3154  005262  000137  005266                JMP     MONTR           ;CHECK FOR 'XXDP' OR 'ACT11' MONITOR
3155
3156                                ;SETUP IF 'XXDP' OR 'ACT11' OPERATION
3157
3158  005266  005737  000042        MONTR:  TST     @#42            ;'XXDP' CHAIN MODE OR 'ACT11' AUTO ACCEPT
3159  005272  001402                        BEQ     1$              ;BR IF NEITHER
3160  005274  004737  024274                JSR     PC,ASGN2        ;ASSIGN DRIVES
3161  005300  005227  177777        1$:     INC     #-1             ;FIRST START ?
3162  005304  001011                        BNE     2$              ;BR IF NOT
3163  005306  105737  000041                TSTB    @#41            ;LOADED FROM PAPER TAPE ?
3164  005312  001406                        BEQ     2$              ;BR IF YES
3165  005314  023727  001256  100000        CMP     LSTAD,#100000   ;MORE THAN 16K ON THE SYSTEM ?
3166  005322  103002                        BHIS    2$              ;BR IF YES
3167  005324  104401  055663                TYPE    ,NOLOAD         ;TELL THE OPERATOR THAT THE 'XXDP' LOADER
3168                                                                ;WILL BE OVERWRITTEN
3169  005330  004737  030050        2$:     JSR     PC,$TKINT       ;INITIALIZE THE KEYBOARD INTERRUPT HANDLER
3170  005334  104401  053341                TYPE    ,INTDON         ;TYPE 'INITIALIZE COMPLETE'
3171  005340  000137  005522                JMP     MAIN1           ;START THE PROGRAM
3172
3173                                ;'XXDP' OR 'ACT11' END OF TEST ROUTINE
3174
3175  005344  013700  000042        $GET42: MOV     @#42,R0         ;MONITOR ADDRESS
3176  005350  001002                        BNE     1$              ;BR IF MONITOR
3177  005352  000137  005522                JMP     MAIN1           ;NONE, CONTINUE
3178  005356  000005                1$:     RESET                   ;CLEAR EVERYTHING
3179  005360  004710                $ENDAD: JSR     PC,(R0)         ;GO TO THE MONITOR
3180  005362  000240                        NOP                     ;SAVE ROOM
3181  005364  000240                        NOP                     ;FOR
3182  005366  000240                        NOP                     ;ACT11
3183  005370  000137  004116        $DOAGN: JMP     START1          ;START AGAIN
3184
3185                                ;;*****************************************************************
```

```
                                .SBTTL  MAIN PROGRAM

                                ;;*****************************************************************

005374  012703  000010  MAIN:   MOV     #8.,R3          ;DRIVE COUNTER
005400  012705  001464          MOV     #DUNIT,R5       ;ADDRESS OF 'DROP DRIVE' TABLE
005404  005715          1$:     TST     (R5)            ;SEE IF ENTRY AT PRESENT POSITION
005406  001011          2$:     BNE     3$              ;BR IF THERE IS ONE
005410  062705  000002          ADD     #2,R5           ;INCREMENT TO NEXT TABLE POSITION
005414  005303                  DEC     R3              ;DECREMENT DRIVE COUNTER
005416  001372                  BNE     1$              ;BR IF MORE TO CHECK
005420  005737  001462          TST     ASNLST          ;ANY DRIVES ACTIVE ?
005424  001036                  BNE     MAIN1           ;BR IF YES
005426  000137  005344          JMP     $GET42          ;CHECK FOR MONITOR RETURN
005432  012701  001530  3$:     MOV     #AVAIL,R1       ;ADDRESS OF 'AVAILABLE DRIVES' TABLE
005436  005711          4$:     TST     (R1)            ;SEE IF AT END OF TABLE
005440  001405                  BEQ     5$              ;BR IF AT END: GO CHECK 'WAIT' TABLE
005442  021115                  CMP     (R1),(R5)       ;IS DRIVE IN 'AVAIL' THE ONE TO BE DROPPED
005444  001414                  BEQ     7$              ;BR IF YES
005446  062701  000002          ADD     #2,R1           ;INCREMENT 'AVAIL' TABLE ADDRESS
005452  000771                  BR      4$              ;CONTINUE LOOKING
005454  012701  001552  5$:     MOV     #WAIT,R1        ;MOVE THE ADDRESS OF THE BUFFER WAIT TABLE
005460  005711          6$:     TST     (R1)            ;AT THE END OF THE 'WAIT' TABLE ?
005462  001752                  BEQ     2$              ;BR IF YES: SEE IF ANY MORE 'DROP' REQUESTS
005464  021115                  CMP     (R1),(R5)       ;DRIVE IN THE 'WAIT' TABLE ?
005466  001403                  BEQ     7$              ;BR IF IT IS
005470  062701  000002          ADD     #2,R1           ;INCREMENT 'WAIT' TABLE ADDRESS
005474  000771                  BR      6$              ;CONTINUE LOOK THROUGH THE 'WAIT' TABLE
005476  011100  7$:     MOV     (R1),R0         ;PUT THE DRIVE'S BLOCK ADDRESS IN R0
005500  104401  052650          TYPE    ,DEASSG         ;TYPE 'DRIVE DEASSIGNED'
005504  004737  022676          JSR     PC,TYPEST       ;TYPE THE DRIVE'S PERFORMANCE SUMMARY
005510  005015                  CLR     (R5)            ;CLEAR THE 'DROP DRIVE' TABLE ENTRY
005512  005011                  CLR     (R1)            ;REMOVE THE DRIVE FROM THE 'AVAIL' OR 'WAIT' TABLE
005514  004737  017530          JSR     PC,CMPRES       ;COMPRESS THE RESPECTIVE TABLE
005520  000733                  BR      2$              ;SEE IF ANY MORE DRIVES

                                ;LOOK FOR DRIVES TO BE ASSIGNED

005522  012703  000010  MAIN1:  MOV     #8.,R3          ;DRIVE COUNT
005526  005002                  CLR     R2              ;'AVAIL' INDEX
005530  005004                  CLR     R4              ;ASSIGN LIST INDEX
005532  005005                  CLR     R5              ;NEW DRIVE INDEX
005534  005765  001506  1$:     TST     NEWUNT(R5)      ;NEW DRIVE IN THIS POSITION
005540  001006                  BNE     3$              ;BR IF THERE IS
005542  062705  000002  2$:     ADD     #2,R5           ;INCREMENT R5
005546  005204                  INC     R4              ;INCREMENT ASSIGN INDEX
005550  005303                  DEC     R3              ;DECREMENT DRIVE COUNT
005552  001370                  BNE     1$              ;BR IF MORE DRIVES
005554  000432                  BR      MAIN2           ;START OPERATIONS FOR THE AVAILABLE DRIVES
005556  104401  052160  3$:     TYPE    ,UNTMSG         ;'DRIVE'
005562  010446                  MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
                                                        ;;TYPE DRIVE NUMBER
005564  104403                  TYPOS                   ;;GO TYPE--OCTAL ASCII
005566     002                  .BYTE   2               ;;TYPE 2 DIGIT(S)
005567     000                  .BYTE   0               ;;SUPPRESS LEADING ZEROS
```

```
3242  005570  104401  052720                    TYPE    ,ASGND          ;'ASSIGNED'
3243  005574  005762  001530            4$:     TST     AVAIL(R2)       ;AT END OF AVAILABLE TABLE
3244  005600  001403                            BEQ     5$              ;BR IF YES
3245  005602  062702  000002                    ADD     #2,R2           ;INCREMENT AVAILABLE TABLE INDEX
3246  005606  000772                            BR      4$              ;CONTINUE LOOKING FOR END OF TABLE
3247  005610  016562  001506  001530  5$:       MOV     NEWUNT(R5),AVAIL(R2)  ;MOVE ADDR OF DRIVE INTO AVAIL LST
3248  005616  005065  001506                    CLR     NEWUNT(R5)      ;TAKE DRIVE OUT OF NEW DRIVE TABLE
3249  005622  156437  033320  001462            BISB    ATABIT(R4),ASNLST ;SET DRIVE ASSIGNED INDICATOR
3250  005630  016200  001530                    MOV     AVAIL(R2),R0    ;PUT STARTING ADDRESS OF BLOCK IN R0
3251  005634  062702  000002                    ADD     #2,R2           ;INCREMENT AVAILABLE TABLE POINTER
3252  005640  000740                            BR      2$              ;LOOK FOR MORE DRIVES
3253
3254                                    ;GET PARAMETERS, BUFFER SPACE, AND START ORDERS FOR DRIVES IN
3255                                    ;       THE 'AVAILABLE' QUEUE
3256
3257  005642  005737  001552            MAIN2:  TST     WAIT            ;OUTSTANDING BUFFER REQUESTS
3258  005646  001113                            BNE     MAIN3           ;BR IF THERE ARE
3259  005650  005002                            CLR     R2              ;CLEAR DRIVE TABLE POINTER
3260  005652  005762  001530            1$:     TST     AVAIL(R2)       ;ANY DRIVES WAITING FOR PARAMETERS
3261  005656  001551                            BEQ     IDLE            ;BRANCH IF NONE
3262  005660  016200  001530                    MOV     AVAIL(R2),R0    ;CONTROL BLOCK ADDR IN R0
3263  005664  005760  000104                    TST     $NEXT(R0)       ;PARAMETERS BEEN SELECTED ?
3264  005670  001021                            BNE     6$              ;BR IF THEY HAVE
3265  005672  105760  000026                    TSTB    $PACK(R0)       ;'R' OR 'W' COMMAND FOR THE DRIVE ?
3266  005676  001403                            BEQ     2$              ;BR IF NOT
3267  005700  004737  017546                    JSR     PC,WRTPK        ;GET DATA PACK PARAMETERS
3268  005704  000415                            BR      7$              ;GET THE BUFFER
3269  005706  012701  001574            2$:     MOV     #PARQ,R1        ;ADDRESS OF THE PARAMETER QUEUE
3270  005712  020011                    3$:     CMP     R0,(R1)         ;IS CURRENT DRIVE IN THE QUEUE ?
3271  005714  001403                            BEQ     4$              ;BR IF IT IS
3272  005716  005721                            TST     (R1)+           ;AT END OF THE QUEUE
3273  005720  001403                            BEQ     5$              ;BR IF AT END
3274  005722  000773                            BR      3$              ;CONTINUE LOOKING
3275  005724  004737  017530            4$:     JSR     PC,CMPRES       ;COMPRESS THE TABLE
3276  005730  004737  016446            5$:     JSR     PC,SELPAR       ;SELECT THE PARAMETERS
3277  005734  004737  017322            6$:     JSR     PC,GETPAR       ;LOAD NEW PARAMETERS
3278  005740  005046                    7$:     CLR     -(SP)           ;MAKE ROOM ON THE STACK FOR THE BUFFER ADDR
3279  005742  004737  015672                    JSR     PC,GETBUF       ;GET BUFFER
3280  005746  012660  000006                    MOV     (SP)+,$BUF(R0)  ;MOVE BUFFER ADDR TO DPB
3281  005752  001424                            BEQ     8$              ;BR IF '0' ADDR (NO BUFFER)
3282  005754  004737  016242                    JSR     PC,FILBUF       ;FILL THE BUFFER
3283  005760  005060  000072                    CLR     $FAIR(R0)       ;CLEAR THE 'FAIRNESS' COUNT
3284  005764  004737  016370                    JSR     PC,GODRIV       ;PUT CURRENT DPB IN DRIVER
3285  005770  012705  001440                    MOV     #ORDERQ,R5      ;ADDRESS OF ORDER QUEUE IN R5
3286  005774  005725                            TST     (R5)+           ;END OF QUEUE ?
3287  005776  001376                            BNE     .-2             ;BR IF NOT
3288  006000  010045                            MOV     R0,-(R5)        ;PUT BLOCK ADDRESS INTO QUEUE
3289  006002  105760  000026                    TSTB    $PACK(R0)       ;'R' OR 'W' COMMAND FOR DRIVE ?
3290  006006  001025                            BNE     10$             ;BR IF EITHER
3291  006010  012705  001574                    MOV     #PARQ,R5        ;PUT BLOCK INTO THE PARAMETER QUEUE
3292  006014  005725                            TST     (R5)+           ;FIND THE END OF THE QUEUE
3293  006016  001376                            BNE     .-2             ;BR IF NOT AT END OF QUEUE
3294  006020  010045                            MOV     R0,-(R5)        ;PUT BLOCK ADDRESS INTO THE QUEUE
3295  006022  000417                            BR      10$             ;CONTINUE LOOKING
3296  006024  026037  000072  001254  8$:       CMP     $FAIR(R0),FAIRNS  ;ENTRY BEEN IN THE QUEUE LONG ENOUGH ?
3297  006032  001405                            BEQ     9$              ;BR IF YES
```

```
 3298  006034  005260  000072                 INC     $FAIR(R0)          ;INCREMENT THE ENTRY COUNT
 3299  006040  062702  000002                 ADD     #2,R2              ;INCREMENT THE POINTER
 3300  006044  000702                         BR      1$                 ;LOOK FOR SOME MORE DRIVES
 3301  006046  012705  001552          9$:    MOV     #WAIT,R5           ;'WAIT' QUEUE ADDRESS
 3302  006052  005725                         TST     (R5)+              ;LOOK FOR AN OPENING
 3303  006054  001376                         BNE     .-2                ;BR IF NONE YET
 3304  006056  016245  001530                 MOV     AVAIL(R2),-(R5)    ;MOVE DRIVE'S BLOCK ADDRESS TO QUEUE
 3305  006062  012701  001530          10$:   MOV     #AVAIL,R1          ;'AVAILABLE' TABLE ADDRESS
 3306  006066  060201                         ADD     R2,R1              ;FORM ADDRESS OF LAST ENTRY
 3307  006070  004737  017530                 JSR     PC,CMPRES          ;COMPRESS THE TABLE
 3308  006074  000666                         BR      1$                 ;CONTINUE LOOKING
 3309
 3310                                  ;GET BUFFER ASSIGNMENTS FOR DRIVES IN THE 'BUFFER WAIT' QUEUE
 3311
 3312  006076  013700  001552          MAIN3: MOV     WAIT,R0            ;MOVE THE 'WAIT' ENTRY TO R0
 3313  006102  005046                         CLR     -(SP)              ;MAKE ROOM ON THE STACK FOR THE BUFFER ADDR
 3314  006104  004737  015672                 JSR     PC,GETBUF          ;TRY TO GET A BUFFER
 3315  006110  012660  000006                 MOV     (SP)+,$BUF(R0)     ;MOVE THE BUFFER ADDR TO THE DPB
 3316  006114  001002                         BNE     1$                 ;BR IF A BUFFER WAS ASSIGNED
 3317  006116  000137  006202                 JMP     IDLE               ;NO BUFFER AVAILABLE YET
 3318  006122  004737  016242          1$:    JSR     PC,FILBUF          ;FILL THE BUFFER
 3319  006126  004737  016370                 JSR     PC,GODRIV          ;PUT THE ENTRY IN THE DRIVER
 3320  006132  005060  000072                 CLR     $FAIR(R0)          ;CLEAR THE 'FAIRNESS' COUNT
 3321  006136  012705  001440                 MOV     #ORDERQ,R5         ;ADDRESS OF ORDER QUEUE IN R5
 3322  006142  005725                         TST     (R5)+              ;AT END OF THE QUEUE
 3323  006144  001376                         BNE     .-2                ;BR IF NOT
 3324  006146  010045                         MOV     R0,-(R5)           ;PUT BLOCK ADDRESS IN QUEUE
 3325  006150  105760  000026                 TSTB    $PACK(R0)          ;'R' OR 'W' COMMAND FOR DRIVE ?
 3326  006154  001005                         BNE     2$                 ;BR IF YES, DON'T PUT BLOCK INTO 'PARQ'
 3327  006156  012705  001574                 MOV     #PARQ,R5           ;FIND THE END OF THE PARAMETER QUEUE
 3328  006162  005725                         TST     (R5)+              ;OPEN SLOT IN THE QUEUE ?
 3329  006164  001376                         BNE     .-2                ;BR IF NOT
 3330  006166  010045                         MOV     R0,-(R5)           ;PUT BLOCK ADDRESS INTO THE QUEUE
 3331  006170  012701  001552          2$:    MOV     #WAIT,R1           ;ADDRESS OF TABLE TO TO COMPRESS
 3332  006174  004737  017530                 JSR     PC,CMPRES          ;COMPRESS THE WAIT TABLE
 3333  006200  000620                         BR      MAIN2              ;LOOK FOR MORE ENTRIES
 3334
 3335                                  ;WAIT FOR AN ORDER TO FINISH
 3336
 3337  006202  012701  001440          IDLE:  MOV     #ORDERQ,R1         ;ADDRESS OF THE ORDER QUEUE IN R1
 3338  006206  012100                 1$:    MOV     (R1)+,R0           ;PUT BLOCK ADDRESS INTO R0
 3339  006210  001433                         BEQ     IDLE1              ;BR IF END OF QUEUE
 3340  006212  005760  000016                 TST     $STATUS(R0)        ;SEE IF DRIVE FINISHED
 3341  006216  001773                         BEQ     1$                 ;BR IF DRIVE NOT FINISHED
 3342  006220  162701  000002                 SUB     #2,R1              ;CORRECT THE QUEUE POINTER
 3343  006224  010146                         MOV     R1,-(SP)           ;SAVE THE QUEUE ADDRESS
 3344  006226  004737  015526                 JSR     PC,STATIS          ;ACCUMULATE STATISTICS FOR DRIVE IN R0
 3345  006232  000240                         NOP                        ;DEBUGGING AID
 3346  006234  004737  006540                 JSR     PC,PROCES          ;PROCESS END OF ORDER
 3347  006240  005037  001264                 CLR     BADSEC             ;CLEAR THE BAD TRK/SEC ERROR INDICATOR
 3348  006244  004737  026524                 JSR     PC,ABNRML          ;SEE IF ANY DRIVES HAVE TOO MANY ERRORS
 3349  006250  004737  026552                 JSR     PC,EOP             ;SEE IF ANY DRIVE HAS XFERED 3X10↑9 BITS
 3350  006254  012601                         MOV     (SP)+,R1           ;RESTORE THE ORDER TABLE INDEX
 3351  006256  012705  001530                 MOV     #AVAIL,R5          ;FIND THE END OF THE 'AVAILABLE' TABLE
 3352  006262  005725                 2$:    TST     (R5)+              ;END OF THE TABLE ?
 3353  006264  001376                         BNE     2$                 ;BR IF NOT AT END OF LIST
```

```
3354  006266  011145                        MOV     (R1),-(R5)      ;MOVE THE BLOCK ADDRESS INTO THE TABLE
3355  006270  004737  017530                JSR     PC,CMPRES       ;COMPRESS THE ORDER QUEUE
3356  006274  004737  016026                JSR     PC,RELBUF       ;RESTORE BUFFER
3357  006300  005737  001262        IDLE1:  TST     CFLAG           ;'CONTROL C' FLAG ENTERED ?
3358  006304  001403                        BEQ     1$              ;BR IF IT WAS
3359  006306  004737  023710                JSR     PC,KSR          ;SERVICE THE KEYBOARD
3360  006312  000733                        BR      IDLE            ;SYSTEM WAS BUSY
3361  006314  032777  000004  172616  1$:   BIT     #SW02,@SWR      ;TYPE PERFORMANCE SUMMARY
3362  006322  001007                        BNE     2$              ;BR IF NOT
3363  006324  005737  001214                TST     STATIN          ;TIME TO TYPE THE PERFORMANCE SUMMARY ?
3364  006330  001404                        BEQ     2$              ;BR IF NOT
3365  006332  005037  001214                CLR     STATIN          ;CLEAR THE INDICATOR
3366  006336  004737  022614                JSR     PC,STATPR       ;TYPE THE SUMMARY
3367  006342  005737  001574        2$:     TST     PARQ            ;ENTRY IN THE PARAMETER QUEUE ?
3368  006346  001410                        BEQ     3$              ;BR IF NOT
3369  006350  013700  001574                MOV     PARQ,R0         ;PUT THE BLOCK ADDRESS INTO R0
3370  006354  004737  016446                JSR     PC,SELPAR       ;GET THE PARAMETERS FOR NEXT OPERATION
3371  006360  012701  001574                MOV     #PARQ,R1        ;SETUP TO COMPRESS THE TABLE
3372  006364  004737  017530                JSR     PC,CMPRES       ;COMPRESS THE PARAMETER QUEUE
3373  006370  000137  005374        3$:     JMP     MAIN            ;CONTINUE THE LOOP
3374
3375                                ;SETUP TO REFORMAT AN ERROR SECTOR
3376
3377  006374  032777  000001  172536  REFMT:  BIT     #SW0,@SWR       ;READ ONLY SWITCH SET ?
3378  006402  001055                        BNE     REFMTX          ;BR IF IT IS
3379  006404  032777  000200  172526        BIT     #SW7,@SWR       ;SWITCH 7 SET ?
3380  006412  001051                        BNE     REFMTX          ;BR IF IT IS
3381  006414  005737  001416                TST     FORMAT          ;WRITE HEADER & DATA ORDERS ALLOWED ?
3382  006420  001446                        BEQ     REFMTX          ;BR IF NOT
3383  006422  016060  000272  000100        MOV     $RPCC(R0),$NCYL(R0)  ;USE PRESENT CYLINDER
3384  006430  004737  022314                JSR     PC,READDR       ;GET CORRECTED SECTOR-TRACK ADDRESSES
3385  006434  112660  000077                MOVB    (SP)+,$NTRK(R0) ;TRACK ADDR TO DPB
3386  006440  112660  000076                MOVB    (SP)+,$NSEC(R0) ;SECTOR ADDR TO DPB
3387  006444  012760  000404  000102        MOV     #260.,$NWRDL(R0)  ;WORD COUNT FOR FORMAT
3388  006452  023727  001404  000404        CMP     MAXDL,#260.     ;CAN A FULL SECTOR BE WRITTEN ?
3389  006460  103003                        BHIS    1$              ;BR IF IT CAN
3390  006462  013760  001404  000102        MOV     MAXDL,$NWRDL(R0)  ;PUT TRANSFER SIZE INTO THE DPB
3391  006470  112760  000003  000074  1$:   MOVB    #3,$NCODE(R0)   ;COMMAND CODE
3392  006476  004737  017274                JSR     PC,GETPAT       ;GET A PATTERN
3393  006502  110560  000075                MOVB    R5,$NPATC(R0)   ;PATTERN CODE TO CONTROL BLOCK
3394  006506  012760  177777  000104        MOV     #-1,$NEXT(R0)   ;SET PARAMETERS SELECTED INDICATOR
3395  006514  012701  001574                MOV     #PARQ,R1        ;SET UP TO SEE IF BLOCK IN THE PARAMETER QUEUE
3396  006520  005711                2$:     TST     (R1)            ;SEE IF AT END OF TABLE
3397  006522  001405                        BEQ     REFMTX          ;BR IF AT END
3398  006524  020021                        CMP     R0,(R1)+        ;SEE IF BLOCK AT PRESENT POSITION
3399  006526  001374                        BNE     2$              ;BR IF NOT
3400  006530  005041                        CLR     -(R1)           ;CLEAR THE ENTRY
3401  006532  004737  017530                JSR     PC,CMPRES       ;COMPRESS THE TABLE
3402  006536  000207                REFMTX: RTS     PC              ;RETURN
3403
3404                                ;PROCESS THE ORDER TERMINATION
3405
```

```
3406  006540  111037  001246          PROCES: MOVB    (R0),UNIT       ;DRIVE NUMBER FOR ANY ERROR MESSAGES
3407  006544  005760  000016                  TST     $STATUS(R0)     ;SEE IF DRIVER SIGNALED AN ERROR
3408  006550  100427                          BMI     ERPROC          ;BR IF ERROR
3409  006552  032760  100000  000234          BIT     #BIT15,$RPCS1(R0) ;SEE IF 'SC' SET
3410  006560  001410                          BEQ     1$              ;BR IF NOT SET
3411  006562  032760  040000  000234          BIT     #BIT14,$RPCS1(R0) ;SEE IF 'TRE' SET
3412  006570  001017                          BNE     ERPROC          ;BR IF SET
3413  006572  032760  040000  000246          BIT     #BIT14,$RPDS1(R0) ;SEE IF 'ERR' SET
3414  006600  001013                          BNE     ERPROC          ;BR IF SET
3415  006602  004737  012770          1$:     JSR     PC,CKERR        ;NO ERROR, CHECK ERROR BITS ANYWAY
3416  006606  004737  013070                  JSR     PC,CKBUS        ;NO ERROR, CHECK BUS ADDR & WC
3417  006612  032777  000002  172320          BIT     #SWD1,@SWR      ;DATA COMPARE ?
3418  006620  001002                          BNE     2$              ;BR IF NOT
3419  006622  004737  013154                  JSR     PC,CMPAR        ;NO ERROR, COMPARE DATA
3420  006626  000207                  2$:     RTS     PC              ;RETURN
3421
3422                                          ;ORDER TERMINATED WITH AN ERROR - PROCESS THE ERROR
3423
3424  006630  032760  000200  000016  ERPROC: BIT     #BIT07,$TATUS(R0) ;DONE BIT SET ?
3425  006636  001402                          BEQ     ERPRC1          ;BR IF ORDER DIDN'T COMPLETE NORMALLY
3426  006640  000137  007264                  JMP     DONE            ;PROCESS ERROR WITH 'DONE' BIT SET
3427
3428                                          ;PROCESS ORDER COMPLETION WITH 'ERROR' & 'DONE NOT' BITS
3429
3430  006644  032760  010000  000016  ERPRC1: BIT     #BIT12,$TATUS(R0) ;SEE IF DRIVE WAS UNSAFE
3431  006652  001025                          BNE     PUNSAF          ;BR IF YES
3432  006654  032760  004000  000016          BIT     #BIT11,$TATUS(R0) ;PARITY ERROR OCCURRED
3433  006662  001055                          BNE     UCPAR           ;BR IF IT DID
3434  006664  032760  002000  000016          BIT     #BIT10,$TATUS(R0) ;FATAL PARITY ERROR?
3435  006672  001056                          BNE     FALPAR          ;BR IF THERE IS ONE
3436  006674  032760  001000  000016          BIT     #BIT09,$TATUS(R0) ;TIMEOUT?
3437  006702  001076                          BNE     SWTIM           ;BR IF YES
3438  006704  032760  040002  000016          BIT     #BIT14!BIT01,$TATUS(R0) ;DRIVE WENT OFFLINE ?
3439  006712  001111                          BNE     OFLIN           ;BR IF IT DID
3440  006714  032760  000004  000016          BIT     #BIT2,$TATUS(R0) ;PORT REQUEST TIME OUT ?
3441  006722  001141                          BNE     PRTIM           ;BR IF IT DID
3442  006724  000207                          RTS     PC              ;ERROR. RETURN
3443
3444                                      ;DRIVE IS PERSISTENTLY UNSAFE
3445
3446  006726  104401  001165          PUNSAF: TYPE    ,$CRLF          ;CR-LF
3447  006732  104401  045410                  TYPE    ,EM12           ;'DRIVE UNSAFE' MESSAGE
3448  006736  104401  052673                  TYPE    ,DRNUM          ;DRIVE NUMBER
3449  006742  013746  001246                  MOV     UNIT,-(SP)      ;;SAVE UNIT FOR TYPEOUT
3450                                                                  ;;TYPE DRIVE NUMBER
3451  006746  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
3452  006750     002                          .BYTE   2               ;;TYPE 2 DIGIT(S)
3453  006751     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
3454  006752  104401  001165                  TYPE    ,$CRLF          ;CR-LF
3455  006756  004737  020200                  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3456  006762  104414  045410                  DISPLY  ,EM12           ;PERSISTENT DEVICE UNSAFE MESSAGE
3457  006766  004737  020244                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
3458  006772  004737  020652                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
3459  006776  004737  021326                  JSR     PC,LINE4        ;PRINT LINE 4 OF THE ERROR MESSAGE
3460  007002  004737  023424                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3461  007006  004737  021762                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
```

```
3462  007012  000137  026446                  JMP     DROP            ;DROP THE DRIVE
3463
3464
3465
3466  007016  104401  001165          UCPAR:  TYPE    ,$CRLF          ;CR-LF
3467  007022  104401  045312                  TYPE    ,EM10           ;'UNCORRECTABLE PARITY ERROR' MESSAGE
3468  007026  000404                          BR      FALPR1          ;FINISH PROCESSING THE ERROR
3469
3470
3471
3472  007030  104401  001165          FALPAR: TYPE    ,$CRLF          ;CR-LF
3473  007034  104401  045355                  TYPE    ,EM11           ;'FATAL PARITY ERROR' MESSAGE
3474  007040  104401  052673          FALPR1: TYPE    ,DRNUM          ;DRIVE NUMBER
3475  007044  013746  001246                  MOV     UNIT,-(SP)      ;;SAVE UNIT FOR TYPEOUT
3476                                                                  ;;TYPE DRIVE NUMBER
3477  007050  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
3478  007052     002                          .BYTE   2               ;;TYPE 2 DIGIT(S)
3479  007053     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
3480  007054  104401  001165                  TYPE    ,$CRLF          ;CR-LF
3481  007060  004737  023424                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3482  007064  032777  100000  172046        BIT     #SW15,@SWR      ;HALT ON ERROR ?
3483  007072  001401                          BEQ     1$              ;BR IF NOT
3484  007074  000000                          HALT                    ;ERROR HALT
3485  007076  000207                  1$:     RTS     PC
3486
3487
3488
3489  007100  004737  020200          SWTIM:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3490  007104  104414  045441                  DISPLY  ,EM13           ;PRINT THE TIME OUT MESSAGE
3491  007110  004737  020244                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
3492  007114  004737  020652                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
3493  007120  004737  021326                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
3494  007124  004737  023424                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3495  007130  004737  021762                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3496  007134  000207                          RTS     PC              ;RETURN
3497
3498
3499
3500  007136  104401  001165          OFLIN:  TYPE    ,$CRLF          ;CR-LF
3501  007142  104401  045513                  TYPE    ,EM14           ;'DRIVE WENT OFFLINE' MESSAGE
3502  007146  104401  052673                  TYPE    ,DRNUM          ;DRIVE NUMBER
3503  007152  013746  001246                  MOV     UNIT,-(SP)      ;;SAVE UNIT FOR TYPEOUT
3504                                                                  ;;TYPE DRIVE NUMBER
3505  007156  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
3506  007160     002                          .BYTE   2               ;;TYPE 2 DIGIT(S)
3507  007161     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
3508  007162  104401  001165                  TYPE    ,$CRLF          ;CR-LF
3509  007166  004737  020200                  JSR     PC,LINE1        ;PRINT LINE 1 OF THE ERROR MESSAGE
3510  007172  104414  045513                  DISPLY  ,EM14           ;PRINT OFFLINE MESSAGE
3511  007176  004737  020244                  JSR     PC,LINE2        ;PRINT LINE 2 OF THE ERROR MESSAGE
3512  007202  004737  020652                  JSR     PC,LINE3        ;PRINT LINE 3 OF THE ERROR MESSAGE
3513  007206  004737  021326                  JSR     PC,LINE4        ;PRINT LINE 4 OF THE ERROR MESSAGE
3514  007212  004737  023424                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3515  007216  004737  021762                  JSR     PC,LINE7        ;PRINT LINE 7 OF THE ERROR MESSAGE
3516  007222  000137  026446                  JMP     DROP            ;DROP THE DRIVE
3517
```

```
                                        ;PORT REQUEST TIMEOUT ERROR
                                        ;
007226  004737  020200        PRTIM:  JSR     PC,LINE1        ;TYPE LINE 1 OF THE ERROR MESSAGE
007232  104414  045536                DISPLY  EM15            ;PRINT PORT TIME OUT MESSAGE
007236  004737  020244                JSR     PC,LINE2        ;TYPE LINE 2 OF THE ERROR MESSAGE
007242  004737  020652                JSR     PC,LINE3        ;TYPE LINE 3 OF THE ERROR MESSAGE
007246  004737  021326                JSR     PC,LINE4        ;TYPE LINE 4 OF THE ERROR MESSAGE
007252  004737  023424                JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
007256  004737  021762                JSR     PC,LINE7        ;TYPE LINE 7 OF THE ERROR MESSAGE
007262  000207                      RTS     PC              ;RETURN
                                        ;
                                        ;PROCESS ORDER COMPLETION WITH 'ERROR' & 'DONE' BITS SET
                                        ;
007264  032760  000030  000016  DONE:  BIT     #BIT04!BIT03,STATUS(R0) ;UNSAFE OCCURRED
007272  001402                        BEQ     .+6             ;BR IF NOT
007274  000137  012466                JMP     UNSAF           ;REPORT UNSAFE
007300  032760  040000  000244        BIT     #BIT14,$RPCS2(R0)  ;IS 'WCE' SET ?
007306  001006                        BNE     1$              ;BR IF SET
007310  032760  040000  000246        BIT     #BIT14,$RPDS1(R0) ;CHECK 'ERR'
007316  001002                        BNE     1$              ;BR IF SET
007320  000137  012232                JMP     TRFER           ;PROCESS 'TRE'
007324  032760  000400  000250  1$:   BIT     #BIT08,$RPER1(R0) ;'HCRC' SET?
007332  001402                        BEQ     .+6             ;BR IF NOT
007334  000137  010710                JMP     HCRCER          ;PROCESS 'HCRC'
007340  032760  000020  000250        BIT     #BIT04,$RPER1(R0) ;'FMT' SET?
007346  001402                        BEQ     .+6             ;BR IF NOT SET
007350  000137  011072                JMP     CKFMT           ;CHECK FORMAT ERROR
007354  032760  000200  000250        BIT     #BIT07,$RPER1(R0) ;'HCE' SET?
007362  001402                        BEQ     .+6             ;BR IF NOT SET
007364  000137  011266                JMP     CKHCE           ;CHECK 'HCE' ERROR
007370  032760  020000  000250        BIT     #BIT13,$RPER1(R0) ;'OPI' SET?
007376  001402                        BEQ     .+6             ;BR IF NOT SET
007400  000137  011566                JMP     OPIER           ;REPORT 'OPI'
007404  032760  000010  000250        BIT     #BIT3,$RPER1(R0) ;'PAR' SET?
007412  001402                        BEQ     .+6             ;BR IF NOT SET
007414  000137  011720                JMP     PARER           ;REPORT 'PAR'
007420  032760  000040  000250        BIT     #BIT5,$RPER1(R0) ;'WCF' SET?
007426  001402                        BEQ     .+6             ;BR IF NOT SET
007430  000137  012370                JMP     WCFER           ;REPORT 'WCF'
007434  032760  002000  000250        BIT     #BIT10,$RPER1(R0) ;'IAE' SET?
007442  001402                        BEQ     .+6             ;BR IF NOT SET
007444  000137  012012                JMP     IAEER           ;REPORT 'IAE'
007450  032760  004000  000250        BIT     #BIT11,$RPER1(R0) ;'WLE' SET?
007456  001402                        BEQ     .+6             ;BR IF NOT SET
007460  000137  012044                JMP     WLEER           ;REPORT 'WLE'
007464  032760  001000  000250        BIT     #BIT9,$RPER1(R0) ;'AOE' SET?
007472  001405                        BEQ     2$              ;BR IF NOT SET
007474  032760  002000  000246        BIT     #BIT10,$RPDS1(R0) ;'LST' SET?
007502  001401                        BEQ     2$              ;BR IF NOT SET
007504  000207                      RTS     PC              ;'AOE' & 'LST' SET, EXIT
007506  032760  010000  000250  2$:   BIT     #BIT12,$RPER1(R0)  ;SEE IF 'DTE' SET
007514  001402                        BEQ     .+6             ;BR IF NOT
007516  000137  011676                JMP     DTEER           ;REPORT 'DTE' ERROR
007522  032760  040000  000244        BIT     #BIT14,$RPCS2(R0)  ;SEE IF 'WCK' SET
007530  001402                        BEQ     .+6             ;BR IF NOT SET
007532  000137  010362                JMP     WCKER           ;REPORT 'WCK'
```

```
3574  007536  005760  000250                  TST     $RPER1(R0)          ;SEE IF 'DCK' SET
3575  007542  100002                          BPL     .+6                 ;BR IF NOT
3576  007544  000137  007570                  JMP     DCKER               ;PROCESS 'DCK'
3577  007550  032760  140000  000276          BIT     #BIT15!BIT14,$RPER3(R0) ;'SKI' OR 'OCYL' SET
3578  007556  001402                          BEQ     .+6                 ;BR IF NOT SET
3579  007560  000137  012332                  JMP     SKIER               ;REPORT ERROR
3580  007564  000137  011040                  JMP     DRVER               ;REPORT DRIVE ERROR
3581
3582                                  ;PROCESS DATA ('DCK') CHECK ERROR
3583
3584  007570  022760  010042  000300  DCKER:  CMP     #10042,$RPEC1(R0)   ;VALID POSITION COUNT ?
3585  007576  101406                          BLOS    1$                  ;BR IF NOT VALID
3586  007600  005760  000300                  TST     $RPEC1(R0)          ;POSITION COUNT 0 ?
3587  007604  001403                          BEQ     1$                  ;BR IF 0'S
3588  007606  005760  000302                  TST     $RPEC2(R0)          ;VALUE IN PATTERN REGISTER ?
3589  007612  001026                          BNE     4$                  ;BR IF YES
3590  007614  004737  020200          1$:     JSR     PC,LINE1            ;TYPE FIRST LINE OF ERROR MESSAGE
3591  007620  104414  047143                  DISPLY  EM45                ;TYPE 'ECC LOGIC ERROR'
3592  007624  004737  020244                  JSR     PC,LINE2            ;TYPE LINE 2 OF ERROR MESSAGE
3593  007630  004737  023424                  JSR     PC,INCTOT           ;INCREMENT TOTAL ERROR COUNT
3594  007634  012737  000003  001252          MOV     #3,RETRY            ;RETRY COUNT
3595  007642  004737  015400                  JSR     PC,$RETRY           ;RETRY THE ORDER
3596  007646  000403                          BR      2$                  ;RETRY WAS NOT SUCCESSFUL
3597  007650  004737  021700                  JSR     PC,LINE6C           ;TYPE LINE 6C OF ERROR MESSAGE
3598  007654  000402                          BR      3$                  ;FINISH THE ERROR REPORT
3599  007656  004737  021706          2$:     JSR     PC,LINE6D           ;TYPE LINE 6D OF ERROR MESSAGE
3600  007662  004737  021762          3$:     JSR     PC,LINE7            ;TYPE LINE 7 OF ERROR MESSAGE
3601  007666  000402                          BR      5$                  ;EXIT
3602  007670  004737  020042          4$:     JSR     PC,SPOTCK           ;SEE IF ERROR AT A BAD SPOT ON THE PACK
3603  007674  000207                  5$:     RTS     PC                  ;IT IS, DON'T REPORT IT
3604  007676  126027  000024  000001          CMPB    $CODE(R0),#1        ;IS ORDER A WRITE CHECK ?
3605  007704  101002                          BHI     6$                  ;BR IF NOT
3606  007706  000137  010362                  JMP     WCKER               ;REPORT ERROR UNDER WRITE CHECK PROCESSING
3607  007712  004737  020200          6$:     JSR     PC,LINE1            ;PRINT LINE 1 OF ERROR MESSAGE
3608  007716  104414  045613                  DISPLY  EM21                ;DATA CHECK ERROR
3609  007722  004737  020244          DCKER1: JSR     PC,LINE2            ;PRINT LINE 2 OF ERROR MESSAGE
3610  007726  004737  020652                  JSR     PC,LINE3            ;PRINT LINE 3 OF ERROR MESSAGE
3611  007732  004737  021326                  JSR     PC,LINE4            ;PRINT LINE 4 OF ERROR MESSAGE
3612  007736  004737  015042                  JSR     PC,PRTBAD           ;SEE IF BAD SECTOR TO BE PRINTED
3613  007742  012737  110100  001250          MOV     #BIT15!BIT12!BIT06,MASK  ;LOAD ERROR MASK
3614  007750  032760  010100  000250          BIT     #BIT12!BIT06,$RPER1(R0)  ;CHECK 'DTE' & 'ECH'
3615  007756  001003                          BNE     1$                  ;BR IF SET
3616  007760  004737  021652                  JSR     PC,LINE6            ;PRINT LINE 6 OF ERROR MESSAGE
3617  007764  000541                          BR      8$                  ;FINISH THE ERROR REPORT
3618  007766  012737  000020  001252  1$:     MOV     #16.,RETRY          ;RETRY COUNT
3619  007774  005001                          CLR     R1                  ;R1 IS OFFSET CODE POINTER
3620  007776  032760  000002  000262          BIT     #BIT01,$RPDT(R0)    ;IS DRIVE AN RP06 ?
3621  010004  001002                          BNE     16$                 ;BR IF IT IS
3622  010006  012701  000007                  MOV     #7,R1               ;INCREMENT PAST RP06 OFFSET CODES
3623  010012  004737  016026          16$:    JSR     PC,RELBUF           ;RELEASE THE BUFFER
3624  010016  004737  022314                  JSR     PC,READDR           ;GET THE ADDRESS OF THE ERROR SECTOR
3625  010022  112660  000011                  MOVB    (SP)+,$TRK(R0)      ;TRACK ADDRESS OF ERROR SECTOR
3626  010026  112660  000010                  MOVB    (SP)+,$SEC(R0)      ;SECTOR ADDRESS OF ERROR SECTOR
3627  010032  016060  000272  000012          MOV     $RPCC(R0),$CYL(R0)  ;PRESENT CYLINDER
3628  010040  026060  000022  000020          CMP     $SSEC(R0),$WRDL(R0)  ;SEE IF TRANSFER LENGTH LESS THAN 1 SECTOR
3629  010046  103010                          BHIS    15$                 ;BR IF IT IS; USE PRESENT TRANSFER LENGTH
```

```
3630  010050  016060  000022  000020          MOV     $SSEC(R0),$WRDL(R0)  ;CHANGE TRANSFER SIZE TO 1 SECTOR
3631  010056  016060  000020  000004          MOV     $WRDL(R0),$WRDM(R0)  ;SETUP WORD COUNT FOR OPERATION
3632  010064  005460  000004                  NEG     $WRDM(R0)            ;CHANGE COUNT TO 2'S COMP
3633  010070  005046                  15$:    CLR     -(SP)                ;SPACE FOR NEW BUFFER ADDRESS
3634  010072  004737  015672                  JSR     PC,GETBUF            ;GET A BUFFER
3635  010076  012660  000006                  MOV     (SP)+,$BUF(R0)       ;NEW BUFFER ADDRESS TO DPB
3636  010102  004737  016370          2$:     JSR     PC,GODRIV            ;RETRY
3637  010106  005760  000016          3$:     TST     $STATUS(R0)          ;TEST FOR DONE
3638  010112  001775                          BEQ     3$                   ;BR IF NOT DONE
3639  010114  100075                          BPL     10$                  ;BR IF NOT ERROR
3640  010116  032760  000200  000016          BIT     #BIT7,$STATUS(R0)    ;SEE IF ORDER TERMINIATED NORMALLY
3641  010124  001006                          BNE     14$                  ;BR IF NOT
3642  010126  004737  023424                  JSR     PC,INCTOT            ;INCREMENT TOTAL ERROR COUNT
3643  010132  104414  051330                  DISPLY  LIN8M                ;'DIFFERENT ERROR DURING RETRY'
3644  010136  000137  006644                  JMP     ERPRC1               ;SEE WHICH ERROR
3645  010142  033760  001250  000250  14$:     BIT     MASK,$RPER1(R0)      ;LOOK AT CURRENT ERROR
3646  010150  001430                          BEQ     5$                   ;BR IF DIFFERENT ERROR
3647  010152  032760  010100  000250          BIT     #BIT12!BIT6,$RPER1(R0)  ;'ECH' OR 'DTE' STILL SET ?
3648  010160  001437                          BEQ     7$                   ;BR IF NEITHER SET
3649  010162  105237  001253                  INCB    RETRY+1              ;INCREMENT RETRY COUNT
3650  010166  123737  001252  001253          CMPB    RETRY,RETRY+1        ;DONE ?
3651  010174  001342                          BNE     2$                   ;BR IF NOT
3652  010176  005201                          INC     R1                   ;INCREMENT TABLE INDEX
3653  010200  116137  002220  044731          MOVB    OFFCOD(R1),GENDPB+$FMT  ;OFFSET CODE
3654  010206  001435                          BEQ     9$                   ;BR IF END OF OFFSET TABLE
3655  010210  062737  000002  001252          ADD     #2,RETRY             ;NEW RETRY LIMIT
3656  010216  004737  015272                  JSR     PC,OFFST             ;OFFSET
3657  010222  005737  044746          4$:     TST     GENDPB+$STATUS       ;SEE IF FINISHED WITH OFFSET
3658  010226  001775                          BEQ     4$                   ;BR IF NOT
3659  010230  100324                          BPL     2$                   ;BR IF NO ERROR PERFORMING OFFSET
3660  010232  004737  022230          5$:     JSR     PC,LINE8             ;PRINT LINE 8 OF ERROR MESSAGE
3661  010236  004737  023330          6$:     JSR     PC,INCHRD            ;INCREMENT 'HARD' ERROR COUNT
3662  010242  004737  023424                  JSR     PC,INCTOT            ;INCREMENT TOTAL ERROR COUNT
3663  010246  004737  021762                  JSR     PC,LINE7             ;PRINT LINE 7 OF ERROR MESSAGE
3664  010252  004737  015042                  JSR     PC,PRTBAD            ;PRINT THE BAD SECTOR
3665  010256  000436                          BR      13$                  ;CLEAN UP AND RETURN
3666  010260  004737  021672          7$:     JSR     PC,LINE6B            ;PRINT LINE 6B OF ERROR MESSAGE
3667  010264  004737  021610                  JSR     PC,LINE5B            ;PRINT LINE 5B OF THE ERROR MESSAGE
3668  010270  004737  023304          8$:     JSR     PC,INCSOF            ;INCREMENT 'SOFT' ERROR COUNT
3669  010274  004737  014302                  JSR     PC,ECC               ;CORRECT THE ERROR USING ECC AND CHECK IT
3670  010300  000407                          BR      11$                  ;COMPARE THE BUFFER
3671  010302  004737  021706          9$:     JSR     PC,LINE6D            ;PRINT LINE 6D OF ERROR MESSAGE
3672  010306  000753                          BR      6$                   ;INCREMENT ERROR COUNT
3673  010310  004737  021664          10$:    JSR     PC,LINE6A            ;PRINT LINE 6A OF ERROR MESSAGE
3674  010314  004737  023304                  JSR     PC,INCSOF            ;INCREMENT 'SOFT' ERROR COUNT
3675  010320  012737  000001  001300  11$:     MOV     #1,FRSTER            ;SET PROCESSING 'DCKER' INDICATOR
3676  010326  004737  013172                  JSR     PC,CMPARD            ;COMPARE THE BUFFER
3677  010332  105737  001301                  TSTB    FRSTER+1             ;ERROR IN COMPARE ?
3678  010336  100406                          BMI     13$                  ;BRANCH IF ERROR
3679  010340  004737  023424                  JSR     PC,INCTOT            ;INCREMENT TOTAL ERROR COUNT
3680  010344  104414  051542                  DISPLY  LIN9G                ;'DATA COMPARE OK' MESSAGE
3681  010350  004737  021762          12$:    JSR     PC,LINE7             ;PRINT LINE 7 OF ERROR MESSAGE
3682  010354  004737  006374          13$:    JSR     PC,REFMT             ;REFORMAT THE ERROR SECTOR
3683  010360  000207                          RTS     PC                   ;RETURN
3684
3685                                  ;WRITE CHECK ERROR PROCESSING
```

```
3686
3687  010362  032760  100000  000250  WCKER:  BIT     #BIT15,$RPER1(R0)  ;SEE IF 'DCK' SET ALSO
3688  010370  001034                          BNE     2$                 ;BR IF IT IS
3689  010372  004737  020200                  JSR     PC,LINE1           ;PRINT LINE 1 OF ERROR MESSAGE
3690  010376  104414  045717                  DISPLY  ,EM23              ;PRINT WCE & DCK NOT
3691  010402  005037  001250                  CLR     MASK               ;CLEAR ERROR MASK
3692  010406  004737  020244                  JSR     PC,LINE2           ;PRINT LINE 2 OF ERROR MESSAGE
3693  010412  004737  020652                  JSR     PC,LINE3           ;PRINT LINE 3 OF ERROR MESSAGE
3694  010416  004737  021326                  JSR     PC,LINE4           ;PRINT LINE 4 OF ERROR MESSAGE
3695  010422  004737  021416                  JSR     PC,LINE5           ;PRINT LINE 5 OF ERROR MESSAGE
3696  010426  004737  023424                  JSR     PC,INCTOT          ;INCREMENT TOTAL ERROR COUNT
3697  010432  012737  000003  001252          MOV     #3,RETRY           ;RETRY LIMIT
3698  010440  004737  015400                  JSR     PC,$RETRY          ;RETRY THE OPERATION
3699  010444  000403                          BR      1$                 ;RETRY UNSUCESSFUL
3700  010446  004737  021700                  JSR     PC,LINE6C          ;PRINT LINE 6C OF ERROR MESSAGE
3701  010452  000502                          BR      8$                 ;FINISH PROCESSING THE ERROR
3702  010454  004737  021706          1$:     JSR     PC,LINE6D          ;PRINT LINE 6D OF ERROR MESSAGE
3703  010460  000506                          BR      10$                ;FINISH PROCESSING THE ERROR
3704  010462  004737  020042          2$:     JSR     PC,SPOTCK          ;SEE IF ERROR AT BAD SPOT ON THE PACK
3705  010466  000507                          BR      11$                ;EXIT IF AT BAD SPOT ON PACK
3706  010470  004737  020200                  JSR     PC,LINE1           ;PRINT LINE 1 OF ERROR MESSAGE
3707  010474  012737  045644  010522          MOV     #EM22,13$          ;ASSUME THAT EM22 WILL BE PRINTED
3708  010502  032760  040000  000244          BIT     #BIT14,$RPCS2(R0)  ;DID 'WCK' ALSO SET ?
3709  010510  001003                          BNE     12$                ;BR IF IT DID
3710  010512  012737  046545  010522          MOV     #EM37,13$          ;MESSAGE FOR 'DCK' AND 'WCK' NOT DURING
3711                                                                     ;WRITE CHECK
3712  010520  104414                  12$:    DISPLY                     ;TYPE THE ERROR MESSAGE
3713  010522  000000                  13$:    .WORD   0                  ;MESSAGE ADDRESS GOES HERE
3714  010524  004737  020244                  JSR     PC,LINE2           ;PRINT LINE 2 OF ERROR MESSAGE
3715  010530  004737  020652                  JSR     PC,LINE3           ;PRINT LINE 3 OF ERROR MESSAGE
3716  010534  004737  021326                  JSR     PC,LINE4           ;PRINT LINE 4 OF ERROR MESSAGE
3717  010540  004737  021416                  JSR     PC,LINE5           ;PRINT LINE 5 OF ERRCR MESSAGE
3718  010544  032760  000100  000250          BIT     #BIT06,$RPER1(R0)  ;ECH SET ALSO ?
3719  010552  001442                          BEQ     8$                 ;FINISH PROCESSING THE ERROR
3720  010554  012737  000020  001252  3$:     MOV     #16.,RETRY         ;RETRY LIMIT - 16 (10)
3721  010562  004737  016370          4$:     JSR     PC,GODRIV          ;RETRY THE ORDER
3722  010566  005760  000016          5$:     TST     $TATUS(R0)         ;ORDER FINISHED ?
3723  010572  001775                          BEQ     5$                 ;BR IF NOT
3724  010574  100405                          BMI     6$                 ;BR IF ERROR ON ORDER
3725  010576  105237  001253                  INCB    RETRY+1            ;INCREMENT RETRY COUNT
3726  010602  004737  021700                  JSR     PC,LINE6C          ;PRINT LINE 6C OF ERROR MESSAGE
3727  010606  000431                          BR      9$                 ;FINISH ERROR PROCESSING
3728  010610  105237  001253          6$:     INCB    RETRY+1            ;INCREMENT RETRY COUNT
3729  010614  123737  001252  001253          CMPB    RETRY,RETRY+1      ;DONE ?
3730  010622  001714                          BEQ     1$                 ;BR IF AT RETRY LIMIT
3731  010624  032760  100000  000250          BIT     #BIT15,$RPER1(R0)  ;'DCK' SET
3732  010632  001407                          BEQ     7$                 ;BR IF NOT - DIFFERENT ERROR
3733  010634  032760  000100  000250          BIT     #BIT06,$RPER1(R0)  ;'ECH' ALSO SET ?
3734  010642  001347                          BNE     4$                 ;BR IF IT IS, RETRY ORDER
3735  010644  004737  021700                  JSR     PC,LINE6C          ;PRINT LINE 6C OF ERROR MESSAGE
3736  010650  000403                          BR      8$                 ;FINISH PROCESSING ERROR
3737  010652  004737  022230          7$:     JSR     PC,LINE8           ;PRINT LINE 8 - 'DIFFERENT ERROR '
3738  010656  000405                          BR      9$                 ;FINISH PROCESSING ERROR
3739  010660  004737  023424          8$:     JSR     PC,INCTOT          ;INCREMENT TOTAL ERROR COUNT
3740  010664  004737  021762                  JSR     PC,LINE7           ;FINISH THE ERROR MESSAGE
3741  010670  000406                          BR      11$                ;EXIT
```

```
3742  010672  004737  023424          9$:     JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3743  010676  004737  021762          10$:    JSR     PC,LINE7        ;FINISH THE ERROR MESSAGE
3744  010702  004737  006374                  JSR     PC,REFMT        ;REFORMAT THE SECTOR IN ERROR
3745  010706  000207                  11$:    RTS     PC              ;RETURN
3746
3747                                  ;REPORT 'HCRC' ERROR
3748
3749  010710  004737  020042          HCRCER: JSR     PC,SPOTCK       ;SEE IF ERROR AT PACK BAD SPOT
3750  010714  000450                          BR      3$              ;EXIT IF IT IS
3751  010716  004737  020200                  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3752  010722  104414  045572                  DISPLY  ,EM20           ;REPORT 'HCRC'
3753  010726  004737  020244                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
3754  010732  004737  020652                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
3755  010736  004737  021326                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
3756  010742  032760  040000  000244          BIT     #BIT14,$RPCS2(R0)  ;'WCE' ERROR ALSO ?
3757  010750  001402                          BEQ     1$              ;BR IF NOT
3758  010752  004737  021416                  JSR     PC,LINE5        ;DISPLAY WORDS WHICH CAUSED 'WCE'
3759  010756  004737  023304          1$:     JSR     PC,INCSOF       ;INCREMENT 'SOFT' ERROR COUNT
3760  010762  004737  023424                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3761  010766  012737  000400  001250          MOV     #BIT8,MASK      ;SET ERROR MASK
3762  010774  012737  000003  001252          MOV     #3,RETRY        ;RETRY LIMIT
3763  011002  004737  015400                  JSR     PC,$RETRY       ;RETRY ORDER
3764  011006  000405                          BR      2$              ;RETRY NOT SUCESSFUL
3765  011010  004737  021700                  JSR     PC,LINE6C       ;PRINT LINE 6C OF ERROR MESSAGE
3766  011014  004737  021762                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3767  011020  000406                          BR      3$              ;EXIT
3768  011022  004737  021706          2$:     JSR     PC,LINE6D       ;PRINT LINE 6D OF ERROR MESSAGE
3769  011026  004737  021762                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3770  011032  004737  006374                  JSR     PC,REFMT        ;REFORMAT THE ERROR SECTOR
3771  011036  000207                  3$:     RTS     PC              ;RETURN
3772
3773                                  ;REPORT DRIVE ERROR
3774
3775  011040  004737  020200          DRVER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3776  011044  104414  046207                  DISPLY  ,EM30           ;REPORT DRIVE ERROR
3777  011050  004737  020244                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
3778  011054  004737  020652                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
3779  011060  004737  023424                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3780  011064  004737  021762                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3781  011070  000207                          RTS     PC              ;RETURN
3782
3783                                  ;PROCESS FORMAT ('FER') ERROR
3784
3785  011072  032760  000400  000250  CKFMT:  BIT     #BIT8,$RPER1(R0) ;'HCRC' SET ON ORIGINAL ERROR ?
3786  011100  001402                          BEQ     1$              ;BR IF NOT SET
3787  011102  000137  010710                  JMP     HCRCER          ;REPORT HCRC ERROR
3788  011106  004737  022314          1$:     JSR     PC,READDR       ;GET CORRECTED TRACK & SECTOR ADDRSSES
3789  011112  004737  015316                  JSR     PC,READHD       ;READ HEADER
3790  011116  032737  000400  044764          BIT     #BIT8,GENREG+RPER1 ;'HCRC' SET WHEN HEADER READ?
3791  011124  001002                          BNE     2$              ;BR IF 'HCRC' SET
3792  011126  000137  012072                  JMP     FMTER           ;NO, ERROR IS 'FMT' ONLY
3793  011132  004737  020042          2$:     JSR     PC,SPOTCK       ;SEE IF ERROR AT BAD SPOT ON THE PACK
3794  011136  000452                          BR      5$              ;EXIT IF IT IS
3795  011140  004737  020200                  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3796  011144  104414  045776                  DISPLY  ,EM24           ;HEADER READ ERROR - FMT BIT DROPPED UP
3797  011150  004737  020244                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
```

```
3798  011154  004737  020652                    JSR     PC,LINE3          ;PRINT LINE 3 OF ERROR MESSAGE
3799  011160  004737  021326                    JSR     PC,LINE4          ;PRINT LINE 4 OF ERROR MESSAGE
3800  011164  032760  040000  000244            BIT     #BIT14,$RPCS2(R0)   ;'WCE' ERROR ALSO ?
3801  011172  001402                            BEQ     3$                ;BR IF NOT
3802  011174  004737  021416                    JSR     PC,LINE5          ;DISPLAY WORDS WHICH CAUSED 'WCE'
3803  011200  004737  021516          3$:       JSR     PC,LINE5A         ;DISPLAY HEADER
3804  011204  004737  023304                    JSR     PC,INCSOF         ;INCREMENT SOFT ERROR COUNT
3805  011210  004737  023424                    JSR     PC,INCTOT         ;INCREMENT TOTAL ERROR COUNT
3806  011214  012737  000020  001250            MOV     #BIT4,MASK        ;SET ERROR MASK
3807  011222  012737  000003  001252            MOV     #3,RETRY          ;RETRY LIMIT
3808  011230  004737  015400                    JSR     PC,$RETRY         ;RETRY THE ORDER
3809  011234  000405                            BR      4$                ;RETRY NOT SUCESSFUL
3810  011236  004737  021700                    JSR     PC,LINE6C         ;PRINT LINE 6C OF ERROR MESSAGE
3811  011242  004737  021762                    JSR     PC,LINE7          ;PRINT LINE 7 OF ERROR MESSAGE
3812  011246  000406                            BR      5$                ;EXIT
3813  011250  004737  021706          4$:       JSR     PC,LINE6D         ;PRINT LINE 6D OF ERROR MESSAGE
3814  011254  004737  021762                    JSR     PC,LINE7          ;PRINT LINE 7 OF ERROR MESSAGE
3815  011260  004737  006374                    JSR     PC,REFMT          ;REFORMAT THE ERROR SECTOR
3816  011264  000207                5$:       RTS     PC                ;RETURN
3817
3818                                 ;PROCESS HEADER COMPARE ('HCE') ERROR
3819
3820  011266  032760  000400  000250  CKHCE:   BIT     #BIT8,$RPER1(R0)  ;HCRC SET ON ORIGINAL ERROR ?
3821  011274  001402                            BEQ     1$                ;BR IF NOT SET
3822  011276  000137  010710                    JMP     HCRCER            ;REPORT HEADER CRC ERROR
3823  011302  004737  022314          1$:       JSR     PC,READDR         ;GET CURRENT SECTOR & TRACK ADDRS
3824  011306  004737  015316                    JSR     PC,READHD         ;READ HEADER OF CURRENT SECTOR
3825  011312  032737  000400  044764            BIT     #BIT8,GENREG+RPER1  ;'HCRC' SET ?
3826  011320  001016                            BNE     3$                ;BR IF SET
3827  011322  042737  010000  054456            BIC     #BIT12,CYLDER     ;CLEAR FORMAT BIT FROM HEADER
3828  011330  026037  000272  054456            CMP     $RPCC(R0),CYLDER    ;CORRECT CYLINDER ?
3829  011336  001402                            BEQ     2$                ;BR IF IT IS
3830  011340  000137  011512                    JMP     POSER             ;REPORT POSITIONING ERROR
3831  011344  052737  010000  054456  2$:       BIS     #BIT12,CYLDER     ;RESTORE THE FORMAT BIT
3832  011352  000137  012150                    JMP     HCEER             ;REPORT 'HCE' ERROR
3833  011356  004737  020042          3$:       JSR     PC,SPOTCK         ;SEE IF ERROR AT BAD SPOT
3834  011362  000452                            BR      6$                ;EXIT IF IT IS
3835  011364  004737  020200                    JSR     PC,LINE1          ;PRINT LINE 1 OF ERROR MESSAGE
3836  011370  104414  046044                    DISPLY  ,EM25             ;HEADER READ ERROR - 'HCE' SET
3837  011374  004737  020244                    JSR     PC,LINE2          ;PRINT LINE 2 OF ERROR MESSAGE
3838  011400  004737  020652                    JSR     PC,LINE3          ;PRINT LINE 3 OF ERROR MESSAGE
3839  011404  004737  021326                    JSR     PC,LINE4          ;PRINT LINE 4 OF ERROR MESSAGE
3840  011410  032760  040000  000244            BIT     #BIT14,$RPCS2(R0)   ;'WCE' ERROR ALSO ?
3841  011416  001402                            BEQ     4$                ;BR IF NOT
3842  011420  004737  021416                    JSR     PC,LINE5          ;DISPLAY WORDS WHICH CAUSED 'WCE'
3843  011424  004737  021516          4$:       JSR     PC,LINE5A         ;PRINT LINE 5 OF ERROR MESSAGE
3844  011430  004737  023304                    JSR     PC,INCSOF         ;INCREMENT SOFT ERROR COUNT
3845  011434  004737  023424                    JSR     PC,INCTOT         ;INCREMENT TOTAL ERROR COUNT
3846  011440  012737  000200  001250            MOV     #BIT7,MASK        ;SET ERROR MASK
3847  011446  012737  000003  001252            MOV     #3,RETRY          ;RETRY LIMIT
3848  011454  004737  015400                    JSR     PC,$RETRY         ;RETRY THE ORDER
3849  011460  000405                            BR      5$                ;RETRY NOT SUCESSFUL
3850  011462  004737  021700                    JSR     PC,LINE6C         ;PRINT LINE 6C OF ERROR MESSAGE
3851  011466  004737  021762                    JSR     PC,LINE7          ;PRINT LINE 7 OF ERROR MESSAGE
3852  011472  000406                            BR      6$                ;EXIT
3853  011474  004737  021706          5$:       JSR     PC,LINE6D         ;PRINT LINE 6D OF ERROR MESSAGE
```

```
3854  011500  004737  021762          JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3855  011504  004737  006374          JSR     PC,REFMT        ;REFORMAT THE ERROR SECTOR
3856  011510  000207          6$:     RTS     PC              ;RETURN
3857
3858                                  ;REPORT POSSIBLE POSITIONING ERROR
3859
3860  011512  004737  015240  POSER:  JSR     PC,RECALT       ;RECALIBRATE
3861  011516  004737  020200          JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3862  011522  104414  047372          DISPLY  .EM51           ;PROGRAM DETECTED POSITIONING ERROR
3863  011526  004737  020244          JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
3864  011532  004737  020700          JSR     PC,LINE3C       ;PRINT LINE 3C OF ERROR MESSAGE
3865  011536  052737  010000  054456  BIS     #BIT12,CYLDER   ;RESTORE THE FORMAT BIT
3866  011544  004737  021516          JSR     PC,LINE5A       ;PRINT LINE 5A OF THE ERROR MESSAGE
3867  011550  004737  023400          JSR     PC,INCMIS       ;INCREMENT MISPOSITIONING COUNT
3868  011554  004737  023424          JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3869  011560  004737  022110          JSR     PC,LINE7A       ;PRINT LINE 7A OF ERROR MESSAGE
3870  011564  000207                  RTS     PC              ;EXIT
3871
3872                                  ;REPORT 'OPI' ERROR
3873
3874  011566  004737  020042  OPIER:  JSR     PC,SPOTCK       ;SEE IF ERROR AT BAD SPOT
3875  011572  000207                  RTS     PC              ;RETURN IF IT IS
3876  011574  004737  020200          JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3877  011600  104414  046241          DISPLY  .EM31           ;'OPI' ERROR
3878  011604  004737  020244          JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
3879  011610  004737  020652          JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
3880  011614  004737  021326          JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
3881  011620  004737  023424          JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3882  011624  012737  020000  001250  MOV     #BIT13,MASK     ;ERROR MASK
3883  011632  012737  000003  001252  OPIER1: MOV     #3,RETRY        ;RETRY LIMIT
3884  011640  004737  015400          JSR     PC,$RETRY       ;RETRY THE ORDER
3885  011644  000405                  BR      1$              ;RETRY UNSUCESSFUL
3886  011646  004737  021700          JSR     PC,LINE6C       ;PRINT LINE 6C OF ERROR MESSAGE
3887  011652  004737  021762          JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3888  011656  000207                  RTS     PC              ;EXIT
3889  011660  004737  021706  1$:     JSR     PC,LINE6D       ;PRINT LINE 6D OF ERROR MESSAGE
3890  011664  004737  021762          JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3891  011670  004737  006374          JSR     PC,REFMT        ;REFORMAT THE ERROR SECTOR
3892  011674  000207                  RTS     PC              ;RETURN
3893
3894                                  ;REPORT 'DTE' ERROR
3895
3896  011676  004737  020042  DTEER:  JSR     PC,SPOTCK       ;SEE IF ERROR AT BAD SPOT
3897  011702  000207                  RTS     PC              ;RETURN IF IT IS
3898  011704  004737  020200          JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3899  011710  104414  046304          DISPLY  .EM32           ;'DTE' ERROR
3900  011714  000137  007722          JMP     DCKER1          ;FINISH PROCESSING THE 'DTE' ERROR
3901
3902                                  ;REPORT 'PAR' ERROR
3903
3904  011720  004737  020200  PARER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3905  011724  104414  046337          DISPLY  .EM33           ;REPORT 'PAR'
3906  011730  004737  020244          JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
3907  011734  004737  020756          JSR     PC,LINE3E       ;PRINT LINE 3E OF ERROR MESSAGE
3908  011740  004737  021326          JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
3909  011744  004737  023424          JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
```

```
3910  011750  012737  000010  001250          MOV     #BIT03,MASK     ;ERROR MASK
3911  011756  012737  000003  001252          MOV     #3,RETRY        ;RETRY LIMIT
3912  011764  004737  015400                  JSR     PC,$RETRY       ;RETRY ORDER
3913  011770  000405                          BR      2$              ;RETRY UNSUCESSFUL
3914  011772  004737  021700                  JSR     PC,LINE6C       ;RETRY SUCESSFUL
3915  011776  004737  021762          1$:     JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3916  012002  000207                          RTS     PC              ;EXIT
3917  012004  004737  021706          2$:     JSR     PC,LINE6D       ;PRINT LINE 6D OF ERROR MESSAGE
3918  012010  000772                          BR      1$              ;FINISH ERROR MESSAGE
3919
3920                                  ;REPORT 'IAE' ERROR
3921
3922  012012  004737  020200          IAEER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3923  012016  104414  046456                  DISPLY  ,EM35           ;REPORT 'IAE'
3924  012022  004737  020244                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
3925  012026  004737  021044                  JSR     PC,LINE3F       ;PRINT LINE 3F OF ERROR MESSAGE
3926  012032  004737  023424                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3927  012036  004737  021762                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3928  012042  000207                          RTS     PC              ;RETURN
3929
3930                                  ;REPORT 'WLE' ERROR
3931
3932  012044  004737  020200          WLEER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3933  012050  104414  046514                  DISPLY  ,EM36           ;REPORT 'WLE'
3934  012054  004737  020244                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
3935  012060  004737  023424                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3936  012064  004737  021762                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3937  012070  000207                          RTS     PC              ;RETURN
3938
3939                                  ;REPORT FORMAT ERROR
3940
3941  012072  004737  020200          FMTER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3942  012076  104414  046125                  DISPLY  ,EM26           ;FORMAT ERROR
3943  012102  004737  020244                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
3944  012106  004737  020652                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
3945  012112  004737  021326                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
3946  012116  032760  040000  000244          BIT     #BIT14,$RPCS2(R0)  ;'WCE' ERROR ALSO ?
3947  012124  001402                          BEQ     1$              ;BR IF NOT
3948  012126  004737  021416                  JSR     PC,LINE5        ;DISPLAY WORDS WHICH CAUSED 'WCE'
3949  012132  004737  021516          1$:     JSR     PC,LINE5A       ;PRINT LINE 5A OF ERROR MESSAGE
3950  012136  004737  023424                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3951  012142  004737  021762                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3952  012146  000207                          RTS     PC
3953
3954                                  ;REPORT HEADER COMPARE ERROR
3955
3956  012150  004737  020200          HCEER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3957  012154  104414  046152                  DISPLY  ,EM27           ;HEADER COMPARE ERROR
3958  012160  004737  020244                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
3959  012164  004737  020652                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
3960  012170  004737  021326                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
3961  012174  032760  040000  000244          BIT     #BIT14,$RPCS2(R0)  ;'WCE' ERROR ALSO ?
3962  012202  001402                          BEQ     1$              ;BR IF NOT
3963  012204  004737  021416                  JSR     PC,LINE5        ;DISPLAY WORDS WHICH CAUSED 'WCE'
3964  012210  004737  021516          1$:     JSR     PC,LINE5A       ;PRINT LINE 5A OF ERROR MESSAGE
3965  012214  004737  023424                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
```

```
3966  012220  004737  021762                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3967  012224  004737  006374                  JSR     PC,REFMT        ;REFORMAT THE ERROR SECTOR
3968  012230  000207                          RTS     PC              ;RETURN
3969
3970                                  ;PROCESS CONTROL/INTERFACE TRANSFER ERROR
3971
3972  012232  004737  020200          TRFER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3973  012236  104414  046627                  DISPLY  ,EM40           ;RH11 OR UNIBUS TRANSFER ERROR
3974  012242  004737  020244                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
3975  012246  004737  020652                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
3976  012252  004737  021326                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
3977  012256  004737  023424                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3978  012262  032760  121400  000244          BIT     #BIT15!BIT13!BIT9!BIT8,$RPCS2(R0)  ;'DLT','UPE','MXF', 'MDPE' SET ?
3979  012270  001415                          BEQ     2$              ;BR IF NONE SET
3980  012272  012737  000003  001252          MOV     #3,RETRY        ;RETRY LIMIT
3981  012300  005037  001250                  CLR     MASK            ;CLEAR ERROR MASK
3982  012304  004737  015400                  JSR     PC,$RETRY       ;RETRY THE OPERATION
3983  012310  000403                          BR      1$              ;RETURN HERE IF RETRY UNSUCESSFUL
3984  012312  004737  021700                  JSR     PC,LINE6C       ;PRINT LINE 6C OF ERROR MESSAGE
3985  012316  000402                          BR      2$              ;FINISH THE ERROR REPORT
3986  012320  004737  021706          1$:     JSR     PC,LINE6D       ;PRINT LINE 6D OF ERROR MESSAGE
3987  012324  004737  021762          2$:     JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
3988  012330  000207                          RTS     PC
3989
3990                                  ;PROCESS 'SKI' OR 'OCYL' ERRORS
3991
3992  012332  004737  020200          SKIER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
3993  012336  104414  047303                  DISPLY  ,EM50           ;'SKI' OR 'OCYL' ERROR
3994  012342  004737  020244                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
3995  012346  004737  020666                  JSR     PC,LINE3B       ;PRINT LINE 3B OF ERROR MESSAGE
3996  012352  004737  023424                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
3997  012356  004737  023354                  JSR     PC,INCSKI       ;INCREMENT 'SKI' OR 'OCYL' ERROR COUNT
3998  012362  004737  022110                  JSR     PC,LINE7A       ;PRINT LINE 7A OF ERROR MESSAGE
3999  012366  000207                          RTS     PC
4000
4001                                  ;REPORT WRITE CLOCK FAILURE ('WCF')
4002
4003  012370  004737  020200          WCFER:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
4004  012374  104414  046414                  DISPLY  ,EM34           ;REPORT WRITE CLOCK FAILURE
4005  012400  004737  020244                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
4006  012404  004737  020660                  JSR     PC,LINE3A       ;PRINT LINE 3A OF ERROR MESSAGE
4007  012410  004737  021326                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
4008  012414  004737  023424                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
4009  012420  004737  015042                  JSR     PC,PRTBAD       ;SEE IF BAD SECTOR TO BE PRINTED
4010  012424  012737  000003  001252          MOV     #3,RETRY        ;RETRY COUNT
4011  012432  012737  000040  001250          MOV     #BIT05,MASK     ;ERROR MASK
4012  012440  004737  015400                  JSR     PC,$RETRY       ;RETRY THE ORDER
4013  012444  000405                          BR      2$              ;RETURN HERE IF RETRY UNSUCESSFUL
4014  012446  004737  021700                  JSR     PC,LINE6C       ;PRINT LINE 6C OF ERROR MESSAGE
4015  012452  004737  021762          1$:     JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
4016  012456  000207                          RTS     PC
4017  012460  004737  021706          2$:     JSR     PC,LINE6D       ;PRINT LINE 6D OF ERROR MESSAGE
4018  012464  000772                          BR      1$
4019
4020                                  ;PROCESS DRIVE UNSAFE ERROR
4021
```

```
4022  012466  004737  020200          UNSAF:  JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
4023  012472  104414  047435                  DISPLY  ,EM60           ;REPORT DRIVE UNSAFE
4024  012476  004737  020244                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
4025  012502  004737  020652                  JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
4026  012506  004737  023424                  JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
4027  012512  012737  000003  001252          MOV     #3,RETRY        ;RETRY COUNT
4028  012520  004737  015400                  JSR     PC,$RETRY       ;RETRY THE ORDER
4029  012524  000403                          BR      1$              ;RETRY WAS UNSUCESSFUL
4030  012526  004737  021700                  JSR     PC,LINE6C       ;PRINT LINE 6C OF ERROR MESSAGE
4031  012532  000402                          BR      2$              ;CONTINUE WITH ERROR REPORT
4032  012534  004737  021706          1$:     JSR     PC,LINE6D       ;PRINT LINE 6D OF ERROR MESSAGE
4033  012540  004737  021762          2$:     JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
4034  012544  000207                          RTS     PC              ;RETURN
4035
4036                                  ;REPORT AN 'UNKNOWN' DATA PATTERN
4037
4038  012546  105737  001300          NOMTCH: TSTB    FRSTER          ;FIRST ERROR IN THE SECTOR ?
4039  012552  001013                          BNE     1$              ;BR IF NOT OR IF PROCESSING 'DCKER'
4040  012554  004737  020200                  JSR     PC,LINE1        ;TYPE LINE 1 OF ERROR MESSAGE
4041  012560  104414  047012                  DISPLY  ,EM43           ;'CAN'T MATCH DATA WITH PATTERN'
4042  012564  004737  020244                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
4043  012570  004737  020660                  JSR     PC,LINE3A       ;PRINT LINE 3A OF ERROR MESSAGE
4044  012574  004737  021326                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
4045  012600  000404                          BR      2$              ;CONTINUE PROCESSING ERROR
4046  012602  104414  047012          1$:     DISPLY  ,EM43           ;'CAN'T MATCH DATA WITH PATTERN'
4047  012606  104414  001165                  DISPLY  ,$CRLF          ;CR-LF
4048  012612  104414  051472          2$:     DISPLY  ,LIN9I          ;HEADER FOR DATA PRINTOUT
4049  012616  010146                          MOV     R1,-(SP)        ;ADDRESS OF WORD 1
4050  012620  004737  022242                  JSR     PC,LINOCT       ;TYPE WORD 1
4051  012624  104414  052155                  DISPLY  ,LINSP          ;SPACES
4052  012630  012146                          MOV     (R1)+,-(SP)     ;ADDRESS OF WORD 1
4053  012632  004737  022242                  JSR     PC,LINOCT       ;TYPE WORD 1
4054  012636  104414  001165                  DISPLY  ,$CRLF          ;CR-LF
4055  012642  010146                          MOV     R1,-(SP)        ;ADDRESS OF WORD 2
4056  012644  004737  022242                  JSR     PC,LINOCT       ;TYPE WORD 2
4057  012650  104414  052155                  DISPLY  ,LINSP          ;SPACES
4058  012654  012146                          MOV     (R1)+,-(SP)     ;ADDRESS OF WORD 2
4059  012656  004737  022242                  JSR     PC,LINOCT       ;TYPE WORD 2
4060  012662  104414  001165                  DISPLY  ,$CRLF          ;CR-LF
4061  012666  010146                          MOV     R1,-(SP)        ;ADDRESS OF WORD 3
4062  012670  004737  022242                  JSR     PC,LINOCT       ;TYPE WORD 3
4063  012674  104414  052155                  DISPLY  ,LINSP          ;SPACES
4064  012700  012146                          MOV     (R1)+,-(SP)     ;ADDRESS OF WORD 3
4065  012702  004737  022242                  JSR     PC,LINOCT       ;TYPE WORD 3
4066  012706  104414  001165                  DISPLY  ,$CRLF          ;CR-LF
4067  012712  010146                          MOV     R1,-(SP)        ;ADDRESS OF WORD 4
4068  012714  004737  022242                  JSR     PC,LINOCT       ;TYPE WORD 4
4069  012720  104414  052155                  DISPLY  ,LINSP          ;SPACES
4070  012724  012146                          MOV     (R1)+,-(SP)     ;ADDRESS OF WORD 4
4071  012726  004737  022242                  JSR     PC,LINOCT       ;TYPE WORD 4
4072  012732  104414  001165                  DISPLY  ,$CRLF          ;CR-LF
4073  012736  062701  000770                  ADD     #<252.*2.>,R1   ;INCREMENT BUFFER POINTER
4074  012742  005002                          CLR     R2              ;CLEAR 'WORDS TO COMPARE' COUNT IN R2
4075  012744  112737  000001  001300          MOVB    #1,FRSTER       ;SET 'NOT FIRST ERROR' INDICATOR
4076  012752  112737  177777  001301          MOVB    #-1,FRSTER+1    ;SET ERROR FOUND INDICATOR
4077  012760  013737  001414  001310          MOV     CMPLMT,LIMIT    ;RESET THE COMPARE ERROR TYPEOUT LIMIT
```

```
4078  012766  000207                        RTS     PC              ;RETURN
4079
4080                                  ;CHECK ERROR BITS IN THE RH11 & RP04/5/6 REGISTERS
4081
4082  012770  032760  060000  000234  CKERR:  BIT     #60000,$RPCS1(R0)  ;SEE IF 'TRE' OR 'MCPE' SET
4083  012776  001015                        BNE     1$              ;BR IT EITHER SET
4084  013000  032760  177400  000244        BIT     #177400,$RPCS2(R0)  ;SEE IF ERROR BITS IN CS2 SET
4085  013006  001011                        BNE     1$              ;BR IF ANY SET
4086  013010  005760  000250              TST     $RPER1(R0)      ;ANY BITS SET IN ER1
4087  013014  001006                        BNE     1$              ;BR IF ANY SET
4088  013016  005760  000274              TST     $RPER2(R0)      ;ANY BITS SET IN ER2 ?
4089  013022  001003                        BNE     1$              ;BR IF ANY SET
4090  013024  005760  000276              TST     $RPER3(R0)      ;ANY BITS SET IN ER3 ?
4091  013030  001416                        BEQ     2$              ;BR IF NONE SET
4092  013032  004737  020200        1$:     JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
4093  013036  104414  047057              DISPLY  EM44            ;ERROR BITS SET, BUT 'SC' OR 'TRE' NOT SET
4094  013042  004737  020244              JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
4095  013046  004737  020652              JSR     PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
4096  013052  004737  021326              JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
4097  013056  004737  023424              JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
4098  013062  004737  021762              JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
4099  013066  000207                2$:     RTS     PC              ;RETURN
4100
4101                                  ;CHECK BUS ADDRESS REGISTER & WORD COUNT REGISTER
4102
4103  013070  005760  000236        CKBUS:  TST     $RPWC(R0)       ;CHECK WORD COUNT
4104  013074  001010                        BNE     1$              ;BR IF NOT ZERO
4105  013076  016046  000020              MOV     $WRDL(R0),-(SP) ;WORD LENGTH
4106  013102  006316                        ASL     (SP)            ;CHANGE INTO BYTE COUNT
4107  013104  066016  000006              ADD     $BUF(R0),(SP)   ;ADD THE STARTING LOCATION
4108  013110  022660  000240              CMP     (SP)+,$RPBA(R0) ;BUFFER ADDRESS PROPER ?
4109  013114  001416                        BEQ     2$              ;BR IF OK
4110  013116  004737  020200        1$:     JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
4111  013122  104414  046665              DISPLY  EM41            ;BUS ADDRESS OR WORD COUNT INCORRECT
4112  013126  004737  020244              JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
4113  013132  004737  020710              JSR     PC,LINE3D       ;PRINT LINE 3D OF ERROR MESSAGE
4114  013136  004737  021326              JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
4115  013142  004737  023424              JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
4116  013146  004737  021762              JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
4117  013152  000207                2$:     RTS     PC
4118
4119                                  ;COMPARE THE BUFFER
4120
4121  013154  132760  000004  000024  CMPAR:  BITB    #BIT02,$CODE(R0)  ;SEE IF READ ORDER
4122  013162  001001                        BNE     1$              ;BR IF IT IS
4123  013164  000207                        RTS     PC              ;RETURN
4124  013166  005037  001300        1$:     CLR     FRSTER          ;CLEAR 'FIRST ERROR' INDICATOR
4125  013172  032777  000002  165740  CMPARD: BIT     #SW01,@SWR      ;IS SWITCH 1 SET?
4126  013200  001401                        BEQ     1$              ;BR IF NOT
4127  013202  000207                        RTS     PC              ;YES  DON'T COMPARE
4128  013204  005037  001306        1$:     CLR     ERCTR           ;CLEAR THE ERROR COUNTER
4129  013210  016001  000006              MOV     $BUF(R0),R1     ;BUFFER ADDRESS
4130  013214  016037  000020  001312        MOV     $WRDL(R0),CMCNT  ;WORD COUNT TO WORKING LOCATION
4131  013222  066037  000236  001312        ADD     $RPWC(R0),CMCNT  ;CALCULATE ACTUAL WORDS TRANSFERED
4132  013230  016037  000012  001314        MOV     $CYL(R0),CMCYL  ;CYLINDER ADDRESS WORKING LOCATION
4133  013236  052737  010000  001314        BIS     #BIT12,CMCYL    ;SET FORMAT BIT
```

```
4134  013244  016037  000010  001316          MOV     $SEC(R0),CMSEC  ;SECTOR & TRACK ADDRESSES TO WORKING LOCNS
4135  013252  013737  001414  001310          MOV     CMPLMT,LIMIT    ;DISPLAY LIMIT
4136  013260  005237  001310                  INC     LIMIT           ;CONVERT PARAMETER INTO LIMIT VALUE
4137  013264  012737  177777  001276  CMSTR:  MOV     #-1,ZROIND      ;CLEAR THE 'ZERO'S' INDICATOR
4138  013272  005037  001302                  CLR     SAVER1          ;CLEAR THE R1 SAVE WORD
4139  013276  005037  001304                  CLR     SAVER5          ;CLEAR THE R5 SAVE WORD
4140  013302  023760  001312  000022          CMP     CMCNT,$SSEC(R0) ;IS BUFFER SIZE GREATER THAN ONE SECTOR ?
4141  013310  101003                          BHI     1$              ;BR IF IT IS
4142  013312  013702  001312                  MOV     CMCNT,R2        ;LESS THAN, USE REMAINING BUFFER
4143  013316  000402                          BR      2$              ;
4144  013320  016002  000022          1$:     MOV     $SSEC(R0),R2    ;COMPARE SECTOR
4145  013324  166037  000022  001312  2$:     SUB     $SSEC(R0),CMCNT ;DECREMENT WORD COUNT
4146  013332  126027  000024  000005          CMPB    $CODE(R0),#5    ;READ HEADER & DATA?
4147  013340  001036                          BNE     CMDAT           ;BR IF NOT
4148  013342  023721  001314          CMHED:  CMP     CMCYL,(R1)+     ;CHECK CYLINDER
4149  013346  001402                          BEQ     1$              ;BR IF COMPARE OK
4150  013350  004737  013424                  JSR     PC,5$           ;REPORT ERROR
4151  013354  023721  001316          1$:     CMP     CMSEC,(R1)+     ;COMPARE SECTOR & TRACK
4152  013360  001402                          BEQ     2$              ;BR IF EQ
4153  013362  004737  013424                  JSR     PC,5$           ;REPORT ERROR
4154  013366  005721                  2$:     TST     (R1)+           ;1ST KEY WORD ZERO?
4155  013370  001402                          BEQ     3$              ;BR IF IT IS
4156  013372  004737  013424                  JSR     PC,5$           ;REPORT ERROR
4157  013376  005721                  3$:     TST     (R1)+           ;CHECK 2ND KEY WORD
4158  013400  001402                          BEQ     4$              ;BR IF ZERO
4159  013402  004737  013424                  JSR     PC,5$           ;REPORT THE ERROR
4160  013406  162702  000004          4$:     SUB     #4,R2           ;SUBTRACT HEADER LENGTH FROM SIZE
4161  013412  003530                          BLE     CMPRX           ;BR IF FINISHED
4162  013414  022702  000004                  CMP     #4,R2           ;SEE IF AT LEAST 4 MORE WORDS TO CHECK
4163  013420  101125                          BHI     CMPRX           ;BR IF NOT
4164  013422  000405                          BR      CMDAT           ;COMPARE THE DATA PORTION
4165  013424  005237  001306          5$:     INC     ERCTR           ;INCREMENT THE ERROR COUNT
4166  013430  004737  013702                  JSR     PC,CMPRT        ;REPORT THE COMPARISON ERROR
4167  013434  000207                          RTS     PC              ;CHECK THE REST OF THE HEADER
4168  013436  004737  014224          CMDAT:  JSR     PC,MATCH        ;FIND THE PATTERN
4169  013442  000403                          BR      2$              ;FOUND A PATTERN
4170  013444  004737  012546                  JSR     PC,NOMTCH       ;RETURN HERE IF NO MATCH WITH PATTERN MADE
4171  013450  000456                          BR      8$              ;BYPASS COMPARE ROUTINE
4172  013452  011405                  2$:     MOV     (R4),R5         ;ADDRESS OF PATTERN ADDRESS IN R4
4173  013454  012703  000020                  MOV     #20,R3          ;R3 IS PATTERN POS COUNTER
4174  013460  022125                  3$:     CMP     (R1)+,(R5)+     ;COMPARE BUFFER WITH PATTERN
4175  013462  001016                          BNE     5$              ;BR IF NOT EQUAL
4176  013464  005737  001306                  TST     ERCTR           ;ERRORS DETECTED ?
4177  013470  001406                          BEQ     4$              ;BR IF NO ERRORS
4178  013472  032777  000010  165440          BIT     #SW3,@SWR       ;SWITCH 3 SET ?
4179  013500  001402                          BEQ     4$              ;BR IF NOT SET
4180  013502  004737  013702                  JSR     PC,CMPRT        ;DISPLAY THE WORD
4181  013506  005302                  4$:     DEC     R2              ;DECREMENT SIZE COUNT
4182  013510  001436                          BEQ     8$              ;BR WHEN AT END
4183  013512  005303                          DEC     R3              ;DECREMENT PATT POS COUNT
4184  013514  001361                          BNE     3$              ;BR IF NOT AT END OF PATT
4185  013516  000755                          BR      2$              ;RESTART THE PATTERN
4186  013520  005761  177776          5$:     TST     -2(R1)          ;IS MISCOMPARED CHARACTER=0
4187  013524  001410                          BEQ     6$              ;BR IF YES
4188  013526  012737  177777  001276          MOV     #-1,ZROIND      ;SET NON-ZERO MISCOMPARED INDCATOR
4189  013534  005237  001306                  INC     ERCTR           ;INCREMENT THE ERROR COUNTER
```

```
 4190  013540  004737  013702                 JSR     PC,CMPRT        ;REPORT ERROR
 4191  013544  000760                         BR      4$              ;CONTINUE COMPARE
 4192  013546  105737  001300         6$:     TSTB    FRSTER          ;FIRST ERROR?
 4193  013552  100407                         BMI     7$              ;BR IF NOT
 4194  013554  005037  001276                 CLR     ZROIND          ;SET THE ZERO INDICATOR
 4195  013560  010137  001302                 MOV     R1,SAVER1       ;SAVE CURRENT R1
 4196  013564  010537  001304                 MOV     R5,SAVER5       ;SAVE CURRENT R5
 4197  013570  000746                         BR      4$              ;CONTINUE COMPARE
 4198  013572  005737  001276         7$:     TST     ZROIND          ;ANY MISCOMPARIONS NOT ZEROS ?
 4199  013576  001743                         BEQ     4$              ;BR IF NONE-ALL ERRORS=ZERO
 4200  013600  004737  013702                 JSR     PC,CMPRT        ;REPORT ERROR
 4201  013604  000740                         BR      4$              ;CONTINUE COMPARING
 4202  013606  005737  001312         8$:     TST     CMCNT           ;AT END OF BUFFER
 4203  013612  003430                         BLE     CMPRX           ;BR IF AT END
 4204  013614  126027  000024  000005         CMPB    $CODE(R0),#5    ;SEE IF READ HEADER & DATA
 4205  013622  001220                         BNE     CMSTR           ;BR IF NOT
 4206  013624  105237  001316                 INCB    CMSEC           ;INCREMENT SECTOR
 4207  013630  123727  001316  000026         CMPB    CMSEC,#22.      ;SECTOR GREATER THAN MAX ?
 4208  013636  103612                         BLO     CMSTR           ;BR IF NOT GREATER THAN MAX
 4209  013640  105037  001316                 CLRB    CMSEC           ;CLEAR SECTOR ADDRESS
 4210  013644  105237  001317                 INCB    CMTRK           ;INCREMENT TRACK
 4211  013650  123727  001317  000023         CMPB    CMTRK,#19.      ;TRACK GREATER THAN MAX ?
 4212  013656  103602                         BLO     CMSTR           ;BR IF NOT GREATER
 4213  013660  105037  001317                 CLRB    CMTRK           ;RESET TRACK ADDRESS
 4214  013664  005237  001314                 INC     CMCYL           ;INCREMENT CYLINDER ADDRESS
 4215  013670  000137  013264                 JMP     CMSTR           ;CONTINUE WITH COMPARE
 4216  013674  004737  014156         CMPRX:  JSR     PC,ENDCMP       ;PRINT LAST LINE IF ERRORS
 4217  013700  000207                         RTS     PC
 4218
 4219
 4220                                         ;TYPE DATA COMPARE ERRORS
 4221  013702  005737  001302         CMPRT:  TST     SAVER1          ;PRINT SAVED VALUES ?
 4222  013706  001010                         BNE     2$              ;BR IF NOT
 4223  013710  105737  001300                 TSTB    FRSTER          ;FIRST ERROR?
 4224  013714  100402                         BMI     1$              ;BR IF NOT
 4225  013716  004737  013776                 JSR     PC,4$           ;PRINT INITIAL MESSAGE INFO
 4226  013722  004737  014060         1$:     JSR     PC,8$           ;PRINT REMAINDER OF MESSAGE
 4227  013726  000422                         BR      3$              ;EXIT
 4228  013730                         2$:
 4229  013730  010146                         MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 4230  013732  010546                         MOV     R5,-(SP)        ;;PUSH R5 ON STACK
 4231  013734  013701  001302                 MOV     SAVER1,R1       ;DISPLAY SAVED R1
 4232  013740  013705  001304                 MOV     SAVER5,R5       ;DISPLAY SAVED R5
 4233  013744  004737  013776                 JSR     PC,4$           ;PRINT INITIAL MESSAGE INFO
 4234  013750  004737  014060                 JSR     PC,8$           ;PRINT SAVED VALUES
 4235  013754  005037  001302                 CLR     SAVER1          ;CLEAR SAVED REGISTER INDICATORS
 4236  013760  005037  001304                 CLR     SAVER5          ;CLEAR THE OTHER ONE
 4237  013764  012605                         MOV     (SP)+,R5        ;;POP STACK INTO R5
 4238  013766  012601                         MOV     (SP)+,R1        ;;POP STACK INTO R1
 4239  013770  004737  014060                 JSR     PC,8$           ;PRINT REMAINDER OF MESSAGE
 4240  013774  000207                 3$:     RTS     PC              ;RETURN
 4241  013776  105737  001300         4$:     TSTB    FRSTER          ;FIRST ERROR ?
 4242  014002  100425                         BMI     7$              ;BR IF NOT
 4243  014004  001013                         BNE     5$              ;BR IF FIRST ERROR AND PROCESSING 'DCK' ERROR
 4244  014006  004737  020200                 JSR     PC,LINE1        ;PRINT LINE 1 OF ERROR MESSAGE
 4245  014012  104414  046731                 DISPLY  ,EM42           ;DATA COMPARE ERROR
```

```
4246  014016  004737  020244                  JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
4247  014022  004737  020660                  JSR     PC,LINE3A       ;PRINT LINE 3A OF ERROR MESSAGE
4248  014026  004737  021326                  JSR     PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
4249  014032  000404                          BR      6$              ;GO TO TYPE HEADER
4250  014034  104414  051365          5$:     DISPLY  ,LIN9B          ;HEADER MESSAGE OF PROCESSING 'DCK' ERROR
4251  014040  104414  001165                  DISPLY  ,$CRLF          ;CR-LF
4252  014044  104414  051414          6$:     DISPLY  ,LIN9H          ;DISPLAY HEADER
4253  014050  012737  177777  001300          MOV     #-1,FRSTER      ;SET ERROR FLAG
4254  014056  000207                  7$:     RTS     PC              ;RETURN
4255  014060  005737  001310          8$:     TST     LIMIT           ;TYPEOUT LIMIT REACHED ?
4256  014064  001403                          BEQ     9$              ;BR IF IT HAS
4257  014066  005337  001310                  DEC     LIMIT           ;DECREMENT LIMIT COUNTER
4258  014072  001005                          BNE     10$             ;BR IF NOT AT LIMIT
4259  014074  032777  000200  165036  9$:     BIT     #SW07,@SWR      ;PRINT ALL DATA COMPARE ERRORS ?
4260  014102  001001                          BNE     10$             ;BR IF YES
4261  014104  000207                          RTS     PC              ;RETURN
4262  014106  010146                  10$:    MOV     R1,-(SP)        ;BUFFER ADDRESS
4263  014110  162716  000002                  SUB     #2,(SP)         ;ADJUST ADDRESS
4264  014114  004737  022242                  JSR     PC,LINOCT       ;TYPE IT
4265  014120  104414  052155                  DISPLY  ,LINSP          ;2 SPACES
4266  014124  016546  177776                  MOV     -2(R5),-(SP)    ;PUT GOOD DATA ON THE STACK
4267  014130  004737  022242                  JSR     PC,LINOCT       ;TYPE IT
4268  014134  104414  052155                  DISPLY  ,LINSP          ;2 SPACES
4269  014140  016146  177776                  MOV     -2(R1),-(SP)    ;BAD DATA
4270  014144  004737  022242                  JSR     PC,LINOCT       ;TYPE IT
4271  014150  104414  001165                  DISPLY  ,$CRLF          ;CR-LF
4272  014154  000207                          RTS     PC              ;RETURN
4273
4274                                  ;LAST LINE OF COMPARE ERROR REPORTING
4275
4276  014156  105737  001301          ENDCMP: TSTB    FRSTER+1        ;ANY COMPARE ERRORS FOUND ?
4277  014162  001417                          BEQ     2$              ;BR IF NOT
4278  014164  005737  001306                  TST     ERCTR           ;SEE HOW MANY ERRORS
4279  014170  001410                          BEQ     1$              ;BR IF ONLY CAN'T MATCH PATTERN
4280  014172  104414  051512                  DISPLY  ,LIN9E          ;'NUMBER OF ERRORS='
4281  014176  013746  001306                  MOV     ERCTR,-(SP)     ;NUMBER OF ERRORS
4282  014202  004737  022274                  JSR     PC,LINDEC       ;TYPE IT
4283  014206  104414  001165                  DISPLY  ,$CRLF          ;CR-LF
4284  014212  004737  023424          1$:     JSR     PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
4285  014216  004737  021762                  JSR     PC,LINE7        ;PRINT LINE 7 OF ERROR MESSAGE
4286  014222  000207                  2$:     RTS     PC              ;RETURN
4287
4288
4289                                  ;ROUTINE TO MATCH THE DATA WITH A PATTERN
4290                                  ;CALL:
4291                                  ;       MOV     #BUFFER,R1      ;BUFFER ADDRESS
4292                                  ;       JSR     PC,MATCH
4293                                  ;       RETURN1                 ;PATTERN ADDRESS IN R4
4294                                  ;       RETURN2                 ;COULDN'T MATCH PATTERN
4295
4296  014224  010146                  MATCH:  MOV     R1,-(SP)        ;SAVE R1 ON THE STACK
4297  014226  012704  000044                  MOV     #44,R4          ;PATTERN TABLE INDEX
4298  014232  011601                  1$:     MOV     (SP),R1         ;RELOAD R1
4299  014234  162704  000002                  SUB     #2,R4           ;DECREMENT INDEX
4300  014240  016405  002762                  MOV     STNDAT(R4),R5   ;ADDRESS OF PATTERN ADDRESS
4301  014244  001411                          BEQ     3$              ;BR IF ALL PATTERNS CHECKED AND NO MATCH
```

```
                                                                 ;FOUND
014246  012703  000004                  MOV     #4,R3            ;NUMBER OF LOCATIONS TO CHECK
014252  022125                  2$:     CMP     (R1)+,(R5)+      ;COMPARE THE BUFFER AGAINST THE PATTERN
014254  001366                          BNE     1$               ;BR IF NOT EQUAL, TRY NEXT PATTERN
014256  005303                          DEC     R3               ;FINISHED CHECKING?
014260  001374                          BNE     2$               ;BR IF NOT FINISHED
014262  062704  002762                  ADD     #STNDAT,R4       ;MAKE PATTERN ADDRESS ABSOLUTE
014266  000403                          BR      4$               ;EXIT
014270  062766  000002  000002  3$:     ADD     #2,2(SP)         ;INCREMENT RETURN ADDRESS
014276  012601                  4$:     MOV     (SP)+,R1         ;RESTORE R1
014300  000207                          RTS     PC               ;RETURN

                                ;USE ECC TO CORRECT THE DATA ERROR

014302  016037  000240  001322  ECC:    MOV     $RPBA(R0),ECSEC  ;ADDRESS OF LAST LOCN XFERED
014310  016046  000236                  MOV     $RPWC(R0),-(SP)  ;ACT WORDS XFERED (2'S COMP)
014314  066016  000020                  ADD     $WRDL(R0),(SP)   ;ADD WORDS REQUESTED
014320  005046                          CLR     -(SP)            ;CLEAR NEXT STACK LOCN
014322  016046  000022                  MOV     $SSEC(R0),-(SP)  ;SECTOR SIZE
014326  004737  027054                  JSR     PC,LINKDV        ;DIVIDE WORDS XFERED BY SECTOR SIZE
014332  005716                          TST     (SP)             ;PARTIAL SECTOR XFERED ?
014334  001413                          BEQ     1$               ;BR IF NOT
014336  006316                          ASL     (SP)             ;CONVERT INTO NUMBER OF BYTES
014340  161637  001322                  SUB     (SP),ECSEC       ;SUBTRACT SECTOR RESIDUE
014344  126027  000024  000005          CMPB    $CODE(R0),#5     ;WAS OP READ HEAD & DATA
014352  001007                          BNE     2$               ;BR IF NOT
014354  062737  000010  001322          ADD     #8.,ECSEC        ;ADD HEADER SIZE (IN BYTES) BACK IN
014362  000403                          BR      2$               ;GO ADJUST THE STACK POINTER
014364  162737  001000  001322  1$:     SUB     #1000,ECSEC      ;SUBTRACT SECTOR DATA FIELD SIZE (IN BYTES)
014372  062706  000004          2$:     ADD     #4,SP            ;ADJUST THE STACK POINTER
014376  016037  000300  001320          MOV     $RPEC1(R0),ECBIT  ;ECC POSITION COUNT
014404  005337  001320                  DEC     ECBIT            ;ADJUST THE POSITION COUNT
014410  013737  001320  001330          MOV     ECBIT,ECWRD      ;LOAD THE WORD COUNT LOCATION
014416  042737  177760  001320          BIC     #^C17,ECBIT      ;SAVE THE BIT OFFSET COUNT
014424  042737  000017  001330          BIC     #17,ECWRD        ;CLEAR THE BIT OFFSET
014432  006237  001330                  ASR     ECWRD            ;CHANGE TO BYTE COUNT
014436  006237  001330                  ASR     ECWRD            ;CHANGE TO BYTE COUNT
014442  006237  001330                  ASR     ECWRD            ;CHANGE TO BYTE COUNT
014446  104414  051571                  DISPLY  ,LIN10A          ;'ERROR BURST BEGINS AT '
014452  013746  001330                  MOV     ECWRD,-(SP)      ;PUT THE WORD COUNT ON THE STACK
014456  006216                          ASR     (SP)             ;CONVERT TO WORD COUNT FOR MESSAGE
014460  004737  027770                  JSR     PC,$SB2D         ;CONVERT THE WORD COUNT
014464  004737  027370                  JSR     PC,$SUPRS        ;PRINT IT
014470  104414  051625                  DISPLY  ,LIN10B          ;' IN DATA FIELD OF ERROR SECTOR'
014474  063737  001322  001330          ADD     ECSEC,ECWRD      ;FIND THE BEGINNING OF THE ERROR BURST
014502  026037  000240  001330          CMP     $RPBA(R0),ECWRD  ;SEE IF BURST WAS IN DATA READ
014510  101002                          BHI     .+6              ;BR IF IN DATA READ
014512  000137  015030                  JMP     ECC2             ;NOT IN DATA READ - REPORT IT
014516  016037  000302  001324          MOV     $RPEC2(R0),ECMSK0  ;GET THE ERROR MASK
014524  005037  001326                  CLR     ECMSK1           ;CLEAR THE UPPER MASK WORD
014530  005737  001320          3$:     TST     ECBIT            ;BIT OFFSET EQUAL ZERO
014534  001407                          BEQ     4$               ;BR IF IT IS
014536  005337  001320                  DEC     ECBIT            ;DECREMENT THE BIT OFFSET COUNT
014542  006337  001324                  ASL     ECMSK0           ;SHIFT THE ERROR MASK
014546  006137  001326                  ROL     ECMSK1           ;SHIFT THE LOWER INTO THE UPPER
014552  000766                          BR      3$               ;CONTINUE THE SHIFT
```

```
4358  014554  017737  164550  001334  4$:     MOV     @ECWRD.ECBADO   ;SAVE THE INCORRECT WORD
4359  014562  005037  001336                  CLR     ECWRD1          ;CLEAR SECOND INCORRECT WORD ADDRESS
4360  014566  013746  001324                  MOV     ECMSK0,-(SP)    ;PUT LOWER MASK ON STACK
4361  014572  047716  164532                  BIC     @ECWRD,(SP)     ;CLEAR ERRONEOUS ONE BITS FROM MASK
4362  014576  043777  001324  164524          BIC     ECMSK0,@ECWRD   ;CLEAR ERRONEOUS ONE BITS FROM BAD WORD
4363  014604  052677  164520                  BIS     (SP)+,@ECWRD    ;SET DROPPED BITS
4364  014610  005737  001326                  TST     ECMSK1          ;DOES BURST GO INTO NEXT WORD ?
4365  014614  001431                          BEQ     ECC1            ;BR IF BURST ONLY IN ONE WORD
4366  014616  013737  001330  001336          MOV     ECWRD,ECWRD1    ;DUPLICATE ADDRESS
4367  014624  062737  000002  001336          ADD     #2,ECWRD1       ;INCREMENT ERROR ADDRESS
4368  014632  026037  000240  001336          CMP     $RPBA(R0),ECWRD1  ;IS NEXT WORD IN THE BUFFER
4369  014640  101003                          BHI     5$              ;BR IF IT IS
4370  014642  005037  001336                  CLR     ECWRD1          ;CLEAR 2ND WORD ADDRESS
4371  014646  000414                          BR      ECC1            ;PRINT WORD CORRECTED
4372  014650  017737  164462  001342  5$:     MOV     @ECWRD1,ECBAD1  ;SAVE THE SECOND BAD WORD
4373  014656  013746  001326                  MOV     ECMSK1,-(SP)    ;PUT THE UPPER MASK ON THE STACK
4374  014662  047716  164450                  BIC     @ECWRD1,(SP)    ;CLEAR ERRONEOUS ONE BITS FROM UPPER MASK
4375  014666  043777  001326  164442          BIC     ECMSK1,@ECWRD1  ;CLEAR ERRONEOUS ONE BITS FROM DATA WORD
4376  014674  052677  164436                  BIS     (SP)+,@ECWRD1   ;SET DROPPED BITS
4377  014700  104414  051773          ECC1:   DISPLY  ,LIN10H         ;HEADER
4378  014704  013746  001330                  MOV     ECWRD,-(SP)     ;PUT ECWRD ON THE STACK
4379  014710  004737  022242                  JSR     PC,LINOCT       ;TYPE ECWRD
4380  014714  104414  052155                  DISPLY  ,LINSP          ;SPACES
4381  014720  013746  001334                  MOV     ECBADO,-(SP)    ;PUT ECBADO ON THE STACK
4382  014724  004737  022242                  JSR     PC,LINOCT       ;TYPE ECBADO
4383  014730  104414  052155                  DISPLY  ,LINSP          ;SPACES
4384  014734  017746  164370                  MOV     @ECWRD,-(SP)    ;PUT @ECWRD ON THE STACK
4385  014740  004737  022242                  JSR     PC,LINOCT       ;TYPE @ECWRD
4386  014744  104414  052155                  DISPLY  ,LINSP          ;SPACES
4387  014750  005737  001336                  TST     ECWRD1          ;PRINT THE NEXT WORD ?
4388  014754  001427                          BEQ     ECCX            ;BR IF NOT
4389  014756  104414  001165                  DISPLY  ,$CRLF          ;CR-LF
4390  014762  013746  001336                  MOV     ECWRD1,-(SP)    ;PUT ECWRD1 ON THE STACK
4391  014766  004737  022242                  JSR     PC,LINOCT       ;TYPE ECWRD1
4392  014772  104414  052155                  DISPLY  ,LINSP          ;SPACES
4393  014776  013746  001342                  MOV     ECBAD1,-(SP)    ;PUT ECBAD1 ON THE STACK
4394  015002  004737  022242                  JSR     PC,LINOCT       ;TYPE ECBAD1
4395  015006  104414  052155                  DISPLY  ,LINSP          ;SPACES
4396  015012  017746  164320                  MOV     @ECWRD1,-(SP)   ;PUT @ECWRD1 ON THE STACK
4397  015016  004737  022242                  JSR     PC,LINOCT       ;TYPE @ECWRD1
4398  015022  104414  052155                  DISPLY  ,LINSP          ;SPACES
4399  015026  000402                          BR      ECCX            ;EXIT
4400  015030  104414  051666          ECC2:   DISPLY  ,LIN10C         ;ERROR BURST WAS NOT TRANSFERED TO MEMORY
4401  015034  104414  001165          ECCX:   DISPLY  ,$CRLF          ;CR-LF
4402  015040  000207                          RTS     PC              ;RETURN
4403
4404                                  ;ROUTINE TO DISPLAY THE SECTOR WHICH GAVE THE HARD ERROR
4405
4406  015042  032777  000010  164070  PRTBAD: BIT     #SW3,@SWR       ;PRINT THE BAD SECTOR ?
4407  015050  001460                          BEQ     6$              ;BR IF NOT
4408  015052  016001  000240                  MOV     $RPBA(R0),R1    ;PUT THE END ADDRESS INTO R1
4409  015056  016046  000020                  MOV     $WRDL(R0),-(SP) ;FIND THE BEGINNING OF THE SECTOR
4410  015062  066016  000236                  ADD     $RPWC(R0),(SP)  ;SUBTRACT THE WORDS NOT TRANSFERED
4411  015066  005046                          CLR     -(SP)           ;MAKE THE UPPER DIVIDEND 0
4412  015070  016046  000022                  MOV     $SSEC(R0),-(SP)  ;DIVDE THE WORDS TRANSFERED BY THE SECTOR SIZE
4413  015074  004737  027054                  JSR     PC,LINKDV       ;DIVIDE
```

```
4414  015100  005716                          TST     (SP)            ;REMANDER = 0 ?
4415  015102  001403                          BEQ     1$              ;BR IF IT IS - COMPLETE SECTOR TRANSFERED
4416  015104  006316                          ASL     (SP)            ;CONVERT THE RESIDUAL SECTOR SIZE INTO BYTE COUNT
4417  015106  161601                          SUB     (SP),R1         ;SUBTRACT IT FROM THE END ADDRESS
4418  015110  000410                          BR      2$              ;FINISH THE SIZING
4419  015112  162701  001000          1$:     SUB     #1000,R1        ;SUBTRACT FULL SECTOR SIZE FROM END ADDR
4420  015116  126027  000024  000005        CMPB    $CODE(R0),#5    ;WAS OPERATION READ HEADER & DATA ?
4421  015124  001002                          BNE     2$              ;BR IF NOT
4422  015126  162701  000010                  SUB     #10,R1          ;SUBTRACK HEADER SIZE FROM ADDR
4423  015132  062706  000004          2$:     ADD     #4,SP           ;RESTORE THE STACK POINTER
4424  015136  104414  052060                  DISPLY  ,LIN11H         ;PRINT THE HEADER
4425  015142  012702  000007          3$:     MOV     #7,R2           ;R2 CONTAINS THE WORDS/LINE COUNT
4426  015146  010146                          MOV     R1,-(SP)        ;PUT THE ADDRESS ON THE STACK
4427  015150  004737  022242                  JSR     PC,LINOCT       ;TYPE THE ADDRESS
4428  015154  020160  000240          4$:     CMP     R1,$RPBA(R0)    ;PRINTED ALL THE SECTOR ?
4429  015160  001412                          BEQ     5$              ;BR IF ALL PRINTED
4430  015162  104414  052155                  DISPLY  ,LINSP          ;SPACES
4431  015166  012146                          MOV     (R1)+,-(SP)     ;PUT THE DATA ON THE STACK
4432  015170  004737  022242                  JSR     PC,LINOCT       ;TYPE THE DATA
4433  015174  005302                          DEC     R2              ;DECREMENT THE HORIZONTAL COUNT
4434  015176  001366                          BNE     4$              ;BR IF NOT AT THE END OF THE LINE
4435  015200  104414  001165                  DISPLY  ,$CRLF          ;CR-LF
4436  015204  000756                          BR      3$              ;RESTORE THE WORDS/LINE COUNT
4437  015206  104414  001165          5$:     DISPLY  ,$CRLF          ;PRINT WHAT REMAINS IN THE BUFFER
4438  015212  000207                  6$:     RTS     PC              ;RETURN
4439
4440                                  ;ROUTINE TO DO AN RTC - DRIVE SELECTED IN R0
4441                                  ;CALL:
4442                                  ;       MOV     #DPB,R0         ;DPB ADDRESS
4443                                  ;       JSR     PC,RTNCTR
4444                                  ;       RETURN
4445
```

```
                                ;DRIVE # TO THE GENERAL DPB
                                ;COMMAND CODE
                                ;DRIVER ENTRANCE
                                ;DPB ADDRESS FOR ORDER
                                ;DRIVER DIDN'T ACCEPT ORDER
                                ;RETURN

                                ;ROUTINE TO DO A RECALIBRATE - DRIVE SELECTED IN R0
                                ;CALL:
                                ;       MOV     #DPB,R0         ;DPB ADDRESS
                                ;       JSR     PC,RECALT
                                ;       RETURN
                                ;
                                ;OR
                                ;
                                ;       MOV     #DPB,R0         ;DPB ADDRESS
                                ;       MOVB    #DRIVE,GENDPB   ;DRIVE ADDRESS
                                ;       JSR     PC,RECALT0
                                ;       RETURN
                                ;
                                ;DRIVE # TO THE GENERAL DPB
                                ;RELCALIBRATE COMMAND
                                ;DRIVER ENTRANCE
                                ;DPB ADDRESS FOR ORDER
                                ;DRIVER DIDN'T ACCEPT THE ORDER
                                        ;SEE IF FINISHED
                                ;BR IF NOT FINISHED
                                ;RETURN
                                ;
                                ;OFFSET THE DRIVE IN R0 (OFFSET CODE PRELOADED INTO 'RPOF')
                                ;CALL:
                                ;       MOVB    #OFFSET,GENDPB+$FMT  ;OFFSET CODE
                                ;       MOV     #DPB,R0         ;DPB ADDRESS
                                ;       JSR     PC,OFFST
                                ;       RETURN
                                ;
                                ;DRIVE # TO GENERAL DPB
                                ;COMMAND
                                ;DRIVER ENTRANCE
                                ;DPB ADDRESS FOR ORDER
                                ;DRIVER DIDN'T ACCEPT ORDER
                                ;
                                ;UTILITY READ HEADER ROUTINE
                                ;CALL:
                                ;       MOV     #DPB,R0         ;DPB ADDRESS
                                ;       MOV     #SECTOR,-(SP)   ;SECTOR ADDRESS
                                ;       MOV     #TRACK,-(SP)    ;TRACK ADDRESS
                                ;       JSR     PC,READDR
                                ;       RETURN
                                ;
                                ;TRACK ADDRESS
                                ;SECTOR ADDRESS
                                ;DRIVE NUMBER
                                ;CYLINDER ADDRESS
                                ;COMMAND
```

```
4502  015352  004037  034120          1$:     JSR     R0,RP04         ;DRIVER ENTRANCE
4503  015356  044730                          GENDPB                  ;DPB ADDRESS FOR ORDER
4504  015360  000774                          BR      1$              ;DRIVER DIDN'T ACCEPT COMMAND
4505  015362  005737  044746          2$:     TST     GENDPB+$STATUS  ;FINISHED?
4506  015366  001775                          BEQ     2$              ;BR IF NOT
4507  015370  012666  000002                  MOV     (SP)+,2(SP)     ;ADJUST STACK FOR RETURN
4508  015374  005726                          TST     (SP)+
4509  015376  000207                          RTS     PC              ;RETURN
4510
4511                                  ;RETRY THE PRESENT OPERATION
4512                                  ;CALL:
4513                                  ;       MOV     #COUNT,RETRY    ;RETRY COUNT
4514                                  ;       JSR     PC,$RETRY
4515                                  ;       RETURN1                 ;RETRY UNSUCESSFUL
4516                                  ;       RETURN2                 ;SUCESSFUL RETRY
4517                                  ;                               ;NOTE: IF A DIFFERENT ERROR OCCURS DURING
4518                                  ;                               ;RETRY, THE ROUTINE EXITS TO 'ERPRC1'
4519
4520  015400  004737  016370          $RETRY: JSR     PC,GODRIV       ;RE-START ORDER
4521  015404  005760  000016          1$:     TST     $STATUS(R0)     ;ORDER FINISHED?
4522  015410  001775                          BEQ     1$              ;BR IF NOT
4523  015412  100405                          BMI     2$              ;BR IF ERROR
4524  015414  105237  001253                  INCB    RETRY+1         ;INCREMENT RETRY COUNT
4525  015420  062716  000002                  ADD     #2,(SP)         ;INCREMENT RETURN
4526  015424  000425                          BR      5$              ;GO TO EXIT
4527  015426  032760  000200  000016  2$:     BIT     #BIT7,$STATUS(R0)  ;DID ORDER TERMINATE NORMALLY ?
4528  015434  001430                          BEQ     7$              ;BR IF NOT
4529  015436  005737  001250                  TST     MASK            ;IS ERROR MASK 0 ?
4530  015442  001004                          BNE     3$              ;BR IF NOT
4531  015444  005760  000250                  TST     $RPER1(R0)      ;MAKE SURE THAT THE DRIVE ERROR REG IS CLEAR
4532  015450  001014                          BNE     6$              ;BR IF NOT
4533  015452  000404                          BR      4$              ;CONTINUE RETRY
4534  015454  033760  001250  000250  3$:     BIT     MASK,$RPER1(R0) ;SAME ERROR?
4535  015462  001407                          BEQ     6$              ;BR IF NOT
4536  015464  105237  001253          4$:     INCB    RETRY+1         ;INCREMENT RETRY COUNT
4537  015470  123737  001252  001253          CMPB    RETRY,RETRY+1   ;DONE ?
4538  015476  001340                          BNE     $RETRY          ;BR IF NOT DONE
4539  015500  000207                  5$:     RTS     PC              ;RETURN
4540  015502  004737  022230          6$:     JSR     PC,LINE8        ;REPORT DIFFERENT ERROR
4541  015506  004737  021762                  JSR     PC,LINE7        ;PRINT LINE 7
4542  015512  005726                          TST     (SP)+           ;ADJUST STACK POINTER FOR DIRECT RETURN
4543  015514  000207                          RTS     PC              ;RETURN
4544  015516  104414  051330          7$:     DISPLY  ,LIN8M          ;'DIFFERENT ERROR DURING RETRY'
4545  015522  000137  006644                  JMP     ERPRC1          ;REPORT THE ERROR
4546
4547                                  ;ROUTINE TO UPDATE THE PERFORMANCE SUMMARY STATISTICS
4548                                  ;CALL:
4549                                  ;       MOV     #DPB,R0         ;DPB ADDRESS
4550                                  ;       JSR     PC,STATIS
4551                                  ;       RETURN
4552
4553  015526  032760  000300  000016  STATIS: BIT     #BIT07!BIT06,$STATUS(R0) ;CHECK FOR DATA TERMINATION
4554  015534  001454                          BEQ     3$              ;BR IF NOT DATA TERMINATION
4555  015536  016037  000240  015670          MOV     $RPBA(R0),FACTOR  ;STORE THE FINAL BUFFER ADDRESS
4556  015544  166037  000006  015670          SUB     $BUF(R0),FACTOR ;SUBTRACT THE INITIAL ADDRESS
4557  015552  001434                          BEQ     2$              ;BR IF NO DATA TRANSFER
```

```
 4558  015554  006237  015670                ASR     FACTOR          ;CONVERT TO A WORD COUNT
 4559  015560  063760  015670  000046        ADD     FACTOR,$TRANS(R0)  ;UPDATE WORD COUNT
 4560  015566  005560  000050                ADC     $TRANS+2(R0)    ;ADD ANY CARRY
 4561  015572  132760  000002  000024        BITB    #BIT01,$CODE(R0)  ;SEE IF ORDER READ OR WRITE
 4562  015600  001021                        BNE     2$              ;BRANCH IF ORDER WRITE
 4563  015602  005737  001424                TST     AUTOCK          ;AUTO WRITE CHECKS BEING PERFORMED
 4564  015606  001411                        BEQ     1$              ;BR IF NOT
 4565  015610  126027  000024  000001        CMPB    $CODE(R0),#1    ;PRESENT OPERATION AN AUTOMATIC WRITE CHECK ?
 4566  015616  101005                        BHI     1$              ;BR IF NOT
 4567  015620  066060  000020  000046        ADD     $WRDL(R0),$TRANS(R0)  ;ADD WORDS WRITTEN
 4568  015626  005560  000050                ADC     $TRANS+2(R0)    ;ADD A CARRY
 4569  015632  063760  015670  000052  1$:   ADD     FACTOR,$READ(R0)  ;UPDATE THE READ WORD COUNT
 4570  015640  005560  000054                ADC     $READ+2(R0)     ;ADD ANY CARRY
 4571  015644  026060  000012  000270  2$:   CMP     $CYL(R0),$RPCA(R0)  ;DID MID-TRANSFER SEEK OCCUR
 4572  015652  001405                        BEQ     3$              ;BR IF NOT
 4573  015654  062760  000001  000042        ADD     #1,$POSIT(R0)   ;INCREMENT SEEK COUNT
 4574  015662  005560  000044                ADC     $POSIT+2(R0)    ;ADD CARRY TO UPPER WORD
 4575  015666  000207            3$:         RTS     PC
 4576
 4577  015670  000000            FACTOR: .WORD   0               ;USED FOR WORDS TRANSFERED
 4578
 4579                            ;ROUTINE TO GET A BUFFER
 4580                            ;CALL:
 4581                            ;       MOV     #DPB,R0         ;DPB ADDRESS
 4582                            ;       CLR     -(SP)           ;CLEAR THE STACK
 4583                            ;       JSR     PC,GETBUF
 4584                            ;       RETURN                  ;BUFFER ADDRESS WILL BE ON THE STACK
 4585                            ;                               ;STACK WILL BE ZERO IF NO BUFFER AVAILABLE
 4586
 4587  015672  010146            GETBUF: MOV     R1,-(SP)        ;SAVE R1
 4588  015674  010246                    MOV     R2,-(SP)        ;SAVE R2
 4589  015676  010346                    MOV     R3,-(SP)        ;SAVE R3
 4590  015700  013702  001616            MOV     BUFTBL,R2       ;NUMBER OF SEPARATE BUFFERS
 4591  015704  001444                    BEQ     6$              ;BR IF NONE AVAILABLE
 4592  015706  012701  001620            MOV     #BUFTBL+2,R1    ;FIRST ADDRESS OF ALLOCATION TABLE
 4593  015712  026061  000020  000002  1$:   CMP     $WRDL(R0),2(R1)  ;SEE IF THERE IS A BLOCK LARGE ENOUGH
 4594  015720  101405                        BLOS    3$              ;BRANCH IF IT IS
 4595  015722  005302                        DEC     R2              ;DECREMENT TABLE COUNT
 4596  015724  001434                        BEQ     6$              ;BR IF THROUGH TABLE
 4597  015726  062701  000004                ADD     #4,R1           ;INCREMENT TABLE POINTER
 4598  015732  000767                        BR      1$              ;CONTINUE LOOKING
 4599  015734  011166  000010        3$:     MOV     (R1),10(SP)     ;BUFFER ADDRESS TO STACK
 4600  015740  166061  000020  000002        SUB     $WRDL(R0),2(R1)  ;ADJUST BUFFER SIZE
 4601  015746  001407                        BEQ     4$              ;BR IF DIFFERENCE IS ZERO
 4602  015750  006360  000020                ASL     $WRDL(R0)       ;CONVERT # WORDS TO BYTES
 4603  015754  066011  000020                ADD     $WRDL(R0),(R1)  ;MAKE NEW STARTING ADDRESS
 4604  015760  006260  000020                ASR     $WRDL(R0)       ;RETURN # BYTES TO WORDS
 4605  015764  000414                        BR      6$              ;RETURN
 4606  015766  005337  001616        4$:     DEC     BUFTBL          ;DECREMENT ENTRIES COUNT
 4607  015772  001411                        BEQ     6$              ;BR IF ALLOCATION TABLE EMPTY
 4608  015774  005302                        DEC     R2              ;DECREMENT TABLE COUNT
 4609  015776  001407                        BEQ     6$              ;BR IF ITEM WERE LAST ENTRY
 4610  016000  010103                        MOV     R1,R3           ;MOVE TABLE POINTER
 4611  016002  062703  000004                ADD     #4,R3           ;POINT TO NEXT ENTRY
 4612  016006  012321            5$:         MOV     (R3)+,(R1)+     ;MOVE ITEMS
 4613  016010  012321                        MOV     (R3)+,(R1)+
```

```
4614  016012  005302                        DEC     R2              ;DECREMENT TABLE COUNT
4615  016014  001374                        BNE     5$              ;CONTINUE IF NOT AT END OF TABLE
4616  016016  012603                6$:     MOV     (SP)+,R3        ;RESTORE R3
4617  016020  012602                        MOV     (SP)+,R2        ;RESTORE R2
4618  016022  012601                        MOV     (SP)+,R1        ;RESTORE R1
4619  016024  000207                        RTS     PC              ;RETURN
4620
4621
4622                                  ;ROUTINE TO PUT BUFFER BACK IN TABLE
4623                                  ;CALL:
4624                                  ;       MOV     #DPB,R0         ;DPB ADDRESS
4625                                  ;       JSR     PC,RELBUF
4626                                  ;       RETURN
4627
4628  016026  010146                RELBUF: MOV     R1,-(SP)        ;SAVE R1
4629  016030  012701  001620                MOV     #BUFTBL+2,R1    ;BEGINNING OF TABLE
4630  016034  013702  001616                MOV     BUFTBL,R2       ;ENTRY COUNT
4631  016040  001424                        BEQ     2$              ;BR IF EMPTY TABLE
4632  016042  016003  000020                MOV     $WRDL(R0),R3    ;TRIAL ADDRESS
4633  016046  006303                        ASL     R3              ;CHANGE TO BYTE COUNT
4634  016050  066003  000006                ADD     $BUF(R0),R3     ;ADDRESS OF HIGHER ADJACENT BLOCK
4635  016054  021103                1$:     CMP     (R1),R3         ;UPPER ADJACENT BLOCK
4636  016056  001424                        BEQ     4$              ;BR IF YES
4637  016060  062701  000004                ADD     #4,R1           ;INCREMENT POINTER
4638  016064  005302                        DEC     R2              ;DECREMENT ENTRY COUNT
4639  016066  001372                        BNE     1$              ;CONTINUE SEARCHING
4640  016070  016011  000006                MOV     $BUF(R0),(R1)   ;PUT THE BUFFER BLOCK INTO THE TABLE
4641  016074  016061  000020  000002        MOV     $WRDL(R0),2(R1) ;BLOCK SIZE
4642  016102  005237  001616                INC     BUFTBL          ;INCREMENT ENTRY COUNT
4643  016106  005202                        INC     R2              ;INCREMENT R2 FOR USE LATER
4644  016110  000414                        BR      5$              ;SEE IF A LOWER ADJACENT BLOCK IS IN THE TABLE
4645  016112  016021  000006        2$:     MOV     $BUF(R0),(R1)+  ;BLOCK ADDRESS TO TABLE
4646  016116  016021  000020                MOV     $WRDL(R0),(R1)+ ;SIZE TO TABLE
4647  016122  005237  001616                INC     BUFTBL          ;INCREMENT ENTRY COUNT
4648  016126  000443                        BR      10$             ;EXIT
4649  016130  016011  000006        4$:     MOV     $BUF(R0),(R1)   ;RELEASED BUFFER IS LOWER ADJACENT
4650  016134  066061  000020  000002        ADD     $WRDL(R0),2(R1) ;INCREMENTED SIZE
4651  016142  010246                5$:     MOV     R2,-(SP)        ;SAVE R2
4652  016144  013702  001616                MOV     BUFTBL,R2       ;ENTRY COUNT
4653  016150  012705  001620                MOV     #BUFTBL+2,R5    ;BEGINNING OF TABLE
4654  016154  016504  000002        6$:     MOV     2(R5),R4        ;BLOCK SIZE (IN WORDS)
4655  016160  006304                        ASL     R4              ;CHANGE TO BYTE COUNT
4656  016162  061504                        ADD     (R5),R4         ;ADD BLOCK BEGINNING ADDRESS
4657  016164  020411                        CMP     R4,(R1)         ;R1 STILL POINTS TO INSERTED ENTRY
4658  016166  001406                        BEQ     8$              ;LOWER ADJACENT IN TABLE
4659  016170  062705  000004                ADD     #4,R5           ;INCREMENT POINTER
4660  016174  005302                        DEC     R2              ;DECREMENT ENTRY COUNT
4661  016176  001366                        BNE     6$              ;CONTINUE LOOKING
4662  016200  005726                        TST     (SP)+           ;RESTORE STACK POINTER
4663  016202  000415                        BR      10$             ;END
4664  016204  012602                8$:     MOV     (SP)+,R2        ;RESTORE R2
4665  016206  066165  000002  000002        ADD     2(R1),2(R5)     ;INCREMENT LOWER BLOCK LENGTH
4666  016214  005337  001616                DEC     BUFTBL          ;DECREMENT ENTRY COUNT
4667  016220  010105                        MOV     R1,R5           ;GET READY TO COMPRESS
4668  016222  062705  000004                ADD     #4,R5           ;INCREMENT TO NEXT ENTRY
4669  016226  012521                9$:     MOV     (R5)+,(R1)+     ;COMPRESS TABLE
```

```
 4670  016230  012521                        MOV     (R5)+,(R1)+     ;MOVE SIZE FIELD DOWN
 4671  016232  005302                        DEC     R2              ;DECREMENT ENTRY COUNT
 4672  016234  001374                        BNE     9$              ;BR IF NOT FINISHED
 4673  016236  012601                10$:    MOV     (SP)+,R1        ;RESTORE R1
 4674  016240  000207                        RTS     PC              ;RETURN
 4675
 4676
 4677                                ;FILL THE ASSIGNED BUFFER (IF WRITE OR WRITE CHECK ORDER)
 4678                                ;CALL:
 4679                                ;       MOV     #DPB,R0         ;DPB ADDRESS
 4680                                ;       MOV     #BUFADR,$BUF(R0)  ;LOAD BUFFER ADDRESS INTO THE DPB
 4681                                ;       MOVB    #PATTERN,$PATTC(R0)  ;PATTERN CODE
 4682                                ;       JSR     PC,FILBUF
 4683                                ;       RETURN
 4684
 4685  016242  104412                FILBUF: SAVREG                  ;SAVE THE REGISTERS
 4686  016244  132760  000004  000024        BITB    #BIT02,$CODE(R0)  ;SEE IF READ ORDER
 4687  016252  001044                        BNE     4$              ;BR IF READ
 4688  016254  016001  000006        1$:     MOV     $BUF(R0),R1     ;BUFFER ADDRESS
 4689  016260  016002  000020                MOV     $WRDL(R0),R2    ;POSITIVE WORD COUNT
 4690  016264  132760  000001  000024        BITB    #BIT00,$CODE(R0)  ;SEE IF WRITE HEADER TYPE ORDER
 4691  016272  001413                        BEQ     2$              ;BR IF NOT
 4692  016274  016011  000012                MOV     $CYL(R0),(R1)   ;CYLINDER ADDRESS
 4693  016300  052721  010000                BIS     #BIT12,(R1)+    ;SET FMT22 BIT
 4694  016304  016021  000010                MOV     $SEC(R0),(R1)+  ;MOVE SECTOR & TRACK
 4695  016310  005021                        CLR     (R1)+           ;CLEAR FIRST KEY WORD
 4696  016312  005021                        CLR     (R1)+           ;CLEAR THE SECOND
 4697  016314  162702  000004                SUB     #4,R2           ;ADJUST THE WORD COUNT
 4698  016320  003421                        BLE     4$              ;BR IF END OF PATTERN
 4699  016322  005004                2$:     CLR     R4              ;CLEAR R4
 4700  016324  116004  000030                MOVB    $PATTC(R0),R4   ;RELATIVE PATTERN ADDRESS
 4701  016330  016405  002762                MOV     STNDAT(R4),R5   ;PATTERN ADDRESS
 4702  016334  012703  000020                MOV     #20,R3          ;PATTERN COUNT
 4703  016340  012521                3$:     MOV     (R5)+,(R1)+     ;MOVE THE PATTERN INTO THE BUFFER
 4704  016342  005302                        DEC     R2              ;DECREMENT THE WORD COUNT
 4705  016344  001407                        BEQ     4$              ;BR IF DONE (WORD COUNT = 0)
 4706  016346  005303                        DEC     R3              ;DECREMENT THE PATTERN COUNT
 4707  016350  001373                        BNE     3$              ;BR IF MORE PATTERN
 4708  016352  012703  000020                MOV     #20,R3          ;RESTORE PATTERN COUNT
 4709  016356  016405  002762                MOV     STNDAT(R4),R5   ;RESTORE THE ADDRESS
 4710  016362  000766                        BR      3$              ;CONTINUE DISTRIBUTING THE PATTERN
 4711  016364  104413                4$:     RESREG                  ;RESTORE THE REGISTERS
 4712  016366  000207                        RTS     PC              ;RETURN
 4713
 4714                                ;START THE ORDER FOR THE DPB IN R0
 4715                                ;CALL:
 4716                                ;       MOV     #DPB,R0         ;DPB ADDRESS
 4717                                ;       JSR     PC,GODRIV
 4718                                ;       RETURN
 4719
 4720  016370  010046                GODRIV: MOV     R0,-(SP)        ;SAVE R0
 4721  016372  010037  016402                MOV     R0,2$           ;CURRENT DPB ADDRESS
 4722  016376  004037  034120        1$:     JSR     R0,RP04         ;CALL THE DRIVE HANDLER
 4723  016402  000000                2$:     .WORD   0               ;DRIVE BLOCK ADDRESS GOES HERE
 4724  016404  000000                        HALT                    ;DRIVER REJECTED REQUEST
 4725  016406  012600                        MOV     (SP)+,R0        ;RESTORE R0
```

```
4726  016410  062760  000001  000036          ADD     #1,$OPERC(R0)   ;INCREMENT THE OPERATION COUNT
4727  016416  005560  000040                  ADC     $OPERC+2(R0)
4728  016422  026060  000034  000012          CMP     $PREVA+2(R0),$CYL(R0)  ;DID ORDER REQUIRE A CYLINDER CHANGE
4729  016430  001405                          BEQ     3$              ;BR IF NOT
4730  016432  062760  000001  000042          ADD     #1,$POSIT(R0)   ;INCREMENT SEEK COUNT
4731  016440  005560  000044                  ADC     $POSIT+2(R0)    ;ADD ANY CARRY
4732  016444  000207                  3$:     RTS     PC
4733
4734                                          ;GENERATE PARAMETERS FOR THE OPERATION
4735                                          ;CALL:
4736                                          ;       MOV     #DPB,R0         ;DPB ADDRESS
4737                                          ;       JSR     PC,SELPAR
4738                                          ;       RETURN
4739
4740  016446  004737  032364          SELPAR: JSR     PC,$RAND        ;CYCLE THE RANDOM NUMBER GENERATOR
4741  016452  032777  000001  162460          BIT     #SWO,@SWR       ;SEE IF SWO SET
4742  016460  001012                          BNE     2$              ;BR IF SET - READ ONLY
4743  016462  012705  000010          1$:     MOV     #10,R5          ;READ/WRITE SELECTION DIVISOR
4744  016466  004737  027026                  JSR     PC,GETREM       ;GET SELECTION VALUE
4745  016472  020537  001422                  CMP     R5,RATIO        ;DETERMINE IF READ OR WRITE
4746  016476  103003                          BHIS    2$              ;BR IF READ
4747  016500  004737  017170                  JSR     PC,RANWRT       ;SELECT A WRITE ORDER
4748  016504  000406                          BR      3$              ;CONTINUE WITH THE SELECTION
4749  016506  013705  032464          2$:     MOV     $LONUM,R5       ;SELECT READ OPERATION CODE
4750  016512  042705  177776                  BIC     #^C1,R5         ;MASK OUT ALL BUT BIT 0
4751  016516  062705  000004                  ADD     #4,R5           ;TABLE OFFSET FOR READ CODE
4752  016522  110560  000074          3$:     MOVB    R5,$NCODE(R0)   ;ORDER SELECTION CODE TO CONTROL BLOCK
4753
4754                                          ;GENERATE A RANDOM SECTOR ADDRESS BETWEEN VALUES 'MINSEC' & 'MAXSEC'
4755
4756  016526  016005  000116          RANSEC: MOV     MAXSEC(R0),R5   ;GET MAXIMUM SECTOR ADDRESS
4757  016532  026005  000120                  CMP     MINSEC(R0),R5   ;'MINSEC' AND 'MAXSEC' THE SAME ?
4758  016536  001417                          BEQ     2$              ;BR IF THEY ARE
4759  016540  166005  000120                  SUB     MINSEC(R0),R5   ;SUBTRACT MINIMUM SECTOR ADDRESS
4760  016544  100002                          BPL     1$              ;BR IF MAX LARGER THAN MIN
4761  016546  062705  000026                  ADD     #22.,R5         ;CORRECT THE NUMBER
4762  016552  005205          1$:     INC     R5              ;INCREMENT DIFFERENCE TO USE AS DIVISOR
4763  016554  004737  027026                  JSR     PC,GETREM       ;GET THE RANDOM AUGMENT
4764  016560  066005  000120                  ADD     MINSEC(R0),R5   ;NEW ADDRESS
4765  016564  020527  000025                  CMP     R5,#21.         ;IS VALUE TOO LARGE ?
4766  016570  101402                          BLOS    2$              ;BR IF NOT
4767  016572  162705  000026                  SUB     #22.,R5         ;CORRECT VALUE
4768  016576  110560  000076          2$:     MOVB    R5,$NSEC(R0)    ;STORE SECTOR ADDRESS IN DPB
4769
4770                                          ;GENERATE A RANDOM TRACK ADDRESS BETWEEN VALUES 'MINTRK' & 'MAXTRK'
4771
4772  016602  016005  000112          RANTRK: MOV     MAXTRK(R0),R5   ;GET MAXIMUM TRACK ADDRESS
4773  016606  026005  000114                  CMP     MINTRK(R0),R5   ;'MINTRK' AND 'MAXTRK' THE SAME ?
4774  016612  001417                          BEQ     2$              ;BR IF THEY ARE
4775  016614  166005  000114                  SUB     MINTRK(R0),R5   ;SUBTRACT MINIMUM TRACK ADDRESS
4776  016620  100002                          BPL     1$              ;BR IF MAX LARGER THAN MIN
4777  016622  062705  000023                  ADD     #19.,R5         ;CORRECT THE NUMBER
4778  016626  005205          1$:     INC     R5              ;INCREMENT DIFFERENCE TO USE AS DIVISOR
4779  016630  004737  027026                  JSR     PC,GETREM       ;GET THE RANDOM AUGMENT
4780  016634  066005  000114                  ADD     MINTRK(R0),R5   ;NEW TRACK ADDRESS
4781  016640  020527  000022                  CMP     R5,#18.         ;IS VALUE TOO LARGE ?
```

```
 4782  016644  101402                        BLOS    2$              ;BR IF NOT
 4783  016646  162705  000023                SUB     #19.,R5         ;CORRECT VALUE
 4784  016652  110560  000077        2$:     MOVB    R5,$NTRK(R0)    ;STORE TRACK ADDRESS IN DPB
 4785
 4786                                        ;GENERATE A RANDOM CYLINDER ADDRESS BETWEEN VALUES 'MINCYL' & 'MAXCYL'
 4787
 4788  016656  012737  000633  001350        MOV     #411.,CYLIMT    ;ASSUME AN RP04/5
 4789  016664  032760  000002  000262        BIT     #BIT01,$RPDT(R0)  ;SEE IF RP06
 4790  016672  001403                        BEQ     RANCYL          ;BR IF NOT
 4791  016674  012737  001457  001350        MOV     #815.,CYLIMT    ;CHANGE CYLINDER LIMIT
 4792  016702  016005  000106        RANCYL: MOV     MAXCYL(R0),R5   ;GET MAXIMUM CYLINDER ADDRESS
 4793  016706  026005  000110                CMP     MINCYL(R0),R5   ;'MINCYL' AND 'MAXCYL' THE SAME ?
 4794  016712  001417                        BEQ     2$              ;BR IF THEY ARE
 4795  016714  166005  000110                SUB     MINCYL(R0),R5   ;SUBTRACT MINIMUM CYLINDER ADDRESS
 4796  016720  100002                        BPL     1$              ;BR IF MAX LARGER THAN MIN
 4797  016722  063705  001350                ADD     CYLIMT,R5       ;CORRECT THE NUMBER
 4798  016726  005205        1$:     INC     R5              ;INCREMENT DIFFERENCE TO USE AS DIVISOR
 4799  016730  004737  027026                JSR     PC,GETREM       ;GET THE RANDOM AUGMENT
 4800  016734  066005  000110                ADD     MINCYL(R0),R5   ;NEW CYLINDER ADDRESS
 4801  016740  023705  001350                CMP     CYLIMT,R5       ;IS VALUE TOO LARGE ?
 4802  016744  101002                        BHI     2$              ;BR IF NOT
 4803  016746  163705  001350                SUB     CYLIMT,R5       ;CORRECT VALUE
 4804  016752  010560  000100        2$:     MOV     R5,$NCYL(R0)    ;STORE CYLINDER ADDRESS IN DPB
 4805  016756  122760  000003  000074        CMPB    #3,$NCODE(R0)   ;WRITE HEADER & DATA ?
 4806  016764  001013                        BNE     RANSIZ          ;BR IF NOT
 4807  016766  012760  000404  000102        MOV     #260.,$NWRDL(R0)  ;CHANGE WORD LENGTH TO 260 FOR WRTHD ORDER
 4808  016774  023727  001404  000404        CMP     MAXDL,#260.     ;CAN A FULL SECTOR BE WRITTEN ?
 4809  017002  103062                        BHIS    RANPAT          ;BR IF IT CAN
 4810  017004  013760  001404  000102        MOV     MAXDL,$NWRDL(R0)  ;CHANGE TRANSFER SIZE
 4811  017012  000456                        BR      RANPAT          ;CONTINUE WITH THE SELECTION
 4812
 4813                                        ;GENERATE A RANDOM BUFFER LENGTH BETWEEN 4 & THE VALUE IN 'MAXDL'
 4814
 4815  017014  013705  001404        RANSIZ: MOV     MAXDL,R5        ;GET BUFFER SIZE
 4816  017020  005737  001420                TST     WCSEL           ;SELECT A RANDOM WORD COUNT ?
 4817  017024  001010                        BNE     1$              ;BR IF NOT
 4818  017026  005205                        INC     R5              ;INCREMENT THE MAXIMUM SIZE
 4819  017030  004737  027026                JSR     PC,GETREM       ;DIVIDE BY MAX VALUE
 4820  017034  005705                        TST     R5              ;IS THE REMAINDER 0 ?
 4821  017036  001003                        BNE     1$              ;NOT 0, CONTINUE
 4822  017040  004737  032364                JSR     PC,$RAND        ;CYCLE THE RANDOM NUMBER GENERATOR
 4823  017044  000763                        BR      RANSIZ          ;TRY AGAIN
 4824  017046  010560  000102        1$:     MOV     R5,$NWRDL(R0)   ;WORD LENGTH TO CONTROL BLOCK
 4825  017052  010546                        MOV     R5,-(SP)        ;NEW WORD LENGTH ON STACK FOR CHECK
 4826  017054  005046                        CLR     -(SP)           ;MAKE UPPER DIVIDEND ZERO
 4827  017056  012746  000400                MOV     #256.,-(SP)     ;SECTOR SIZE IS THE DIVISOR
 4828  017062  132760  000001  000074        BITB    #1,$NCODE(R0)   ;SEE IF NEXT ORDER IS A HEADER ORDER
 4829  017070  001402                        BEQ     2$              ;BR IF NOT
 4830  017072  062716  000004                ADD     #4,(SP)         ;ADD HEADER SIZE TO SECTOR SIZE
 4831  017076  004737  027054        2$:     JSR     PC,LINKDV       ;DIVIDE BUFFER SIZE BY SECTOR SIZE
 4832  017102  012616                        MOV     (SP)+,(SP)      ;MOV REMAINDER UP THE STACK
 4833  017104  021627  000004                CMP     (SP),#4 ;SEE IF REMAINDER LESS THAN 4
 4834  017110  103012                        BHIS    4$              ;BR IF NOT
 4835  017112  005737  001420                TST     WCSEL           ;SELECTING RANDOM TRANSFER SIZES ?
 4836  017116  001403                        BEQ     3$              ;BR IF YES
 4837  017120  161660  000102                SUB     (SP),$NWRDL(R0)  ;ADJUST WORD LENGTH DOWNWARD
```

```
4838  017124  000404                        BR      4$                ;CONTINUE
4839  017126  005726                3$:     TST     (SP)+             ;CORRECT THE STACK POINTER
4840  017130  004737  032364                JSR     PC,$RAND          ;CYCLE THE RANDOM NUMBER GENERATOR
4841  017134  000727                        BR      RANSIZ            ;TRY AGAIN
4842  017136  005726                4$:     TST     (SP)+             ;CORRECT THE STACK POINTER
4843  017140  122760  000002  000074        CMPB    #2,$NCODE(R0)     ;SEE IF WRITE DATA
4844  017146  001004                        BNE     RANXIT            ;BR IF NOT WRITE DATA
4845
4846                                ;GET A RANDOM PATTERN NUMBER
4847
4848  017150  004737  017274        RANPAT: JSR     PC,GETPAT         ;GET PATTERN CODE
4849  017154  110560  000075                MOVB    R5,$NPATC(R0)     ;MOVE PATTERN CODE TO CONTROL BLOCK
4850  017160  012760  177777  000104  RANXIT: MOV     #-1,$NEXT(R0)   ;SET PARAMETERS SELECTED INDICATOR
4851  017166  000207                        RTS     PC                ;RETURN
4852
4853                                ;ROUTINE TO SELECT A WRITE (OR WRITE CHECK) OPERATION
4854
4855  017170  012705  000004        RANWRT: MOV     #4,R5             ;WRITE OPERATION SELECTION DIVISOR
4856  017174  004737  027026                JSR     PC,GETREM         ;GET SELECTION CODE
4857  017200  005737  001424                TST     AUTOCK            ;ARE WRITE CHECK ORDERS TO BE SELECTED
4858                                                                  ;RANDOMLY ?
4859  017204  001403                        BEQ     1$                ;BR IF THEY ARE
4860  017206  152705  000002                BISB    #2,R5             ;SET CODE TO EXCLUDE WRITE CHECK ORDERS
4861  017212  000420                        BR      3$                ;COMPLETE SELECTION
4862  017214  020527  000001        1$:     CMP     R5,#1             ;WRITE CHECK SELECTED ?
4863  017220  101015                        BHI     3$                ;BR IF NOT
4864  017222  132760  000002  000024        BITB    #2,$CODE(R0)      ;PREVIOUS WRITE OPERATION ?
4865  017230  001407                        BEQ     2$                ;BR IF PREVIOUS WAS READ OR WRITE CHECK
4866  017232  116060  000024  000074        MOVB    $CODE(R0),$NCODE(R0)  ;MOVE CODE TO 'NEXT CODE'
4867  017240  142760  000002  000074        BICB    #2,$NCODE(R0)     ;CHANGE WRITE TO WRITE CHECK
4868  017246  000411                        BR      5$                ;EXIT
4869  017250  052705  000002        2$:     BIS     #2,R5             ;CHANGE WRITE CHECK TO WRITE
4870  017254  005737  001416        3$:     TST     FORMAT            ;WRITE HEADER ORDERS ALLOWED ?
4871  017260  001002                        BNE     4$                ;BR IF THEY ARE
4872  017262  042705  000001                BIC     #1,R5             ;ALTER POSSIBLE WRITE HEADER
4873  017266  110560  000074        4$:     MOVB    R5,$NCODE(R0)     ;SETUP 'NEXT' CODE
4874  017272  000207                5$:     RTS     PC                ;RETURN
4875
4876                                ;ROUTINE TO SELECT A PATTERN
4877
4878  017274  012705  000020        GETPAT: MOV     #20,R5            ;SELECT PATTERN
4879  017300  004737  027026                JSR     PC,GETREM         ;GET CODE
4880  017304  005705                        TST     R5                ;WAS PATTERN ZERO SELECTED ?
4881  017306  001003                        BNE     1$                ;BR IF NOT ZERO
4882  017310  004737  032364                JSR     PC,$RAND          ;CYCLE THE RANDOM NUMBER GENERATOR
4883  017314  000767                        BR      GETPAT            ;TRY AGAIN
4884  017316  006305                1$:     ASL     R5                ;MAKE CODE INTO TABLE INDEX
4885  017320  000207                        RTS     PC
4886
4887                                ;ROUTINE TO GET THE PREVIOUSLY SELECTED PARAMETER VALUES
4888                                ;CALL:
4889                                ;       MOV     #DPB,R0           ;DPB ADDRESS
4890                                ;       JSR     PC,SELPAR         ;SELECT THE PARAMETERS
4891                                ;       JSR     PC,GETPAR
4892                                ;       RETURN
4893
```

```
 4894  017322  010546                    GETPAR: MOV     R5,-(SP)        ;SAVE R5
 4895  017324  116060  000234  000027            MOVB    $RPCS1(R0),$PREVO(R0)  ;SAVE CURRENT PARAMETERS
 4896  017332  032760  000006  000074            BIT     #6,$NCODE(R0)   ;SEE IF NEXT OPERATION IS READ OR WRITE
 4897  017340  001007                            BNE     1$              ;BR IF EITHER
 4898  017342  016060  000012  000034            MOV     $CYL(R0),$PREVA+2(R0)  ;SAVE STARTING CYLINDER
 4899  017350  016060  000010  000032            MOV     $SEC(R0),$PREVA(R0)  ;SAVE STARTING SECTOR AND TRACK
 4900  017356  000411                            BR      2$
 4901  017360  004737  022314              1$:   JSR     PC,READDR       ;GET THE DECREMENTED SECTOR AND TRACK ADDRESSES
 4902  017364  112660  000033                    MOVB    (SP)+,$PREVA+1(R0)  ;TRACK ADDRESS
 4903  017370  112660  000032                    MOVB    (SP)+,$PREVA(R0)  ;SECTOR ADDRESS
 4904  017374  016060  000272  000034            MOV     $RPCC(R0),$PREVA+2(R0)  ;CURRENT CYLINDER
 4905  017402  032777  000100  161530      2$:   BIT     #SWD6,@SWR      ;SWITCH 6 SET ?
 4906  017410  001043                            BNE     3$              ;BR IF SET
 4907  017412  116060  000074  000024            MOVB    $NCODE(R0),$CODE(R0)  ;LOGICAL CODE FOR OPERATION
 4908  017420  116005  000074                    MOVB    $NCODE(R0),R5   ;LOAD R5 FOR USE AS TABLE INDEX
 4909  017424  116560  001760  000002            MOVB    COMTBL(R5),$COMND(R0)  ;RP04 COMMAND CODE
 4910  017432  116060  000075  000030            MOVB    $NPATC(R0),$PATTC(R0)  ;PATTERN CODE
 4911  017440  016060  000076  000010            MOV     $NSEC(R0),$SEC(R0)  ;TRACK AND SECTOR ADDRESSES
 4912  017446  016060  000100  000012            MOV     $NCYL(R0),$CYL(R0)  ;CYLINDER ADDRESS
 4913  017454  016060  000102  000020            MOV     $NWRDL(R0),$WRDL(R0)  ;BUFFER SIZE
 4914  017462  016060  000102  000004            MOV     $NWRDL(R0),$WRDM(R0)  ;WORD COUNT FOR THE RH11
 4915  017470  005460  000004                    NEG     $WRDM(R0)       ;COMPLEMENT IT
 4916  017474  012760  000400  000022            MOV     #256.,$SSEC(R0) ;INITIAL VALUE OF SECTOR SIZE
 4917  017502  032760  000001  000024            BIT     #1,$CODE(R0)    ;HEADER OPERATION ?
 4918  017510  001403                            BEQ     3$              ;BR IF NOT
 4919  017512  062760  000004  000022            ADD     #4,$SSEC(R0)    ;ADD HEADER SIZE
 4920  017520  005060  000104              3$:   CLR     $NEXT(R0)       ;RESET 'PARAMETERS LOADED' INDICATOR
 4921  017524  012605                            MOV     (SP)+,R5        ;RESTORE R5
 4922  017526  000207                            RTS     PC              ;RETURN
 4923
 4924                                    ;ROUTINE TO COMPRESS A LIST
 4925                                    ;CALL:
 4926                                    ;       MOV     #ADDRS,R1       ;COMPRESS LIST STARTING AT THIS ADDRESS
 4927                                    ;       JSR     PC,CMPRES
 4928                                    ;       RETURN
 4929
 4930  017530  016111  000002            CMPRES: MOV     2(R1),(R1)      ;COMPRESS THE TABLE IN R1
 4931  017534  001403                            BEQ     1$              ;BR WHEN ZERO FOUND
 4932  017536  062701  000002                    ADD     #2,R1           ;INCREMENT R1
 4933  017542  000772                            BR      CMPRES          ;CONTINUE COMPRESSING TABLE
 4934  017544  000207                      1$:   RTS     PC              ;RETURN
 4935
 4936                                    ;ROUTINE TO SETUP PARAMETERS FOR A SEQUENTIAL READ OR WRITE OF THE DISK
 4937                                    ;CALL:
 4938                                    ;       MOV     #DPB,R0         ;DPB ADDRESS
 4939                                    ;       MOV     #-1,$PACK(R0)   ;'WRITE PACK' FLAG
 4940                                    ;  OR
 4941                                    ;       MOV     #1,$PACK(R0)    ;'READ PACK' FLAG
 4942                                    ;       JSR     PC,WRTPK
 4943                                    ;       RETURN
 4944
 4945  017546  004737  032364            WRTPK:  JSR     PC,$RAND        ;CYCLE THE RANDOM NUMBER GENERATOR
 4946  017552  005760  000040                    TST     $OPERC+2(R0)    ;SEE IF FIRST OPERATION
 4947  017556  001007                            BNE     WRTPK1          ;BR IF UPPER WORD OF COUNTER NOT ZERO
 4948  017560  005760  000036                    TST     $OPERC(R0)      ;LOWER WORD ZERO ?
 4949  017564  001004                            BNE     WRTPK1          ;BR IF NOT 1ST OPERATION
```

```
 4950  017566  105760  000026                 TSTB    $PACK(R0)              ;SEE WHICH - 'R' OR 'W'
 4951  017572  100503                         BMI     WRTPK3                 ;BR IF 'W'
 4952  017574  000470                         BR      WRTPK2                 ;'R' OPERATION
 4953  017576  116060  000234  000027  WRTPK1: MOVB    $RPCS1(R0),$PREV0(R0)  ;SAVE CURRENT PARAMETERS
 4954  017604  004737  022314                 JSR     PC,READDR              ;GET THE DECREMENTED SECTOR AND TRACK ADDRESSES
 4955  017610  112660  000033                 MOVB    (SP)+,$PREVA+1(R0)     ;TRACK ADDRESS
 4956  017614  112660  000032                 MOVB    (SP)+,$PREVA(R0)       ;SECTOR ADDRESS
 4957  017620  016060  000272  000034         MOV     $RPCC(R0),$PREVA+2(R0) ;CURRENT CYLINDER
 4958  017626  016060  000242  000010         MOV     $RPDA(R0),$SEC(R0)     ;NEW SECTOR & TRACK ADDRESS
 4959  017634  016060  000270  000012         MOV     $RPCA(R0),$CYL(R0)     ;NEW CYLINDER ADDRESS
 4960  017642  026060  000012  000106         CMP     $CYL(R0),MAXCYL(R0)    ;SEE IF AT END
 4961  017650  103427                         BLO     2$                     ;BR IF LESS THAN 'MAXCYL'
 4962  017652  101004                         BHI     1$                     ;BR IF GREATER THAN 'MAXCYL'
 4963  017654  126060  000011  000112         CMPB    $TRK(R0),MAXTRK(R0)    ;SEE IF AT MAX TRACK
 4964  017662  101422                         BLOS    2$                     ;BR IF NOT GREATER
 4965  017664  116060  000114  000011  1$:    MOVB    MINTRK(R0),$TRK(R0)    ;RESET TRACK ADDRESS
 4966  017672  116060  000120  000010         MOVB    MINSEC(R0),$SEC(R0)    ;RESET SECTOR ADDRESS
 4967  017700  016060  000110  000012         MOV     MINCYL(R0),$CYL(R0)    ;RESET CYLINDER ADDRESS
 4968  017706  004737  026626                 JSR     PC,EOP2                ;DROP THE DRIVE (NORMAL TERMINATION)
 4969  017712  032777  000020  161220         BIT     #SW04,@SWR             ;IS SWITCH 4 SET ?
 4970  017720  001003                         BNE     2$                     ;BR IF SET
 4971  017722  005726                         TST     (SP)+                  ;INCREMENT THE STACK POINTER
 4972  017724  000137  005374                 JMP     MAIN                   ;RETURN DIRECTLY TO 'MAIN'
 4973  017730  013760  001404  000020  2$:    MOV     MAXDL,$WRDL(R0)        ;BUFFER SIZE IS MAXIMUM
 4974  017736  013760  001404  000004         MOV     MAXDL,$WRDM(R0)        ;WORD COUNT
 4975  017744  005460  000004                 NEG     $WRDM(R0)              ;CHANGE WORD COUNT TO 2'S COMPLEMENT
 4976  017750  105760  000026                 TSTB    $PACK(R0)              ;READ OR WRITE ?
 4977  017754  100412                         BMI     WRTPK3                 ;BR IF WRITE
 4978  017756  012760  000404  000022  WRTPK2: MOV     #260.,$SSEC(R0)        ;SECTOR SIZE FOR READ
 4979  017764  112760  000005  000024         MOVB    #5,$CODE(R0)           ;CODE FOR READ HEADER & DATA
 4980  017772  112760  000173  000002         MOVB    #RDHD,$COMND(R0)       ;DRIVE CODE FOR OPERATION
 4981  020000  000415                         BR      WRTPK4                 ;SET UP FOR EXIT
 4982  020002  012760  000400  000022  WRTPK3: MOV     #256.,$SSEC(R0)        ;SECTOR SIZE
 4983  020010  112760  000002  000024         MOVB    #2,$CODE(R0)           ;CODE FOR WRTDAT
 4984  020016  112760  000161  000002         MOVB    #WRTDAT,$COMND(R0)     ;OP CODE
 4985  020024  004737  017274                 JSR     PC,GETPAT              ;GET PATTERN CODE
 4986  020030  110560  000030                 MOVB    R5,$PATTC(R0)          ;PATTERN CODE
 4987  020034  005060  000104         WRTPK4: CLR     $NEXT(R0)              ;CLEAR 'PARAMETER SELECTED' INDICATOR
 4988  020040  000207                         RTS     PC                     ;RETURN
 4989
 4990                                 ;ROUTINE TO DETERMINE OF ERROR IS AT A LOCATION ON THE PACK DEFINED
 4991                                 ;       IN THE BAD TRACK/SECTOR TABLE FOR THE DRIVE.
 4992                                 ;CALL:
 4993                                 ;       JSR     PC,SPOTCK
 4994                                 ;       RETURN1                        ;ERROR AT AN ADDRESS IN TABLE
 4995                                 ;       RETURN2                        ;NO TABLE ENTRY FOR ERROR ADDRESS OR
 4996                                 ;                                      ;PARAMETER 'NOTPRT' IS 0
 4997
 4998  020042                         SPOTCK:
 4999  020042  010146                         MOV     R1,-(SP)               ;;PUSH R1 ON STACK
 5000  020044  010246                         MOV     R2,-(SP)               ;;PUSH R2 ON STACK
 5001  020046  012701  000124                 MOV     #$BDSEC,R1             ;INCREMENT FOR BAD SECTOR TABLE
 5002  020052  060001                         ADD     R0,R1                  ;ADD THE BLOCK'S STARTING ADDRESS
 5003  020054  012702  000020                 MOV     #16.,R2                ;BAD SECTOR TABLE SIZE COUNT
 5004  020060  021160  000272         1$:     CMP     (R1),$RPCC(R0)         ;IS CYLINDER IN THE TABLE ?
 5005  020064  001022                         BNE     4$                     ;BR IF NOT
```

```
5006  020066  105761  000003                  TSTB    3(R1)           ;TRACK ENTRY ?
5007  020072  100426                          BMI     5$              ;BR IF NOT
5008  020074  004737  022314                  JSR     PC,READDR       ;DECREMENT THE SECTOR/TRACK ADDRESS
5009  020100  122661  000003                  CMPB    (SP)+,3(R1)     ;COMPARE THE TRACK ADDRESS
5010  020104  001011                          BNE     3$              ;BR IF IT IS NOT EQUAL
5011  020106  105761  000002                  TSTB    2(R1)           ;IS A SECTOR ADDRESS IN THE TABLE ?
5012  020112  100002                          BPL     2$              ;BR IF ONE IS
5013  020114  005726                          TST     (SP)+           ;INCREMENT THE STACK POINTER
5014  020116  000414                          BR      5$              ;DISPLAY THE MESSAGE
5015  020120  122661  000002          2$:     CMPB    (SP)+,2(R1)     ;COMPARE THE SECTOR ADDRESS
5016  020124  001002                          BNE     4$              ;BR IF NOT EQUAL
5017  020126  000410                          BR      5$              ;CHECK 'NOTPRT'
5018  020130  005726                  3$:     TST     (SP)+           ;INCREMENT THE STACK POINTER
5019  020132  062701  000004          4$:     ADD     #4,R1           ;GO TO THE NEXT LOCATION IN THE TABLE
5020  020136  005711                          TST     (R1)            ;PAST THE TABLE ENTRIES ?
5021  020140  100411                          BMI     6$              ;BR IF PAST
5022  020142  005302                          DEC     R2              ;DECREMENT THE MAXIMUM ENTRY COUNT
5023  020144  001345                          BNE     1$              ;BR IF MORE TO CHECK
5024  020146  000406                          BR      6$              ;END, EXIT
5025  020150  005737  001426          5$:     TST     NOTPRT          ;PRINT THE ERROR ANYWAY ?
5026  020154  001006                          BNE     7$              ;BR IF NOT
5027  020156  012737  177777  001264          MOV     #-1,BADSEC      ;SET THE INDICATOR FOR THE IDENTIFICATION LINE
5028  020164  062766  000002  000004  6$:     ADD     #2,4(SP)        ;INCREMENT THE RETURN
5029  020172                          7$:
5030  020172  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
5031  020174  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
5032  020176  000207                          RTS     PC              ;RETURN
5033
5034                          ;;*****************************************************************
5035
5036                          .SBTTL  ERROR MESSAGE GENERATION ROUTINES
5037
5038                          ;;*****************************************************************
5039
5040                          ;PRINT LINE 1 OF ERROR MESSAGE:
5041                          ;'HH:MM:SS'
5042
5043  020200  032777  002000  160732  LINE1:  BIT     #SW10,@SWR      ;SWITCH 10 SET ?
5044  020206  001402                          BEQ     1$              ;BR IF NOT
5045  020210  104401  001160                  TYPE    ,$BELL          ;RING THE BELL
5046  020214  032777  020000  160716  1$:     BIT     #SW13,@SWR      ;INHIBIT TYPEOUT ?
5047  020222  001403                          BEQ     2$              ;BR IF NOT
5048  020224  104414  001165                  DISPLY  ,$CRLF          ;CR-LF
5049  020230  000404                          BR      3$              ;EXIT
5050  020232  004737  023450          2$:     JSR     PC,$TIME        ;TYPE THE TIME
5051  020236  104414  052156                  DISPLY  ,LINSPO         ;SPACES
5052  020242  000207                  3$:     RTS     PC              ;RETURN & TYPE DESCRIPTION
5053
5054                          ;PRINT LINE 2 OF ERROR MESSAGE
5055                          ;'PRESENT ORDER = XXXX  PREVIOUS ORDER = XXXX'
5056                          ;'* ERROR AT BAD TRACK/SECTOR'
5057                          ;'DRV RPCS1   RPCS2   RPDS1   RPER1   RPER2   RPER3   RPEC1   RPEC2'
5058                          ;'RPWC    RPBA    RPDA    RPAS    RPLA    RPDB    RPMR    RPDT'
5059                          ;'RPSN    RPOF    RPCA    RPCC    STATUS'
5060                          ;'BUS ADDRESS OR WORD COUNT NOT CONSISTENT'
5061                          ;'RPBA = XXXXXX   RPWC = XXXXXX'
```

```
                                        ;'BUFFER ADR = XXXXXX   SIZE = XXXX   ACTUAL NMBR WRDS XFRD = XXX'

020244                                  LINE2:
020244  010346                                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
020246  010446                                  MOV     R4,-(SP)        ;;PUSH R4 ON STACK
020250  010546                                  MOV     R5,-(SP)        ;;PUSH R5 ON STACK
020252  104414  001165                          DISPLY  $CRLF           ;CR-LF
020256  005037  020404                          CLR     4$              ;CLEAR MESSAGE ADDRESS STORAGE
020262  005004                                  CLR     R4              ;WORKING REGISTER
020264  012737  050242  020404                  MOV     #LIN2C,4$       ;ADDRESS OF 'PRESENT ORDER = ' MSG
020272  116004  000234                          MOVB    $RPCS1(R0),R4   ;GET THE OPCODE
020276  042704  177701                          BIC     #^C76,R4        ;SAVE ONLY SIGNIFICANT BITS
020302  004737  020340                          JSR     PC,1$           ;TYPE THE FIRST MNEMONIC
020306  005737  020410                          TST     5$              ;SEE IF MNEMONIC ENTRY FOUND
020312  001440                                  BEQ     LINE2A          ;BR IF NOT
020314  012737  050263  020404                  MOV     #LIN2P,4$       ;ADDRESS OF 'PREVIOUS ORDER = ' MSG
020322  116004  000027                          MOVB    $PREVO(R0),R4   ;PREVIOUS OPERATION CODE
020326  042704  177701                          BIC     #^C76,R4        ;SAVE ONLY SIGNIFICANT BITS
020332  004737  020340                          JSR     PC,1$           ;TYPE THE PREVIOUS MNEMONIC
020336  000426                                  BR      LINE2A          ;CONTINUE
020340  005005                          1$:     CLR     R5              ;CLEAR THE TABLE INDEX
020342  126504  001766                  2$:     CMPB    OPTBL(R5),R4    ;LOOK FOR THE OPCODE
020346  001405                                  BEQ     3$              ;BR WHEN OPCODE COUNT EQUALS OPCODE
020350  105765  001766                          TSTB    OPTBL(R5)       ;LOOK FOR END OF TABLE
020354  100402                                  BMI     3$              ;BR IF END
020356  005205                                  INC     R5              ;INCREMENT THE POINTER
020360  000770                                  BR      2$              ;CONTINUE - NOT END OF TABLE
020362  006305                          3$:     ASL     R5              ;SHIFT INDEX
020364  006305                                  ASL     R5              ;SHIFT THE INDEX
020366  006305                                  ASL     R5              ;SHIFT THE INDEX
020370  012737  002010  020410                  MOV     #MNTBL,5$       ;ADDRESS OF ASCII TEXT TABLE
020376  060537  020410                          ADD     R5,5$           ;ADD THE INDEX
020402  104414                                  DISPLY                  ;TYPE IT
020404  000000                          4$:     .WORD   0               ;ADDRESS OF 'PRESENT' OR 'PREVIOUS' MESSAGE
020406  104414                                  DISPLY                  ;TYPE THE OPERATION MNEMONIC
020410  000000                          5$:     .WORD   0               ;ADDRESS OF MESSAGE
020412  000207                                  RTS     PC              ;RETURN TO MAIN ROUTINE
020414  005737  001264                  LINE2A: TST     BADSEC          ;PRINT THE BAD SECTOR LINE ?
020420  001404                                  BEQ     LINE2B          ;BR IF NOT
020422  104414  001165                          DISPLY  ,$CRLF          ;CR-LF
020426  104414  050307                          DISPLY  ,LIN2S          ;ERROR ADDRESS DEFINED AS BAD AREA
020432  104414  001165                  LINE2B: DISPLY  ,$CRLF          ;CR-LF
020436  104414  047636                          DISPLY  ,DH14           ;STANDARD RP04/5/6 REGISTER HEADER
020442  104414  052156                          DISPLY  ,LINSPO         ;TYPE A SPACE
020446  013746  001246                          MOV     UNIT,-(SP)      ;PUT THE DRIVE NUMBER ON THE STACK
020452  004737  022274                          JSR     PC,LINDEC       ;TYPE DRIVE NUMBER
020456  104414  052155                          DISPLY  ,LINSP          ;SPACES
020462  012705  050162                          MOV     #DT14,R5        ;REGISTER INDEXES
020466  004737  020616                          JSR     PC,3$           ;PRINT THE REGISTERS
020472  032777  000040  160440                  BIT     #SWO5,@SWR      ;PRINT THE OPTIONAL REGISTERS ?
020500  001014                                  BNE     1$              ;BR IF NOT
020502  104414  047742                          DISPLY  ,DH15
020506  012705  050204                          MOV     #DT15,R5        ;SECOND DATA LINE
020512  004737  020616                          JSR     PC,3$           ;PRINT THEM
020516  104414  050041                          DISPLY  ,DH16
020522  012705  050226                          MOV     #DT16,R5        ;THIRD DATA LINE
```

```
020526  004737  020616                  JSR     PC,3$                  ;PRINT THE REGISTERS
020532  032760  000100  000016  1$:     BIT     #BIT6,$STATUS(R0)      ;DATA ERROR ?
020540  001422                          BEQ     2$                     ;BR IF NOT
020542  016046  000020                  MOV     $WRDL(R0),-(SP)        ;TRANSFER SIZE
020546  066016  000236                  ADD     $RPWC(R0),(SP)         ;ADD REMAINING WORD COUNT
020552  006316                          ASL     (SP)                   ;CONVERT TO AN BYTE INCREMENT
020554  066016  000006                  ADD     $BUF(R0),(SP)          ;BUFFER STARTING ADDRESS
020560  022660  000240                  CMP     (SP)+,$RPBA(R0)        ;CORRECT BUFFER ADDRESS ?
020564  001410                          BEQ     2$                     ;BR IF YES
020566  104414  047231                  DISPLY  ,EM46                  ;'BUS ADDRESS AND WORD COUNT ARE NOT CONSISTENT'
020572  104414  001165                  DISPLY  ,$CRLF                 ;CR-LF
020576  004737  020710                  JSR     PC,LINE3D              ;PRINT LINE 3D OF ERROR MESSAGE
020602  004737  021326                  JSR     PC,LINE4               ;PRINT LINE 4 OF ERROR MESSAGE
020606                          2$:
020606  012605                          MOV     (SP)+,R5               ;;POP STACK INTO R5
020610  012604                          MOV     (SP)+,R4               ;;POP STACK INTO R4
020612  012603                          MOV     (SP)+,R3               ;;POP STACK INTO R3
020614  000207                          RTS     PC                     ;RETURN TO ERROR PROCESSING ROUTINE
020616  012546                  3$:     MOV     (R5)+,-(SP)            ;PUT THE REGISTER INDEX ON THE STACK
020620  060016                          ADD     R0,(SP)                ;ADD DRIVE'S TABLE ADDRESS
020622  017646  000000                  MOV     @(SP),-(SP)            ;VALUE
020626  004737  022242                  JSR     PC,LINOCT              ;TYPE IT
020632  005726                          TST     (SP)+                  ;CORRECT THE STACK POINTER
020634  104414  052155                  DISPLY  ,LINSP                 ;PRINT 2 SPACES
020640  005715                          TST     (R5)                   ;AT END OF LINE ?
020642  001365                          BNE     3$                     ;BR IF NOT
020644  104414  001165          4$:     DISPLY  ,$CRLF                 ;CR-LF
020650  000207                          RTS     PC                     ;RETURN

                                ;PRINT LINE 3 OF ERROR MESSAGE
                                ;'ERROR AT CCC TT SS   PREVIOUS ADR = CCC TT SS'

020652  104414  050343          LINE3:  DISPLY  ,LINM3                 ;LINE 3 ENTRANCE
020656  000517                          BR      LIN3.1                 ;FINISH PRINTOUT

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

020660  104414  050361          LINE3A: DISPLY  ,LINN3                 ;LINE 3A ENTRANCE
020664  000514                          BR      LIN3.1                 ;FINISH ERROR LINE

                                ;PRINT LINE 3B OF ERROR MESSAGE
                                ;'START CYL = CCC   END CYL = CCC   ACTUAL CYL = CCC'

020666  004737  021230          LINE3B: JSR     PC,LIN3.3              ;LINE 3B ENTRANCE
020672  104414  001165                  DISPLY  ,$CRLF
020676  000207                          RTS     PC

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

020700  004737  021230          LINE3C: JSR     PC,LIN3.3              ;LINE 3C ENTRANCE
020704  000137  021262                  JMP     LIN3.4                 ;FINISH MESSAGE

                                ;PRINT LINE 3D OF ERROR MESSAGE
                                ;'RPBA = XXXXXX   RPWC = XXXXXX'
```

```
020710  032777  000040  160222  LINE3D: BIT     #SW05,@SWR          ;SWITCH 5 SET ?
020716  001416                          BEQ     1$                  ;BR IF IT IS
020720  104414  050532                  DISPLY  ,LINB3              ;'RPBA = '
020724  016046  000240                  MOV     $RPBA(R0),-(SP)     ;BUFFER ADDR REG CONTENTS
020730  004737  022242                  JSR     PC,LINOCT           ;CONVERT TO OCTAL AND TYPE IT
020734  104414  050542                  DISPLY  ,LINW3              ;'   RPWC = '
020740  016046  000236                  MOV     $RPWC(R0),-(SP)     ;WORD COUNT REGISTER CONTENTS
020744  004737  022242                  JSR     PC,LINOCT           ;CONVERT TO OCTAL AND TYPE IT
020750  104414  001165                  DISPLY  ,$CRLF
020754  000207                  1$:     RTS     PC

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

020756  104414  050426          LINE3E: DISPLY  ,LINS3              ;'START CYL = '
020762  016046  000012                  MOV     $CYL(R0),-(SP)      ;MOVE CYL TO STACK
020766  004737  022274                  JSR     PC,LINDEC           ;TYPE IT IN DECIMAL
020772  104414  052155                  DISPLY  ,LINSP              ;SPACES
020776  104414  050554                  DISPLY  ,LINST3             ;'START TRK = '
021002  005046                          CLR     -(SP)               ;CLEAR STACK
021004  116016  000011                  MOVB    $TRK(R0),(SP)       ;TRACK TO STACK
021010  004737  022274                  JSR     PC,LINDEC           ;TYPE IT IN DECIMAL
021014  104414  052155                  DISPLY  ,LINSP              ;SPACES
021020  104414  050571                  DISPLY  ,LINSS3             ;'START SEC = '
021024  005046                          CLR     -(SP)               ;CLEAR STACK
021026  116016  000010                  MOVB    $SEC(R0),(SP)       ;SECTOR ADDR TO STACK
021032  004737  022274                  JSR     PC,LINDEC           ;TYPE IT IN DECIMAL
021036  104414  001165                  DISPLY  ,$CRLF
021042  000207                          RTS     PC

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

021044  032777  000040  160066  LINE3F: BIT     #SW5,@SWR           ;SWITCH 5 SET ?
021052  001420                          BEQ     1$                  ;BR IF NOT
021054  104414  050522                  DISPLY  ,LINDA3             ;'RPDA = '
021060  016046  000242                  MOV     $RPDA(R0),-(SP)     ;PUT SECTOR/TRACK ADDRESS ON THE STACK
021064  004737  022242                  JSR     PC,LINOCT           ;TYPE IT
021070  104414  052155                  DISPLY  ,LINSP              ;SPACES
021074  104414  050511                  DISPLY  ,LINCA3             ;' RPCA = '
021100  016046  000270                  MOV     $RPCA(R0),-(SP)     ;PUT DESIRED CYLINDER ADDRESS ON THE STACK
021104  004737  022242                  JSR     PC,LINOCT           ;TYPE IT
021110  104414  001165                  DISPLY  ,$CRLF
021114  000207                  1$:     RTS     PC

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

021116  016046  000272          LIN3.1: MOV     $RPCC(R0),-(SP)     ;PUT CYLINDER ADDR ON STACK
021122  004737  022274                  JSR     PC,LINDEC           ;TYPE IT IN DECIMAL
021126  104414  050356                  DISPLY  ,T                  ;PRINT ' T'
021132  004737  022314                  JSR     PC,READDR           ;DECREMENT TRACK AND SECTOR ADDRESSES
021136  004737  022274                  JSR     PC,LINDEC           ;TYPE TRACK IN DECIMAL
021142  104414  050402                  DISPLY  ,S                  ;PRINT ' S'
021146  004737  022274                  JSR     PC,LINDEC           ;TYPE SECTOR ADDRESS
021152  104414  050405                  DISPLY  ,LINP3              ;PRINT 'PREV ADDR'
```

```
5230  021156  016046  000034          MOV     $PREVA+2(R0),-(SP) ;PREVIOUS CYLINDER
5231  021162  004737  022274          JSR     PC,LINDEC          ;TYPE IT IN DECIMAL
5232  021166  104414  050356          DISPLY  ,T                 ;PRINT ' T'
5233  021172  005046                  CLR     -(SP)              ;MAKE ROOM ON THE STACK
5234  021174  116016  000033          MOVB    $PREVA+1(R0),(SP)  ;PREVIOUS TRACK ADDRESS
5235  021200  004737  022274          JSR     PC,LINDEC          ;TYPE IT IN DECIMAL
5236  021204  104414  050402          DISPLY  ,S                 ;PRINT ' S'
5237  021210  005046                  CLR     -(SP)              ;MAKE ROOM ON THE STACK
5238  021212  116016  000032          MOVB    $PREVA(R0),(SP)    ;PREVIOUS SECTOR DDRESS
5239  021216  004737  022274          JSR     PC,LINDEC          ;TYPE IT IN DECIMAL
5240  021222  104414  001165          DISPLY  ,$CRLF
5241  021226  000207                  RTS     PC
5242
5243                                  ;'START CYL = CCC   END CYL = CCC'
5244
5245  021230  104414  050426  LIN3.3: DISPLY  ,LINS3             ;LINE '3B & 3C' ENTRANCE
5246  021234  016046  000034          MOV     $PREVA+2(R0),-(SP) ;PREVIOUS CYLINDER
5247  021240  004737  022274          JSR     PC,LINDEC          ;TYPE IT IN DECIMAL
5248  021244  104414  050443          DISPLY  ,LINEN3            ;PRINT 'END CYL'
5249  021250  016046  000272          MOV     $RPCC(R0),-(SP)    ;PRESENT CYLINDER
5250  021254  004737  022274          JSR     PC,LINDEC          ;TYPE IT IN DECIMAL
5251  021260  000207                  RTS     PC
5252
5253                                  ;'ACTUAL CYL = CCC   TRK = TT'
5254
5255  021262  104414  050460  LIN3.4: DISPLY  ,LINA3             ;PRINT 'ACTUAL'
5256  021266  013746  054456          MOV     CYLDER,-(SP)       ;ACTUAL CYLINDER
5257  021272  042716  010000          BIC     #BIT12,(SP)        ;CLEAR THE FORMAT BIT
5258  021276  004737  022274          JSR     PC,LINDEC          ;TYPE IT IN DECIMAL
5259  021302  104414  050500          DISPLY  ,LINT3             ;PRINT TRACK
5260  021306  005046                  CLR     -(SP)              ;CLEAR STACK WORD
5261  021310  116016  000243          MOVB    $RPDA+1(R0),(SP)   ;PUT TRACK ON STACK
5262  021314  004737  022274          JSR     PC,LINDEC          ;TYPE IT IN DECIMAL
5263  021320  104414  001165          DISPLY  ,$CRLF
5264  021324  000207                  RTS     PC
5265
5266                                  ;PRINT LINE 4 OF ERROR MESSAGE
5267                                  ;'BUFFER ADR = XXXXXX   SIZE = XXXX   ACTUAL NMBR WRDS XFRD = XXX'
5268
5269  021326  032760  000100  000016 LINE4:  BIT     #BIT06,$TATUS(R0)  ;DATA ERROR ?
5270  021334  001427                  BEQ     1$                 ;BR IF NOT
5271  021336  104414  050606          DISPLY  ,LINM4             ;'PRINT BUFFER'
5272  021342  016046  000006          MOV     $BUF(R0),-(SP)     ;BUFFER ADDR ON STACK
5273  021346  004737  022242          JSR     PC,LINOCT          ;CONVERT TO OCTAL & PRINT
5274  021352  104414  050625          DISPLY  ,LINS4             ;PRINT 'SIZE'
5275  021356  016046  000020          MOV     $WRDL(R0),-(SP)    ;BUFFER SIZE
5276  021362  004737  022274          JSR     PC,LINDEC          ;TYPE IT IN DECIMAL
5277  021366  104414  050637          DISPLY  ,LINX4             ;'ACTUAL NMBR WRDS XFRD = '
5278  021372  016046  000240          MOV     $RPBA(R0),-(SP)    ;VALUE IN BUFFER ADDR REGISTER
5279  021376  166016  000006          SUB     $BUF(R0),(SP)      ;SUBTRACT STARTING ADDRESS
5280  021402  006216                  ASR     (SP)               ;CONVERT INTO A WORD COUNT
5281  021404  004737  022274          JSR     PC,LINDEC          ;TYPE IT IN DECIMAL
5282  021410  104414  001165          DISPLY  ,$CRLF             ;CR-LF
5283  021414  000207          1$:     RTS     PC                 ;RETURN
5284
5285                                  ;PRINT LINE 5 OF ERROR MESSAGE
```

```
                                ;'GOOD DATA = XXXXXX   BAD DATA = XXXXXX   SECT POS = XXX'
021416  104414  050672        LINE5:  DISPLY  ,LIND5          ;PRINT 'GOOD DATA'
021422  162760  000002 000240         SUB     #2,$RPBA(R0)    ;BACK THE ADDRESS UP
021430  017046  000240               MOV     @$RPBA(R0),-(SP)  ;'GOOD' DATA - AT THE BUFFER LOCATION
021434  004737  022242               JSR     PC,LINOCT       ;TYPE IT
021440  104414  050707               DISPLY  ,LINB5          ;PRINT 'BAD DATA'
021444  016046  000256               MOV     $RPDB(R0),-(SP) ;BAD DATA FROM BUFFER
021450  004737  022242               JSR     PC,LINOCT       ;TYPE IT
021454  016046  000236               MOV     $RPWC(R0),-(SP) ;WORD LENGTH ON STACK
021460  066016  000020               ADD     $WRDL(R0),(SP)  ;MAKE INTO A POSITIVE NUMBER
021464  005046                       CLR     -(SP)           ;UPPER DIVIDEND TO ZERO
021466  016046  000022               MOV     $SSEC(R0),-(SP)  ;SECTOR SIZE ON THE STACK
021472  004737  027054               JSR     PC,LINKDV       ;DIVIDE WORDS XFERED BY SECTOR SIZE
021476  012616                       MOV     (SP)+,(SP)      ;MOVE REMAINDER UP THE STACK
021500  104414  050725               DISPLY  ,LINP5          ;PRINT 'SECT POS'
021504  004737  022274               JSR     PC,LINDEC       ;TYPE THE POSITION
021510  104414  001165               DISPLY  ,$CRLF
021514  000207                       RTS     PC

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

021516  104414  050743        LINE5A: DISPLY  ,LIN5S          ;'HEADER CONTENTS OF ERROR SECTOR'
021522  013746  054456               MOV     CYLDER,-(SP)    ;HEADER POSITION
021526  004737  022242               JSR     PC,LINOCT       ;TYPE IT
021532  104414  052155               DISPLY  ,LINSP          ;SPACES
021536  013746  054460               MOV     CYLDER+2,-(SP)  ;HEADER POSITION +2
021542  004737  022242               JSR     PC,LINOCT       ;TYPE IT
021546  104414  052155               DISPLY  ,LINSP          ;SPACES
021552  013746  054462               MOV     CYLDER+4,-(SP)  ;HEADER POSITION +4
021556  004737  022242               JSR     PC,LINOCT       ;TYPE IT
021562  104414  052155               DISPLY  ,LINSP          ;SPACES
021566  013746  054464               MOV     CYLDER+6,-(SP)  ;HEADER POSITION +6
021572  004737  022242               JSR     PC,LINOCT       ;TYPE IT
021576  104414  052155               DISPLY  ,LINSP          ;SPACES
021602  104414  001165               DISPLY  ,$CRLF
021606  000207                       RTS     PC

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

021610  104414  050777        LINE5B: DISPLY  ,LINEP5         ;'RPEC1 = '
021614  016046  000300               MOV     $RPEC1(R0),-(SP)  ;PUT REGISTER CONTENTS ON THE STACK
021620  004737  022242               JSR     PC,LINOCT       ;TYPE IT
021624  104414  052155               DISPLY  ,LINSP          ;SPACES
021630  104414  051010               DISPLY  ,LINE05         ;' RPEC2 = '
021634  016046  000302               MOV     $RPEC2(R0),-(SP)  ;PUT REGISTER CONTENTS ON THE STACK
021640  004737  022242               JSR     PC,LINOCT       ;TYPE IT
021644  104414  001165               DISPLY  ,$CRLF
021650  000207                       RTS     PC              ;RETURN

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

021652  104414  051022        LINE6:  DISPLY  ,LIN86          ;ECC CORRECTABLE
```

```
5342  021656  104414  001165            DISPLY  ,$CRLF
5343  021662  000207                    RTS     PC
5344
5345                                    ;PRINT LINE 6A OF THE ERROR MESSAGE
5346                                    ;'SECTOR READ CORRECTLY AT OFFSET N'
5347
5348  021664  104414  051055    LINE6A: DISPLY  ,LINC6          ;PRINT 'READ CORRECTLY AT OFFSET N'
5349  021670  000411                    BR      LIN6.1          ;TYPE THE REST OF THE LINE
5350
5351                                    ;PRINT LINE 6B OF THE ERROR MESSAGE
5352                                    ;'SECTOR IS ECC CORRECTABLE AT OFFSET N'
5353
5354  021672  104414  051022    LINE6B: DISPLY  ,LINB6          ;PRINT 'SECTOR IS ECC CORRECTABLE '
5355  021676  000406                    BR      LIN6.1
5356
5357                                    ;PRINT LINE 6C OF THE ERROR MESSAGE
5358                                    ;'CORRECTED ON NTH RETRY'
5359
5360  021700  104414  051104    LINE6C: DISPLY  ,LING6          ;'CORRECTED ON NTH RETRY'
5361  021704  000414                    BR      LIN6.2          ;TYPE THE REST OF THE LINE
5362
5363                                    ;PRINT LINE 6D OF THE ERROR MESSAGE
5364                                    ;'UNCORRECTABLE AFTER N RETRIES'
5365
5366  021706  104414  051133    LINE6D: DISPLY  ,LINU06         ;'UNCORRECTABLE AFTER N RETRIES'
5367  021712  000411                    BR      LIN6.2          ;FINISH
5368
5369                                    ;TYPE THE OFFSET VALUE IN MICRO-INCHES
5370
5371  021714  006301            LIN6.1: ASL     R1              ;DOUBLE THE OFFSET TABLE INDEX
5372  021716  016137  002240  021726    MOV     OFMTBL(R1),1$   ;ADDRESS OF OFFSET POSITION MESSAGE
5373  021724  104414                    DISPLY
5374  021726  000000            1$:     .WORD   0               ;OFFSET VALUE
5375  021730  104414  001165            DISPLY  ,$CRLF
5376  021734  000207                    RTS     PC
5377
5378                                    ;RETRY COUNT TYPEOUT
5379
5380  021736  005046    LIN6.2: CLR     -(SP)           ;CLEAR STACK
5381  021740  113716  001253            MOVB    RETRY+1,(SP)    ;RETRY COUNT
5382  021744  004737  022274            JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
5383  021750  104414  051122            DISPLY  ,LINR6          ;'RETRY'
5384  021754  104414  001165            DISPLY  ,$CRLF
5385  021760  000207                    RTS     PC
5386
5387                                    ;PRINT LINE 7 OF THE ERROR MESSAGE
5388                                    ;'ORDERS:XXXXX   TOTAL ERRORS:XXX   WRDS XFRD:XXXXXXX   WRDS READ:XXXXXXX'
5389
5390  021762  104414  051206    LINE7:  DISPLY  ,LIN7O          ;PRINT ORDER COUNT
5391  021766  012746  000036            MOV     #$OPERC,-(SP)   ;TO STACK
5392  021772  060016                    ADD     R0,(SP)         ;ADD THE BASE ADDRESS
5393  021774  004737  032562            JSR     PC,$DB2D        ;CONVERT IT
5394  022000  004737  027370            JSR     PC,$SUPRS       ;PRINT IT
5395  022004  104414  051265            DISPLY  ,LIN7T          ;TOTAL ERRORS
5396  022010  016046  000056            MOV     $TOTAL(R0),-(SP)        ;TO STACK
5397  022014  004737  022274            JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
```

```
5398  022020  104414  051277                DISPLY  ,LIN7X          ;PRINT 'WRDS XFR'
5399  022024  012746  000046                MOV     #$TRANS,-(SP)   ;ADDRESS OF LOW WORD ON STACK
5400  022030  060016                        ADD     R0,(SP)
5401  022032  004737  032562                JSR     PC,$DB2D        ;CONVERT
5402  022036  004737  027370                JSR     PC,$SUPRS       ;PRINT
5403  022042  104414  051313                DISPLY  ,LIN7R          ;'BITS READ'
5404  022046  012746  000052                MOV     #$READ,-(SP)    ;LOW WORD ADDRESS
5405  022052  060016                        ADD     R0,(SP)
5406  022054  004737  032562                JSR     PC,$DB2D        ;CONVERT
5407  022060  004737  027370                JSR     PC,$SUPRS       ;PRINT
5408  022064  104414  001165                DISPLY  ,$CRLF
5409  022070  104414  001165                DISPLY  ,$CRLF
5410  022074  032777  100000  157036        BIT     #SW15,@SWR      ;SEE IF 'HALT ON ERROR' - SWITCH 15
5411  022102  001401                        BEQ     1$              ;BR IF NOT
5412  022104  000000                        HALT                    ;SWITCH 15 HALT
5413  022106  000207                1$:     RTS     PC
5414
5415                                  ;PRINT LINE 7A OF ERROR MESSAGE
5416                                  ;'ORDERS:XXXXX   TOTAL SEEKS=XXXXX   TOTAL MISPOS ERR = XXX   TOTAL SKI,OCYL ERR = XXX'
5417
5418  022110  104414  051206        LINE7A: DISPLY  ,LIN7O          ;'ORDERS = '
5419  022114  012746  000036                MOV     #$OPERC,-(SP)   ;ORDER COUNT INCREMENT
5420  022120  060016                        ADD     R0,(SP)         ;ADD BASE ADDRESS
5421  022122  004737  032562                JSR     PC,$DB2D        ;CONVERT THE COUNT
5422  022126  004737  027370                JSR     PC,$SUPRS       ;PRINT IT
5423  022132  104414  051216                DISPLY  ,LIN7P          ;'TOTAL SEEKS = '
5424  022136  012746  000042                MOV     #$POSIT,-(SP)   ;TOTAL SEEKS
5425  022142  060016                        ADD     R0,(SP)         ;DEVICE TABLE ADDRESS
5426  022144  004737  032562                JSR     PC,$DB2D        ;CONVERT THE SEEK COUNT
5427  022150  004737  027370                JSR     PC,$SUPRS       ;PRINT IT
5428  022154  104414  051160                DISPLY  ,LIN7M          ;'  TOTAL MISPOS ERR = '
5429  022160  016046  000066                MOV     $MISPO(R0),-(SP) ;TOTAL ERRORS
5430  022164  004737  022274                JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
5431  022170  104414  051236                DISPLY  ,LIN7S          ;'  TOTAL SKI,OCYL ERR = '
5432  022174  016046  000064                MOV     $SKI(R0),-(SP)  ;CONVERT & PRINT IT
5433  022200  004737  022274                JSR     PC,LINDEC       ;TYPE IT IN DECIMAL
5434  022204  104414  001165                DISPLY  ,$CRLF
5435  022210  104414  001165                DISPLY  ,$CRLF
5436  022214  032777  100000  156716        BIT     #SW15,@SWR      ;SEE IF HALT ON ERROR - SWITCH 15 SET
5437  022222  001401                        BEQ     1$              ;BR IF NOT
5438  022224  000000                        HALT                    ;SWITCH 15 HALT
5439  022226  000207                1$:     RTS     PC
5440
5441                                  ;PRINT LINE 8 OF THE ERROR MESSAGE
5442                                  ;'DIFFERENT ERROR DURING RETRY'
5443
5444  022230  104414  051330        LINE8:  DISPLY  ,LIN8M
5445  022234  004737  020244                JSR     PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
5446  022240  000207                        RTS     PC
5447
5448                                  ;OCTAL TYPEOUT ROUTINE
5449                                  ;CALL:
5450                                  ;       MOV     NUM,-(SP)       ;PUT THE NUMBER ON THE STACK
5451                                  ;       JSR     PC,LINOCT
5452                                  ;       RETURN
5453
```

```
 5454  022242  016646  000002          LINOCT: MOV     2(SP),-(SP)     ;PUT NUMBER IN PROPER LOCATION ON STACK
 5455  022246  004737  030020                  JSR     PC,$SB20        ;CONVERT THE NUMBER TO OCTAL
 5456  022252  012637  022266                  MOV     (SP)+,1$        ;GET THE ADDRESS OF THE ASCII STRING
 5457  022256  062737  000005  022266          ADD     #5.,1$          ;ADDRESS THE LAST 6 ASCII DIGITS
 5458  022264  104414                          DISPLY                  ;TYPE IT
 5459  022266  000000                  1$:     .WORD   0               ;ADDRESS
 5460  022270  012616                          MOV     (SP)+,(SP)      ;CORRECT THE STACK
 5461  022272  000207                          RTS     PC              ;RETURN
```

```
                                        ;ROUTINE TO CONVERT THE INPUT NUMBER TO DECIMAL AND TYPE IT WITH
                                        ;LEADING ZERO SUPRESSION
                                        ;CALL:
                                        ;       MOV     NUM,-(SP)       ;PUT THE NUMBER ON THE STACK
                                        ;       JSR     PC,LINDEC
                                        ;       RETURN
                                        ;
022274  016646  000002          LINDEC: MOV     2(SP),-(SP)     ;SET UP STACK FOR CONVERT
022300  004737  027770                  JSR     PC,$$B2D        ;CONVERT IT TO DECIMAL
022304  004737  027370                  JSR     PC,$SUPRS       ;TYPE IT (WITH LEADING ZEROS SUPRESSED)
022310  012616                          MOV     (SP)+,(SP)      ;RESTORE STACK POINTER
022312  000207                          RTS     PC

                                        ;;******************************************************************

                                        .SBTTL  GENERAL SUPPORT SUBROUTINES

                                        ;;******************************************************************

                                        ;DECREMENT THE SECTOR-TRACK ADDRESS
                                        ;CALL:
                                        ;       MOV     #DPB,R0         ;DPB ADDRESS
                                        ;       JSR     PC,READDR
                                        ;       RETURN
                                        ;       (SP) CONTAINS THE TRACK ADDRESS
                                        ;       2(SP) CONTAINS THE SECTOR ADDRESS

022314  162706  000004          READDR: SUB     #4,SP           ;DECREMENT THE STACK POINTER
022320  016616  000004                  MOV     4(SP),(SP)      ;MOVE THE RETURN ADDR DOWN THE STACK
022324  005066  000004                  CLR     4(SP)           ;CLEAR STACK FOR SECTOR
022330  005066  000002                  CLR     2(SP)           ;CLEAR STACK FOR TRACK
022334  116066  000242  000004          MOVB    $RPDA(R0),4(SP)  ;INCREMENTED SECTOR ON STACK
022342  005366  000004                  DEC     4(SP)           ;DECREMENT THE SECTOR ADDRESS
022346  100015                          BPL     1$              ;BR IF SECTOR GREATER THAN 0
022350  012766  000025  000004          MOV     #21.,4(SP)      ;JAM SECTOR ADDRESS TO 21(10)
022356  116066  000243  000002          MOVB    $RPDA+1(R0),2(SP)  ;TRACK ADDRESS
022364  005366  000002                  DEC     2(SP)           ;DECREMENT TRACK ADDRESS
022370  100007                          BPL     2$              ;BR IF IT DIDN'T GO NEG
022372  012766  000022  000002          MOV     #18.,2(SP)      ;RESET TRACK TO 18(10)
022400  000403                          BR      2$
022402  116066  000243  000002  1$:     MOVB    $RPDA+1(R0),2(SP)  ;TRACK ADDRESS
022410  000207                  2$:     RTS     PC              ;RETURN

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

022412  012737  177777  001210  CKCLK:  MOV     #-1,CLKFLG      ;CLEAR CLOCK AVAILABILITY FLAG
022420  012737  177777  001206          MOV     #-1,PCLOCK      ;CLEAR KW11-P CLOCK AVAILABILITY FLAG
022426  012737  022506  000004          MOV     #CKCLK1,ERRVEC  ;SET UP VECTOR FOR CLOCK CHECK
022434  005037  000006                  CLR     @#ERRVEC+2      ;NEW PSW
022440  005777  156530                  TST     @$LKCSR         ;CHECK FOR KW11-P
022444  005037  001210                  CLR     CLKFLG          ;SET CLOCK AVAILABILITY FLAG
022450  005037  001206                  CLR     PCLOCK          ;SET KW11-P CLOCK FLAG
022454  013701  001200                  MOV     $LPVEC,R1       ;KW11-P VECTOR ADDRESS
022460  012721  023546                  MOV     #CLOCK,(R1)+    ;SET UP KW11-P VECTOR
022464  012711  000300                  MOV     #300,(R1)       ;PSW - PRI 6
```

```
5518  022470  012777  174575  156500          MOV     #-1667.,@$LKCSB ;LOAD COUNTER BUFFER WITH 16.67
5519  022476  012777  000131  156470          MOV     #131,@$LKCSR    ;SET CLOCK - CNT UP, 10US, CONT INT
5520  022504  000437                          BR      CKCLK3
5521  022506  062706  000004          CKCLK1: ADD     #4,SP           ;RESTORE THE STACK POINTER
5522  022512  012737  022554  000004          MOV     #CKCLK2,@#ERRVEC  ;CHANGE ERROR VECTOR TO CHECK FOR KW11-L
5523  022520  005777  156456                  TST     @$LKS           ;LOOK FOR KW11-L
5524  022524  005037  001210                  CLR     CLKFLG          ;SET CLOCK FLAG
5525  022530  013701  001204                  MOV     $LLVEC,R1       ;KW11-L VECTOR ADDRESS
5526  022534  012721  023546                  MOV     #CLOCK,(R1)+    ;SET UP KW11-L VECTOR
5527  022540  012711  000300                  MOV     #300,(R1)       ;PSW - PRI 6
5528  022544  012777  000100  156430          MOV     #100,@$LKS      ;SET KW11-L INTERRUPT
5529  022552  000414                          BR      CKCLK3
5530  022554  062706  000004          CKCLK2: ADD     #4,SP           ;RESTORE THE STACK POINTER
5531  022560  104401  052733                  TYPE    ,NEDCLK         ;'P OR L CLOCK MUST BE ON SYSTEM'
5532  022564  005737  000042                  TST     42              ;UNDER MONITOR CONTROL ?
5533  022570  001402                          BEQ     1$              ;BR IF NOT
5534  022572  000137  005344                  JMP     $GET42          ;ABORT PROGRAM
5535  022576  000000                  1$:     HALT                    ;HALT
5536  022600  000137  004106                  JMP     START           ;TRY AGAIN
5537  022604  012737  000006  000004  CKCLK3: MOV     #6,@#ERRVEC     ;RESTORE THE ERROR VECTOR
5538  022612  000207                          RTS     PC
5539
5540                                  ;ROUTINE TO DISPLAY STATISTICS FOR ALL DRIVES ASSIGNED
5541                                  ;CALL:
5542                                  ;       JSR     PC,STATPR
5543                                  ;       RETURN
5544
5545  022614  010046          STATPR: MOV     R0,-(SP)        ;SAVE R0
5546  022616  010446                  MOV     R4,-(SP)        ;SAVE R4
5547  022620  005737  001462          TST     ASNLST          ;ANY DRIVES ASSIGNED ?
5548  022624  001421                  BEQ     3$              ;BR IF NOT
5549  022626  004737  022724          JSR     PC,SHDTYP       ;TYPE THE HEADING
5550  022632  005004                  CLR     R4              ;CLEAR THE DRIVE INDEX
5551  022634  006304          1$:     ASL     R4              ;CHANGE TO INDEX WORDS
5552  022636  016400  001740          MOV     BLKADR(R4),R0   ;GET THE DRIVE'S BLOCK ADDRESS
5553  022642  006204                  ASR     R4              ;RESTORE R4
5554  022644  136437  033320  001462  BITB    ATABIT(R4),ASNLST  ;IS THIS DRIVE ASSIGNED ?
5555  022652  001402                  BEQ     2$              ;BR IF NOT
5556  022654  004737  022746          JSR     PC,SDETAL       ;TYPE THE PERFORMANCE SUMMARY
5557  022660  005204          2$:     INC     R4              ;INCREMENT THE INDEX
5558  022662  020427  000010          CMP     R4,#8.          ;FINISHED ?
5559  022666  001362                  BNE     1$              ;BR IF NOT
5560  022670  012604          3$:     MOV     (SP)+,R4        ;RESTORE R4
5561  022672  012600                  MOV     (SP)+,R0        ;RESTORE R0
5562  022674  000207                  RTS     PC              ;RETURN
5563
5564                                  ;ROUTINE TO TYPE STATISTICS FOR AN INDIVIDUAL DRIVE
5565                                  ;CALL:
5566                                  ;       MOV     #DPB,R0         ;DPB ADDRESS
5567                                  ;       JSR     PC,TYPEST
5568                                  ;       RETURN
5569
5570  022676  010046          TYPEST: MOV     R0,-(SP)        ;SAVE R0
5571  022700  010446                  MOV     R4,-(SP)        ;SAVE R4
5572  022702  004737  022724          JSR     PC,SHDTYP       ;TYPE THE HEADING
5573  022706  005004                  CLR     R4              ;CLEAR R4 FOR DRIVE NUMBER
```

```
                                        MOVB    (R0),R4         ;DRIVE NUMBER
                                        JSR     PC,$DETAL       ;TYPE THE STATISTICS
                                        MOV     (SP)+,R4        ;RESTORE R4
                                        MOV     (SP)+,R0        ;RESTORE R0
                                        RTS     PC              ;RETURN

                                ;TYPE THE HEADER FOR THE DRIVE PERFORMANCE SUMMARY TYPEOUT
                                ;CALL:
                                ;       JSR     PC,SHDTYP
                                ;       RETURN

                                SHDTYP: JSR     PC,$TIME        ;TYPE THE TIME OF DAY
                                        JSR     R5,TYPRI4       ;TYPE AT PRIORITY 4
                                        $CRLF                   ;CR-LF
                                        JSR     R5,TYPRI4       ;TYPE THE HEADER
                                        STATHD                  ;HEADER
                                        RTS     PC              ;RETURN

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

                                SDETAL: MOV     R2,-(SP)        ;SAVE R2
                                        MOV     R0,R2           ;DPB ADDRESS
                                        ADD     #$OPERC,R2      ;FIRST STATISTICAL FIELD
                                        MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
                                                                ;;TYPE DRIVE NUMBER
                                        TYPOS                   ;;GO TYPE--OCTAL ASCII
                                        .BYTE   2               ;;TYPE 2 DIGIT(S)
                                        .BYTE   0               ;;SUPPRESS LEADING ZEROS
                                        TYPE    ,LINSP          ;SPACES
                                        MOV     $PASSC(R0),-(SP)  ;PUT THE PASS COUNT ON THE STACK
                                        JSR     PC,$SB2D        ;CONVERT IT
                                        JSR     R5,REPLZ        ;TYPE IT
                                        .WORD   3               ;TYPE 3 DIGITS
                                        TYPE    ,LINSP          ;SPACES
                                        MOV     R2,-(SP)        ;PUT $OPERC ON THE STACK
                                        JSR     PC,$DB2D        ;CONVERT IT
                                        JSR     R5,REPLZ        ;TYPE $OPERC
                                        .WORD   6               ;TYPE 6 DIGITS
                                        TYPE    ,LINSP          ;SPACES
                                        ADD     #4,R2           ;INCREMENT R2
                                        MOV     R2,-(SP)        ;PUT $POSIT ON THE STACK
                                        JSR     PC,$DB2D        ;CONVERT IT
                                        JSR     R5,REPLZ        ;TYPE $POSIT
                                        .WORD   6               ;TYPE 6 DIGITS
                                        TYPE    ,LINSP          ;SPACES
                                        ADD     #4,R2           ;INCREMENT R2
                                        MOV     R2,-(SP)        ;PUT $TRANS ON THE STACK
                                        JSR     PC,$DB2D        ;CONVERT $TRANS
                                        JSR     R5,REPLZ        ;TYPE IT
                                        .WORD   10.             ;TYPE 10 DIGITS
                                        TYPE    ,LINSP          ;SPACES
                                        ADD     #4,R2           ;INCREMENT R2
```

```
5630   023106  010246
5631   023110  004737  032562          MOV     R2,-(SP)        ;PUT $READ ON THE STACK
5632   023114  004537  027300          JSR     PC,$DB2D        ;CONVERT $READ
5633   023120  000012                  JSR     R5,REPLZ        ;TYPE IT
5634   023122  104401  052156          .WORD   10.             ;TYPE 10 DIGITS
5635   023126  062702  000004          TYPE    ,LINSPO         ;1 SPACE
5636   023132  062702  000002          ADD     #4,R2           ;INCREMENT R2
5637   023136  012246                  ADD     #2,R2           ;INCREMENT R2 AGAIN
5638   023140  004737  027770          MOV     (R2)+,-(SP)     ;PUT $SOFT ON THE STACK
5639   023144  004537  027300          JSR     PC,$SB2D        ;CONVERT $SOFT
5640   023150  000004                  JSR     R5,REPLZ        ;TYPEOUT $SOFT
5641   023152  104401  052156          .WORD   4               ;TYPE 4 DIGITS
5642   023156  012246                  TYPE    ,LINSPO         ;SPACES
5643   023160  004737  027770          MOV     (R2)+,-(SP)     ;PUT $HARD ON THE STACK
5644   023164  004537  027300          JSR     PC,$SB2D        ;CONVERT $HARD
5645   023170  000004                  JSR     R5,REPLZ        ;TYPEOUT $HARD
5646   023172  104401  052156          .WORD   4               ;TYPE 4 DIGITS
5647   023176  012246                  TYPE    ,LINSPO         ;SPACES
5648   023200  004737  027770          MOV     (R2)+,-(SP)     ;PUT $SKI ON THE STACK
5649   023204  004537  027300          JSR     PC,$SB2D        ;CONVERT $SKI
5650   023210  000004                  JSR     R5,REPLZ        ;TYPEOUT $SKI
5651   023212  104401  052156          .WORD   4               ;TYPE 4 DIGITS
5652   023216  012246                  TYPE    ,LINSPO         ;SPACES
5653   023220  004737  027770          MOV     (R2)+,-(SP)     ;PUT $MISPO ON THE STACK
5654   023224  004537  027300          JSR     PC,$SB2D        ;CONVERT $MISPO
5655   023230  000004                  JSR     R5,REPLZ        ;TYPEOUT $MISPO
5656   023232  104401  052156          .WORD   4               ;TYPE 4 DIGITS
5657   023236  016046  000056          TYPE    ,LINSPO         ;SPACES
5658   023242  166016  000060          MOV     $TOTAL(R0),-(SP)  ;CALCULATE NUMBER OF OTHER ERRORS
5659   023246  166016  000062          SUB     $SOFT(R0),(SP)  ;SUBTRACT $SOFT FROM $TOTAL
5660   023252  166016  000064          SUB     $HARD(R0),(SP)  ;SUBTRACT $HARD FROM $TOTAL
5661   023256  166016  000066          SUB     $SKI(R0),(SP)   ;SUBTRACT $SKI FROM $TOTAL
5662   023262  004737  027770          SUB     $MISPO(R0),(SP) ;SUBTRACT $MISPO FROM $TOTAL
5663   023266  004537  027300          JSR     PC,$SB2D        ;CONVERT 'OTHER' COUNT
5664   023272  000004                  JSR     R5,REPLZ        ;TYPE IT
5665   023274  104401  001165          .WORD   4               ;TYPE 4 DIGITS
5666   023300  012602                  TYPE    ,$CRLF
5667   023302  000207                  MOV     (SP)+,R2        ;RESTORE R2
5668                                   RTS     PC
5669
5670
5671                                   ;ROUTINE TO INCREMENT $SOFT
5672                                   ;
5673                                   ;NOTE:  $SOFT WILL NOT BE INCREMENTED BEYOND 9999 (10)
5674
5675   023304  005737  001264          INCSOF: TST     BADSEC          ;SEE IF BAD TRK/SEC INDICATOR SET
5676   023310  001006                  BNE     1$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
5677   023312  026027  000060  023417  CMP     $SOFT(R0),#9999.  ;IS $SOFT ALREADY AT MAXIMUM ?
5678   023320  103002                  BHIS    1$              ;BR IF IT IS
5679   023322  005260  000060          INC     $SOFT(R0)               ;INCREMENT $SOFT
5680   023326  000207                  1$:     RTS     PC              ;RETURN
5681
5682
5683                                   ;ROUTINE TO INCREMENT $HARD
5684                                   ;
5685                                   ;NOTE:  $HARD WILL NOT BE INCREMENTED BEYOND 9999 (10)
```

```
                                ;ROUTINE TO INCREMENT $SKI
                                ;
                                ;NOTE:  $SKI WILL NOT BE INCREMENTED BEYOND 9999 (10)

023354  005737  001264          INCSKI: TST     BADSEC          ;SEE IF BAD TRK/SEC INDICATOR SET
023360  001006                          BNE     1$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
023362  026027  000064  023417          CMP     $SKI(R0),#9999. ;IS $SKI ALREADY AT MAXIMUM ?
023370  103002                          BHIS    1$              ;BR IF IT IS
023372  005260  000064                  INC     $SKI(R0)                ;INCREMENT $SKI
023376  000207                  1$:     RTS     PC              ;RETURN


                                ;ROUTINE TO INCREMENT $MISPO
                                ;
                                ;NOTE:  $MISPO WILL NOT BE INCREMENTED BEYOND 9999 (10)

023400  005737  001264          INCMIS: TST     BADSEC          ;SEE IF BAD TRK/SEC INDICATOR SET
023404  001006                          BNE     1$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
023406  026027  000066  023417          CMP     $MISPO(R0),#9999.       ;IS $MISPO ALREADY AT MAXIMUM ?
023414  103002                          BHIS    1$              ;BR IF IT IS
023416  005260  000066                  INC     $MISPO(R0)              ;INCREMENT $MISPO
023422  000207                  1$:     RTS     PC              ;RETURN


                                ;ROUTINE TO INCREMENT $TOTAL
                                ;
                                ;NOTE:  $TOTAL WILL NOT BE INCREMENTED BEYOND 9999 (10)

023424  005737  001264          INCTOT: TST     BADSEC          ;SEE IF BAD TRK/SEC INDICATOR SET
023430  001006                          BNE     1$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
023432  026027  000056  023417          CMP     $TOTAL(R0),#9999.       ;IS $TOTAL ALREADY AT MAXIMUM ?
023440  103002                          BHIS    1$              ;BR IF IT IS
023442  005260  000056                  INC     $TOTAL(R0)              ;INCREMENT $TOTAL
023446  000207                  1$:     RTS     PC              ;RETURN


                                ;ROUTINE TO TYPE THE TIME

023450  005737  001210          $TIME:  TST     CLKFLG          ;CLOCK ON THE SYSTEM ?
023454  001033                          BNE     1$              ;BR IF NOT
023456  104401  001165                  TYPE    ,$CRLF          ;CR-LF
023462  013746  001266                  MOV     HOUR,-(SP)      ;PUT 'HOURS' ON THE STACK
023466  004737  027770                  JSR     PC,$$B2D        ;CONVERT TO DECIMAL
023472  004537  027300                  JSR     R5,REPLZ        ;TYPE IT
023476  000002                          .WORD   2               ;TYPE 2 DIGITS
023500  104401  053207                  TYPE    ,COLON          ;':'
023504  013746  001270                  MOV     MINUTE,-(SP)    ;PUT 'MINUTES' ON THE STACK
```

```
5742  023510  004737  027770          JSR     PC,$SB2D        ;CONVERT TO DECIMAL
5743  023514  004537  027300          JSR     R5,REPLZ        ;TYPE IT
5744  023520  000002                  .WORD   2               ;TYPE 2 DIGITS
5745  023522  104401  053207          TYPE    ,COLON          ;':'
5746  023526  013746  001272          MOV     SECOND,-(SP)    ;PUT SECONDS ON THE STACK
5747  023532  004737  027770          JSR     PC,$SB2D        ;CONVERT TO DECIMAL
5748  023536  004537  027300          JSR     R5,REPLZ        ;TYPE IT
5749  023542  000002                  .WORD   2               ;TYPE 2 DIGITS
5750  023544  000207          1$:     RTS     PC
5751
5752                                  ;CLOCK HANDLER ROUTINE
5753
5754  023546  005337  001274  CLOCK:  DEC     SIXTEE          ;INCREMENT THE 1/60 SECOND COUNTER
5755  023552  001035                  BNE     1$              ;BR IF A SECOND NOT COUNTED
5756  023554  013737  001212  001274  MOV     HZ,SIXTEE       ;RESTORE THE VALUE
5757  023562  005237  001272          INC     SECOND          ;COUNT THE SECOND
5758  023566  022737  000074  001272  CMP     #60.,SECOND     ;AT MAXIMUM ?
5759  023574  001024                  BNE     1$              ;BR IF NOT
5760  023576  005037  001272          CLR     SECOND          ;CLEAR THE SECOND'S COUNTER
5761  023602  005237  001412          INC     INTRVL+2        ;COUNT THE PERFORMANCE SUMMARY INTERVAL
5762  023606  005237  001270          INC     MINUTE          ;COUNT THE MINUTE
5763  023612  022737  000074  001270  CMP     #60.,MINUTE     ;AT MAXIMUM ?
5764  023620  001012                  BNE     1$              ;BR IF NOT
5765  023622  005037  001270          CLR     MINUTE          ;CLEAR THE MINUTE'S COUNTER
5766  023626  005237  001266          INC     HOUR            ;COUNT THE HOURS
5767  023632  022737  001747  001266  CMP     #999.,HOUR      ;AT MAXIMUM
5768  023640  103002                  BHIS    1$              ;BR IF NOT
5769  023642  005037  001266          CLR     HOUR            ;CLEAR THE HOURS
5770  023646  012746  000021  1$:     MOV     #17.,-(SP)      ;17 MS ON THE STACK
5771  023652  004737  037660          JSR     PC,RPTMR        ;DRIVER TIMER ROUTINE
5772  023656  005737  001410          TST     INTRVL          ;DISPLAY THE PERFORMANCE SUMMARY ?
5773  023662  001411                  BEQ     2$              ;BR IF NOT
5774  023664  023737  001410  001412  CMP     INTRVL,INTRVL+2 ;DISPLAY INTERVAL FINISHED ?
5775  023672  001005                  BNE     2$              ;BR IF NOT
5776  023674  012737  177777  001214  MOV     #-1,STATIN      ;SET PERFORMANCE SUMMARY DISPLAY FLAG
5777  023702  005037  001412          CLR     INTRVL+2        ;CLEAR THE PERFORMANCE INTERVAL COUNTER
5778  023706  000002          2$:     RTI                     ;RETURN
5779
5780                                  ;COMMAND DECODE ROUTINE
5781                                  ;CALL:
5782                                  ;       MOV     #-1,CFLAG       ;'CFLAG' IS NORMALLY SET BY THE TTY SERVICE
5783                                  ;                               ;ROUTINE IN INTERRUPT MODE
5784                                  ;       JSR     PC,KSR
5785                                  ;       RETURN1                 ;SYSTEM BUSY RETURN
5786                                  ;       RETURN2                 ;RETURN AFTER KEYBOARD SERVICED
5787
5788  023710  005737  001440  KSR:    TST     ORDERQ          ;ANY OPERATIONS ACTIVE ?
5789  023714  001003                  BNE     1$              ;BR IF SOME ARE
5790  023716  005037  023742          CLR     3$              ;CLEAR THE LOOP COUNTER
5791  023722  000410                  BR      KSR1            ;PROCESS THE KEYBOARD REQUEST
5792  023724  062737  000001  023742  1$:     ADD     #1,3$           ;COUNT THE TIMES THROUGH THE LOOP
5793  023732  001002                  BNE     2$              ;BR IF NOT ENOUGH
5794  023734  104401  053320          TYPE    ,BUSY           ;'SYSTEM BUSY...'
5795  023740  000207          2$:     RTS     PC              ;PROCESS ANY COMPLETED DRIVES
5796  023742  000000          3$:     .WORD   0               ;LOOP COUNTER
5797  023744  104412  KSR1:   SAVREG                          ;SAVE THE REGISTERS
```

```
5798  023746  012737  000200  177776        MOV     #PR4,PS         ;SET PRIORITY TO 4
5799  023754  005037  001262                CLR     CFLAG           ;CLEAR THE 'CONTROL C' FLAG
5800  023760  004737  023450                JSR     PC,$TIME        ;TYPE THE TIME
5801  023764  005777  155156                TST     @$TKB           ;CLEAR ANY GARBAGE IN THE TTY BUFFER
5802  023770  104401  053033                TYPE    ,ENTCOM         ;'ENTER COMMANDS'
5803  023774  104411                        RDLIN                   ;READ THE KEYBOARD
5804  023776  012605                        MOV     (SP)+,R5        ;GET ADDRESS OF INPUT STRING
5805  024000  005737  001262                TST     CFLAG           ;CHECK THE CONTROL C FLAG
5806  024004  001065                        BNE     7$              ;EXIT IF 'CONTROL C' ENTERED
5807  024006  005205                        INC     R5              ;POINT TO SECOND CHARACTER
5808  024010  122715  000101                CMPB    #'A,(R5)        ;EQ TO AN 'A'
5809  024014  001410                        BEQ     1$              ;BR IF IT IS
5810  024016  121527  000067                CMPB    (R5),#'7        ;DRIVE NUMBER GREATER THAN AN ASCII 7 ?
5811  024022  101054                        BHI     6$              ;BR IF IT IS
5812  024024  121527  000060                CMPB    (R5),#'0        ;DRIVE NUMBER LESS THAN AN ASCII 0 ?
5813  024030  103451                        BLO     6$              ;BR IF IT IS
5814  024032  142715  177770                BICB    #^C7,(R5)       ;LEAVE ONLY LOWER 3 BITS IF CHAR NOT 'A'
5815  024036  122765  000124  177777  1$:   CMPB    #'T,-1(R5)      ;EQ TO 'T'
5816  024044  001003                        BNE     2$              ;BR IF NOT EQ
5817  024046  004737  024552                JSR     PC,NEWASN       ;ASSIGN DRIVE FOR TEST
5818  024052  000442                        BR      7$              ;EXIT
5819  024054  122765  000104  177777  2$:   CMPB    #'D,-1(R5)      ;EQ TO 'D' ?
5820  024062  001003                        BNE     3$              ;BR IF NOT EQ
5821  024064  004737  024562                JSR     PC,DEASGN       ;DEASSIGN DRIVE
5822  024070  000433                        BR      7$              ;EXIT
5823  024072  122765  000123  177777  3$:   CMPB    #'S,-1(R5)      ;EQ TO 'S'
5824  024100  001003                        BNE     4$              ;BR IF NOT EQ
5825  024102  004737  024670                JSR     PC,SCMND        ;TYPE STATISTICS
5826  024106  000424                        BR      7$              ;EXIT
5827  024110  122765  000127  177777  4$:   CMPB    #'W,-1(R5)      ;EQ TO 'W'
5828  024116  001007                        BNE     5$              ;BR IF NOT EQ
5829  024120  032777  000001  155012        BIT     #SWD,@SWR       ;IS SWITCH 0 SET ?
5830  024126  001012                        BNE     6$              ;BR IF SET, CAN'T DO 'W' COMMAND
5831  024130  004737  025140                JSR     PC,DATAPK       ;WRITE A DATA PACK
5832  024134  000411                        BR      7$              ;EXIT
5833  024136  122765  000122  177777  5$:   CMPB    #'R,-1(R5)      ;EQ TO 'R' ?
5834  024144  001003                        BNE     6$              ;BR IF NOT EQ
5835  024146  004737  025152                JSR     PC,REDAPK       ;READ A DATA PACK
5836  024152  000402                        BR      7$              ;EXIT
5837  024154  104401  053011        6$:     TYPE    ,INVLD          ;TYPE 'INVALID COMMAND' MESSAGE
5838  024160  104413                7$:     RESREG                  ;RESTORE R0 - R5
5839  024162  062716  000002                ADD     #2,(SP)         ;INCREMENT THE RETURN ADDRESS
5840  024166  005777  154754                TST     @$TKB           ;CLEAR THE TTY BUFFER
5841  024172  052777  000100  154744        BIS     #BIT06,@$TKS    ;SET TTY INTERRUPT ENABLE
5842  024200  005037  177776                CLR     PS              ;SET PRIORITY BACK TO ZERO
5843  024204  000207                        RTS     PC              ;RETURN
5844
5845                                ;ROUTINE TO PROCESS THE ASSIGN REQUEST ('T', 'R', OR 'W' COMMANDS)
5846
5847  024206  122715  000101        ASSIGN: CMPB    #'A,(R5)        ;ASSIGN ALL DRIVES?
5848  024212  001430                        BEQ     ASGN2           ;BR IF ALL DRIVES
5849  024214  111504                ASGN1:  MOVB    (R5),R4         ;PUT DRIVE # IN R4
5850  024216  012737  052231  026436        MOV     #UNTASN,ASNMSG  ;'DRIVE ASSIGNED' MESSAGE ADDRESS
5851  024224  012703  000001                MOV     #1,R3           ;RELOAD R3 FOR 1 UNIT
5852  024230  136437  033320  001462        BITB    ATABIT(R4),ASNLST  ;DRIVE ALREADY ASSIGNED ?
5853  024236  001013                        BNE     2$              ;BR IF IT IS
```

```
5854  024240  005704                        TST     R4              ;TRYING TO ASSIGN DRIVE 0 ?
5855  024242  001007                        BNE     1$              ;BR IF NOT
5856  024244  012737  052315  026436        MOV     #NOTAVL,ASNMSG  ;'NOT AVAILABLE' MESSAGE ADDRESS
5857  024252  122737  000011  000041        CMPB    #11,41          ;SEE IF LOADED FROM AN RP0/4/5/6
5858  024260  001402                        BEQ     2$              ;BR IF RP04/5/6 IS THE LOAD DEVICE
5859  024262  004737  024346          1$:   JSR     PC,ASGN3        ;SEE IF DRIVE ON THE SYSTEM
5860  024266  000137  026416          2$:   JMP     ASNERR          ;RETURN HERE IF DRIVE NOT AVAIL
5861  024272  000207                        RTS     PC              ;EXIT
5862  024274  122737  000011  000041  ASGN2: CMPB    #11,41          ;LOADED FROM AN RP04/5/6 ?
5863  024302  001005                        BNE     1$              ;BR IF NOT
5864  024304  012704  000001                MOV     #1,R4           ;START WITH DRIVE 1
5865  024310  012703  000007                MOV     #7.,R3          ;SETUP FOR ONLY 7 DRIVES
5866  024314  000403                        BR      2$              ;CONTINUE
5867  024316  005004                  1$:   CLR     R4              ;START WITH DRIVE 0
5868  024320  012703  000010                MOV     #8.,R3          ;DRIVE COUNT FOR 8 DRIVES
5869  024324  004737  024346          2$:   JSR     PC,ASGN3        ;ASSIGN ALL UNASSIGNED, AVAIL DRIVES
5870  024330  000137  024336          3$:   JMP     4$              ;DRIVE NOT ON SYSTEM
5871  024334  000207                        RTS     PC              ;RETURN
5872  024336  012746  024330          4$:   MOV     #3$,-(SP)       ;PUT RETURN ADDRESS ON THE STACK
5873  024342  000137  024464                JMP     ASGN4           ;LOOK FOR MORE DRIVES
5874  024346  136437  033320  001462  ASGN3: BITB    ATABIT(R4),ASNLST  ;DRIVE ALREADY ASSIGNED ?
5875  024354  001043                        BNE     ASGN4           ;BR IF IT IS
5876  024356  005737  033316          1$:   TST     DTUW            ;DATA TRANSFER UNDER WAY ?
5877  024362  100375                        BPL     1$              ;BR IF IT IS
5878  024364  110437  044730                MOVB    R4,GENDPB       ;DRIVE NUMBER
5879  024370  004737  015244                JSR     PC,RECALO       ;RECALIBRATE DRIVE
5880  024374  105764  033204                TSTB    DRVSTA(R4)      ;DRIVE AVAILABLE?
5881  024400  001444                        BEQ     ASGN7           ;BR IF DRIVE OFFLINE OR NONEXISTENT
5882  024402  100437                        BMI     ASGN6           ;BR IF DRIVE UNSAFE
5883  024404  006304                        ASL     R4              ;MAKE R4 INTO WORD INDEX
5884  024406  016464  001740  001506        MOV     BLKADR(R4),NEWUNT(R4) ;DPB ADDRESS
5885  024414  016400  001740                MOV     BLKADR(R4),R0   ;PUT BLOCK'S ADDR INTO R0
5886  024420  004737  025164                JSR     PC,CLRDPB       ;CLEAR BLOCK FOR DRIVE JUST ASSIGNED
5887  024424  004737  025360                JSR     PC,DRVPRM       ;GET THE DRIVE'S ADDRESS LIMITS
5888  024430  004737  025642                JSR     PC,GETID        ;GET DRIVE I.D.
5889  024434  004737  025752                JSR     PC,GETADR       ;GET BAD SECTOR ADDRESSES
5890  024440  012760  000001  000070        MOV     #1,$PASSC(R0)   ;PRESET PASS COUNT TO 1
5891  024446  005737  001216                TST     PACK            ;WRITE DATA PACK ?
5892  024452  001403                        BEQ     2$              ;BR IF NOT
5893  024454  113760  001216  000026        MOVB    PACK,$PACK(R0)  ;SET READ/WRITE DATA PACK INDICATOR
5894  024462  006204                  2$:   ASR     R4              ;RESTORE DRIVE ADDRESS
5895  024464  005303            ASGN4: DEC     R3              ;DECREMENT DRIVE COUNT
5896  024466  001402                        BEQ     ASGN5           ;BR IF FINISHED
5897  024470  005204                        INC     R4              ;INCREMENT DRIVE NUMBER
5898  024472  000725                        BR      ASGN3           ;CONTINUE
5899  024474  062716  000004          ASGN5: ADD     #4,(SP)         ;INCREMENT RETURN
5900  024500  000207                        RTS     PC              ;RETURN
5901  024502  012737  052334  026436  ASGN6: MOV     #NOTSAF,ASNMSG  ;'UNSAFE' MESSAGE ADDRESS
5902  024510  000207                        RTS     PC              ;RETURN
5903  024512  105764  033214          ASGN7: TSTB    DRVTYP(R4)      ;DRIVE PRESENT?
5904  024516  001405                        BEQ     1$              ;BR IF NOT
5905  024520  100010                        BPL     2$              ;BR IF DRIVE OFFLINE
5906  024522  012737  052257  026436        MOV     #NOTRP,ASNMSG   ;ADDRESS OF 'NOT RP04/5/6' MSG
5907  024530  000407                        BR      3$              ;EXIT
5908  024532  012737  052300  026436  1$:   MOV     #NOTPRS,ASNMSG  ;ADDRESS OF 'NOT PRESENT' MSG
5909  024540  000403                        BR      3$              ;EXIT
```

```
5910  024542  012737  052166  026436  2$:     MOV     #UNTOFF,ASNMSG  ;ADDRESS OF 'DRIVE OFFLINE' MESSAGE
5911  024550  000207                  3$:     RTS     PC              ;ERROR RETURN
5912
5913                                  ;'T' COMMAND (ROUTINE TO ASSIGN A DRIVE)
5914
5915  024552  005037  001216          NEWASN: CLR     PACK            ;CLEAR 'W' COMMAND INDICATOR
5916  024556  000137  024206                  JMP     ASSIGN          ;GO TO THE ASSIGN ROUTINE
5917
5918                                  ;'D' COMMAND (ROUTINE TO DEASSIGN A DRIVE)
5919
5920  024562  005004                  DEASGN: CLR     R4
5921  024564  122715  000101                  CMPB    #'A,(R5)        ;DEASSIGN ALL DRIVES ?
5922  024570  001434                          BEQ     5$              ;BR IF YES
5923  024572  012703  000001                  MOV     #BIT00,R3       ;SET R3 FOR ONE UNIT
5924  024576  111504                          MOVB    (R5),R4         ;GET DRIVE NUMBER
5925  024600  136437  033320  001462  1$:     BITB    ATABIT(R4),ASNLST  ;DRIVE ASSIGNED ?
5926  024606  001414                          BEQ     3$              ;BR IF NOT
5927  024610  146437  033320  001462          BICB    ATABIT(R4),ASNLST  ;DELETE THE DRIVE FROM THE ASSIGNED LIST
5928  024616  006304                          ASL     R4              ;MAKE ADDR INTO A WORD INDEX
5929  024620  016464  001740  001464          MOV     BLKADR(R4),DUNIT(R4)  ;PUT ADDRESS IN DEASSIGN LIST
5930  024626  006204                          ASR     R4
5931  024630  005303                  2$:     DEC     R3              ;ANY MORE DRIVES ?
5932  024632  001412                          BEQ     4$              ;BR IF NOT
5933  024634  005204                          INC     R4
5934  024636  000760                          BR      1$
5935  024640  122715  000101          3$:     CMPB    #'A,(R5)        ;DEASSIGN ALL DRIVES ?
5936  024644  001771                          BEQ     2$              ;BR IF YES
5937  024646  012737  052207  026436          MOV     #UNTNOT,ASNMSG  ;ADDR OF 'NOT ASSIGNED' MESSAGE
5938  024654  004737  026416                  JSR     PC,ASNERR       ;REPORT IT
5939  024660  000207                  4$:     RTS     PC
5940  024662  012703  000010          5$:     MOV     #8.,R3          ;SET UNIT COUNT TO 8
5941  024666  000744                          BR      1$
5942
5943                                  ;'S' COMMAND (ROUTINE TO TYPE DRIVE PERFORMANCE SUMMARY)
5944
5945  024670  005004                  SCMND:  CLR     R4
5946  024672  122715  000101                  CMPB    #'A,(R5)        ;ALL STATISTICS ?
5947  024676  001421                          BEQ     2$              ;BR IF YES
5948  024700  111504                          MOVB    (R5),R4         ;GET DRIVE NUMBER
5949  024702  136437  033320  001462          BITB    ATABIT(R4),ASNLST  ;SEE IF DRIVE ASSIGNED
5950  024710  001406                          BEQ     1$              ;BR IF NOT
5951  024712  006304                          ASL     R4              ;MAKE DRIVE ADDR INTO WORD INDEX
5952  024714  016400  001740                  MOV     BLKADR(R4),R0   ;ADDR OF BLOCK
5953  024720  004737  022676                  JSR     PC,TYPEST       ;TYPE DRIVE STATISTICS
5954  024724  000504                          BR      9$              ;EXIT
5955  024726  012737  052207  026436  1$:     MOV     #UNTNOT,ASNMSG  ;ADDR OF 'NOT ASSIGNED' MSG
5956  024734  004737  026416                  JSR     PC,ASNERR       ;TYPE ERROR MESSAGE
5957  024740  000476                          BR      9$              ;EXIT
5958  024742  012703  000010          2$:     MOV     #8.,R3          ;DRIVE COUNT
5959  024746  136437  033320  001462  3$:     BITB    ATABIT(R4),ASNLST  ;SEE IF DRIVE ASSIGNED
5960  024754  001004                          BNE     4$              ;BR IF YES
5961  024756  005204                          INC     R4              ;INCREMENT DRIVE ADDRESS
5962  024760  005303                          DEC     R3              ;DECREMENT COUNTER
5963  024762  001371                          BNE     3$              ;MORE TO CHECK
5964  024764  000464                          BR      9$              ;NONE ASSINGED, RETURN
5965  024766  004737  022614          4$:     JSR     PC,STATPR       ;TYPE ALL STATISTICS
```

```
                                        TSTB    DATE            ;SEE IF 'DATE' ENTERED
                                        BEQ     11$             ;BR IF NOT
                                        TYPE    ,DATEIS         ;'DATE: '
                                        TYPE    ,DATE           ;THE OPERATOR ENTERED DATE
                                11$:    TSTB    OPERID          ;SEE IF OPERATOR I.D. ENTERED
                                        BEQ     12$             ;BR IF NOT
                                        TYPE    ,IDIS           ;'OPERATOR I.D.: '
                                        TYPE    ,OPERID         ;THE OPERATOR I.D.
                                12$:    TYPE    ,HEDLIN         ;HEADER LINE
                                        MOV     #DRIVED+$DRVID,6$  ;DRIVE I.D. FIELD ADDRESS
                                        CLR     R4              ;DRIVE ADDRESS
                                        MOV     #8.,R3          ;COUNTER
                                5$:     BITB    ATABIT(R4),ASNLST  ;SEE IF DRIVE ASSIGNED
                                        BEQ     7$              ;BR IF NOT ASSIGNED
                                        MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
                                                                ;;TYPE DRIVE NUMBER
                                        TYPOS                   ;;GO TYPE--OCTAL ASCII
                                        .BYTE   2               ;;TYPE 2 DIGIT(S)
                                        .BYTE   0               ;;SUPPRESS LEADING ZEROS
                                        TYPE    ,LIN4SP         ;4 SPACES
                                        TSTB    @6$             ;SEE IF DRIVE I.D. ENTERED
                                        BNE     10$             ;BR IF DRIVE I.D. PRESENT
                                        TYPE    ,NONE           ;TYPE 'NONE'
                                        BR      7$              ;CONTINUE
                                10$:    TYPE                    ;TYPE THE DRIVE I.D.
                                6$:     .WORD   0               ;ADDRESS OF DRIVE I.D. FIELD HERE
                                        TYPE    ,$CRLF          ;CR-LF
                                7$:     DEC     R3              ;DECREMENT THE COUNTER
                                        BEQ     8$              ;BR IF AT END
                                        ADD     #$RPEC2+2,6$    ;INCREMENT THE MESSAGE FIELD ADDRESS
                                        INC     R4              ;INCREMENT DRIVE ADDRESS
                                        BR      5$              ;CONTINUE
                                8$:     TYPE    ,$CRLF          ;CR-LF
                                9$:     RTS     PC

                                ;'W' COMMAND (ROUTINE TO WRITE A DATA PACK)

                        001216  DATAPK: MOV     #-1,PACK        ;SET THE 'W' COMMAND INDICATOR
                                        JMP     ASSIGN          ;ASSIGN REQUESTED DRIVE

                                ;'R' COMMAND (ROUTINE TO READ A DATA PACK)

                        001216  REDAPK: MOV     #1,PACK         ;SET THE 'READ' INDICATOR
                                        JMP     ASSIGN          ;ASSIGN THE REQUESTED DRIVE

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

                                CLRDPB:
                                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
                                        MOV     R4,-(SP)        ;;PUSH R4 ON STACK
                                        MOV     R5,-(SP)        ;;PUSH R5 ON STACK
```

```
                                        ;GET THE DPB ADDRESS
                                        ;ADDRESS OF FIRST LOCN TO BE CLEARED
                                        ;NUMBER OF LOCNS TO BE CLEARED
                                1$:     ;CLEAR THE LOCATION
                                        ;DECREMENT THE COUNTER
                                        ;BR IF NOT FINISHED
                                        ;MOVE THE ADDRESS PAST THE 'REG' ADDR
                                        ;NUMBER OF LOCNS TO BE CLEARED
                                2$:     ;CLEAR
                                        ;DECREMENT THE LOCN COUNTER
                                        ;BR IF NOT FINISHED
                                        ;MOVE PAST ADDRESS LIMITS
                                        ;NUMBER OF LOCNS TO BE CLEARED
                                3$:     ;CLEAR A LOCATION
                                        ;DECREMENT THE COUNTER
                                        ;BR IF NOT DONE
                                        ;INITIAL COMMAND CODE
                                        ;GET THE ACTUAL OP CODE
                                        ;OPERATION CODE
                                        ;PATTERN CODE
                                        ;CONVERT CODE TO A TABLE INDEX
                                        ;BEGINNING RECORD SIZE
                                        ;VALUE FOR DATA TRANSFER
                                        ;MAKE IT INTO 2'S COMPLEMENT
                                        ;INITIAL VALUE OF SECTOR SIZE
                                        ;HEADER ORDER ?
                                        ;BR IF NOT
                                        ;ADD HEADER SIZE TO SECTOR SIZE
                                4$:
                                        ;;POP STACK INTO R5
                                        ;;POP STACK INTO R4
                                        ;;POP STACK INTO R3
                                        ;RETURN

                                ;ROUTINE TO GET ADDRESS LIMITS FROM THE OPERATOR

                                DRVPRM: ;SAVE R3
                                        ;SAVE R4
                                        ;RUNNING UNDER MONITOR CONTROL
                                        ;BR IF YES
                                        ;'ENTER ADDRESSES'
                                        ;CONVERT INDEX TO DRIVE NUMBER
                                        ;;SAVE R4 FOR TYPEOUT
                                        ;;TYPE DRIVE NUMBER
                                        ;;GO TYPE--OCTAL ASCII
                                        ;;TYPE 2 DIGIT(S)
                                        ;;SUPPRESS LEADING ZEROS
                                        ;/
                                        ;ADDRESS OF 'RP04' MESSAGE
                                        ;RP04 ?
                                        ;BR IF YES
                                        ;ADDRESS OF 'RP05' MESSAGE
                                        ;RP05 ?
                                        ;BR IF YES
                                        ;ADDRESS OF 'RP06' MESSAGE
                                1$:     ;TYPE THE MESSAGE WHICH FOLLOWS
```

```
6078  025456  000000            2$:     .WORD   0               ;MESSAGE ADDRESS
6079  025460  104401  001165            TYPE    ,$CRLF          ;CR-LF
6080  025464  012737  000632  001350  3$:     MOV     #410.,CYLIMT    ;ASSUME AN RP04/5
6081  025472  132764  000004  033214          BITB    #BIT02,DRVTYP(R4)   ;SEE IF RP06
6082  025500  001403                          BEQ     4$              ;BR IF NOT
6083  025502  012737  001456  001350          MOV     #814.,CYLIMT    ;CHANGE LIMIT TO 814
6084  025510  062760  177777  000122  4$:     ADD     #-1,$FIRST(R0)  ;SEE IF FIRST TIME STARTED
6085  025516  103417                          BCS     5$              ;BR IF NOT
6086  025520  013760  001350  000106          MOV     CYLIMT,MAXCYL(R0)   ;LOAD MAXIMUM CYLINDER
6087  025526  005060  000110                  CLR     MINCYL(R0)      ;CLEAR MINIMUM CYLINDER
6088  025532  013760  001346  000112          MOV     TRKLMT,MAXTRK(R0)   ;LOAD MAXIMUM TRACK
6089  025540  005060  000114                  CLR     MINTRK(R0)      ;CLEAR MINIMUM TRACK
6090  025544  013760  001344  000116          MOV     SECLMT,MAXSEC(R0)   ;LOAD MAXIMUM SECTOR
6091  025552  005060  000120                  CLR     MINSEC(R0)      ;CLEAR MINIMUM SECTOR
6092  025556  006304                  5$:     ASL     R4              ;SETUP TO ADDRESS WORDS
6093  025560  016403  053756                  MOV     TABLE(R4),R3    ;PARAMETER TABLE ADDRESS
6094  025564  013763  001350  000002          MOV     CYLIMT,2(R3)    ;LOAD CYLINDER LIMIT FOR LAST CYLINDER
6095  025572  013763  001350  000010          MOV     CYLIMT,10(R3)   ;LOAD CYLINDER LIMIT FOR STARTING CYLINDER
6096  025600  005737  000042                  TST     42              ;UNDER MONITOR CONTROL ?
6097  025604  001002                          BNE     6$              ;BR IF YES
6098  025606  004737  026272                  JSR     PC,PARENT       ;GET THE DRIVE'S PARAMETERS
6099  025612  116060  000120  000010  6$:     MOVB    MINSEC(R0),$SEC(R0)  ;INITIAL SECTOR VALUE
6100  025620  116060  000114  000011          MOVB    MINTRK(R0),$TRK(R0)  ;INITIAL TRACK VALUE
6101  025626  016060  000110  000012          MOV     MINCYL(R0),$CYL(R0)  ;INITIAL CYLINDER VALUE
6102  025634  012604                          MOV     (SP)+,R4        ;RESTORE R4
6103  025636  012603                          MOV     (SP)+,R3        ;RESTORE R3
6104  025640  000207                          RTS     PC              ;RETURN
6105
6106                          ;ROUTINE TO GET THE DRIVE I.D. FROM THE OPERATOR
6107
6108  025642  010546          GETID:  MOV     R5,-(SP)        ;SAVE R5
6109  025644  005737  000042                  TST     42              ;UNDER MONITOR CONTROL ?
6110  025650  001036                          BNE     2$              ;BR IF NOT
6111  025652  005037  001262          1$:     CLR     CFLAG           ;CLEAR THE 'CONTROL C' FLAG
6112  025656  104401  053060                  TYPE    ,ENTDRV         ;'ENTER DRV I.D.: '
6113  025662  005046                          CLR     -(SP)           ;CLEAR THE STACK
6114  025664  111016                          MOVB    (R0),(SP)       ;PUT THE DRIVE NUMBER ON THE STACK
6115  025666  104403                          TYPOS                   ;TYPE THE DRIVE NUMBER
6116  025670     002                          .BYTE   2               ;TYPE 2 DIGITS
6117  025671     000                          .BYTE   0               ;SUPRESS LEADING ZEROS
6118  025672  104401  001165                  TYPE    ,$CRLF          ;CR-LF
6119  025676  104411                          RDLIN                   ;READ THE ENTRY
6120  025700  012605                          MOV     (SP)+,R5        ;GET THE ENTRY ADDRESS
6121  025702  005737  001262                  TST     CFLAG           ;'CONTROL C' ENTERED ?
6122  025706  001361                          BNE     1$              ;BR IF IT WAS
6123  025710  121527  000056                  CMPB    (R5),#'.        ;PERIOD ENTERED ?
6124  025714  001414                          BEQ     2$              ;BR IF YES
6125  025716  112560  000224                  MOVB    (R5)+,$DRVID(R0)    ;STORE THE I.D.
6126  025722  112560  000225                  MOVB    (R5)+,$DRVID+1(R0)  ;STORE THE I.D.
6127  025726  112560  000226                  MOVB    (R5)+,$DRVID+2(R0)  ;STORE THE I.D.
6128  025732  112560  000227                  MOVB    (R5)+,$DRVID+3(R0)  ;STORE THE I.D.
6129  025736  112560  000230                  MOVB    (R5)+,$DRVID+4(R0)  ;STORE THE I.D.
6130  025742  112560  000231                  MOVB    (R5)+,$DRVID+5(R0)  ;STORE THE I.D.
6131  025746  012605          2$:     MOV     (SP)+,R5        ;RESTORE R5
6132  025750  000207                  RTS     PC              ;RETURN
6133
```

```
                                ;ROUTINE TO GET THE ADDRESSES OF ANY BAD SECTORS (UP TO A MAX OF 16)

                                GETADR:
025752 010146                           MOV     R1,-(SP)        ;;PUSH R1 ON STACK
025754 010246                           MOV     R2,-(SP)        ;;PUSH R2 ON STACK
025756 010346                           MOV     R3,-(SP)        ;;PUSH R3 ON STACK
025760 010446                           MOV     R4,-(SP)        ;PUSH R4 ON STACK
025762 005737 000042                    TST     42              ;UNDER MONITOR CONTROL ?
025766 001012                           BNE     15$             ;BR IF YES
025770 005037 001262            14$:    CLR     CFLAG           ;CLEAR 'CONTROL C' FLAG
025774 104401 053146                    TYPE    ,ENTADR         ;ENTER SECTOR ADDRESSES
026000 005046                           CLR     -(SP)           ;CLEAR THE STACK
026002 111016                           MOVB    (R0),(SP)       ;PUT THE DRIVE NUMBER ON THE STACK
026004 104403                           TYPOS                   ;TYPE THE DRIVE NUMBER
026006     002                          .BYTE   2               ;TYPE 2 DIGITS
026007     000                          .BYTE   0               ;SUPRESS LEADING ZEROS
026010 104401 001165                    TYPE    ,$CRLF          ;CR-LF
026014 012703 000040            15$:    MOV     #32.,R3         ;NUMBER OF LOCATIONS IN THE TABLE TO PRESET
026020 012704 000124                    MOV     #$BDSEC,R4      ;TABLE INCREMENT
026024 060004                           ADD     R0,R4           ;BLOCK STARTING ADDRESS
026026 012724 177777            1$:     MOV     #-1,(R4)+       ;SET LOCATION TO 1'S
026032 005303                           DEC     R3              ;DECREMENT TABLE SIZE COUNT
026034 001374                           BNE     1$              ;BR IF NOT FINISHED WITH TABLE
026036 005737 000042                    TST     42              ;UNDER MONITOR CO TROL ?
026042 001106                           BNE     13$             ;BR IF YES
026044 162704 000100                    SUB     #64.,R4         ;SET POINTER TO BEGINNING OF TABLE
026050 012703 000020                    MOV     #16.,R3         ;NUMBER OF ADDRESSES IN TABLE
026054 104411                   2$:     RDLIN                   ;GET ADDRESS FROM OPERATOR
026056 012601                           MOV     (SP)+,R1        ;TEXT POINTER
026060 005737 001262                    TST     CFLAG           ;'CONTROL C' ENTERED ?
026064 001341                           BNE     14$             ;BR IF IT WAS
026066 013702 001350                    MOV     CYLIMT,R2       ;UPPER LIMIT OF INPUT
026072 004537 027632                    JSR     R5,CK.DIG       ;CHECK THE DIGIT(S)
026076 026240                           12$                     ;CARRIAGE RETURN ONLY ENTERED
026100 026260                           13$                     ;PERIOD ONLY ENTERED
026102 026240                           12$                     ;ILLEGAL INPUT
026104 026116                           4$                      ;TERMINATED WITH A CARRIAGE RETURN
026106 026122                           5$                      ;TERMINATED WITH A ";"
026110 026112                           3$                      ;TERMINATED WITH A "."
026112 010214                   3$:     MOV     R2,(R4)         ;CYLINDER ADDRESS
026114 000461                           BR      13$             ;EXIT, PERIOD ENTERED
026116 010214                   4$:     MOV     R2,(R4)         ;CYLINDER ADDRESS
026120 000442                           BR      11$             ;FINISHED WITH THIS ADDRESS, 'CR' ENTERED
026122 010214                   5$:     MOV     R2,(R4)         ;CYLINDER ADDRESS
026124 013702 001346                    MOV     TRKLMT,R2       ;UPPER LIMIT OF INPUT
026130 004537 027632                    JSR     R5,CK.DIG       ;CHECK THE DIGIT(S)
026134 026240                           12$                     ;CARRIAGE RETURN ONLY ENTERED
026136 026260                           13$                     ;PERIOD ONLY ENTERED
026140 026240                           12$                     ;ILLEGAL INPUT
026142 026156                           7$                      ;TERMINATED WITH A CARRIAGE RETURN
026144 026164                           8$                      ;TERMINATED WITH A ";"
026146 026150                           6$                      ;TERMINATED WITH A "."
026150 110264 000003            6$:     MOVB    R2,3(R4)        ;TRACK ADDRESS
026154 000441                           BR      13$             ;EXIT, ENTRY TERMINATED BY PERIOD
026156 110264 000003            7$:     MOVB    R2,3(R4)        ;TRACK ADDRESS
026162 000421                           BR      11$             ;FINISHED WITH THIS ADDRESS, 'CR' ENTERED
```

```
6190  026164  110264  000003          8$:     MOVB    R2,3(R4)        ;TRACK ADDRESS
6191  026170  013702  001344                  MOV     SECLMT,R2       ;UPPER LIMIT OF INPUT
6192  026174  004537  027632                  JSR     R5,CK.DIG       ;CHECK THE DIGIT(S)
6193  026200  026240                          12$                     ;CARRIAGE RETURN ONLY ENTERED
6194  026202  026260                          13$                     ;PERIOD ONLY ENTERED
6195  026204  026240                          12$                     ;ILLEGAL INPUT
6196  026206  026222                          10$                     ;TERMINATED WITH A CARRIAGE RETURN
6197  026210  026240                          12$                     ;TERMINATED WITH A ":"
6198  026212  026214                          9$                      ;TERMINATED WITH A "'"
6199  026214  110264  000002          9$:     MOVB    R2,2(R4)        ;SECTOR ADDRESS
6200  026220  000417                  BR      13$                     ;EXIT, ENTRY TERMINATED BY PERIOD
6201  026222  110264  000002          10$:    MOVB    R2,2(R4)        ;SECTOR ADDRESS
6202  026226  005303                  11$:    DEC     R3              ;MORE ENTRIES ?
6203  026230  001413                          BEQ     13$             ;BR IF NOT
6204  026232  062704  000004                  ADD     #4,R4           ;INCREMENT THE TABLE POINTER
6205  026236  000706                  BR      2$                      ;CONTINUE
6206  026240  012714  177777          12$:    MOV     #-1,(R4)        ;CLEAR PRESENT TABLE ENTRY
6207  026244  012764  177777  000002        MOV     #-1,2(R4)       ;CLEAR PRESENT TABLE ENTRY
6208  026252  104401  053276                  TYPE    ,BADENT         ;'INVALID ENTRY'
6209  026256  000676                  BR      2$                      ;TRY AGAIN
6210  026260                          13$:
6211  026260  012604                  MOV     (SP)+,R4                ;;POP STACK INTO R4
6212  026262  012603                  MOV     (SP)+,R3                ;;POP STACK INTO R3
6213  026264  012602                  MOV     (SP)+,R2                ;;POP STACK INTO R2
6214  026266  012601                  MOV     (SP)+,R1                ;;POP STACK INTO R1
6215  026270  000207                  RTS     PC                      ;RETURN
6216
6217                                  ;PARAMETER ENTRY ROUTINE
6218                                  ;CALL
6219                                  ;       MOV     #ADR,R3         ;PARAMETER TABLE ADDRESS
6220                                  ;       JSR     PC,PARENT       ;GET THE PARAMETERS
6221
6222  026272  010346                  PARENT: MOV     R3,-(SP)        ;SAVE THE PARAMETER TABLE ADDRESS
6223  026274  005037  001262                  CLR     CFLAG           ;CLEAR THE 'CONTROL C' FLAG
6224  026300  012337  026310          1$:     MOV     (R3)+,3$        ;ADDRESS OF PARAMETER NAME
6225  026304  001442                          BEQ     9$              ;BR IF AT END OF TABLE
6226  026306  104401                          TYPE                    ;TYPE THE PARAMETER NAME
6227  026310  000000                  3$:     .WORD   0               ;ADDRESS OF PARAMETER NAME TEXT
6228  026312  012302                          MOV     (R3)+,R2        ;MAXIMUM PARAMETER VALUE
6229  026314  012305                          MOV     (R3)+,R5        ;ADDRESS OF PARAMETER
6230  026316  011546                          MOV     (R5),-(SP)      ;CURRENT VALUE OF PARAMETER
6231  026320  104405                          TYPDS                   ;TYPE THE CURRENT VALUE OF THE PARAMETER
6232  026322  104401  053576                  TYPE    ,SLASH          ;' /'
6233  026326  104411                          RDLIN                   ;READ THE KEYBOARD
6234  026330  012601                          MOV     (SP)+,R1        ;INPUT ASCII STRING ADDRESS
6235  026332  005737  001262                  TST     CFLAG           ;'CONTROL C' ENTERED ?
6236  026336  001021                          BNE     8$              ;BR IF IT WAS
6237  026340  004537  027632                  JSR     R5,CK.DIG       ;CHECK THE DIGIT(S)
6238  026344  026300                          1$                      ;CARRIAGE RETURN ONLY ENTERED
6239  026346  026412                          9$                      ;PERIOD ONLY ENTERED
6240  026350  026364                          6$                      ;ILLEGAL INPUT
6241  026352  026360                          5$                      ;TERMINATED WITH A CARRIAGE RETURN
6242  026354  026364                          6$                      ;TERMINATED WITH A ":"
6243  026356  026376                          7$                      ;TERMINATED WITH A "'"
6244  026360  010215                  5$:     MOV     R2,(R5)         ;MOVE NEW VALUE TO PARAMETER LOCATION
6245  026362  000746                  BR      1$                      ;GET MORE PARAMETERS
```

```
 6246  026364  104401  053276          6$:     TYPE    ,BADENT         ;'BAD ENTRY'
 6247  026370  162703  000006                  SUB     #6,R3           ;DECREMENT THE TABLE POINTER
 6248  026374  000741                          BR      1$              ;TRY AGAIN
 6249  026376  010215                  7$:     MOV     R2,(R5)         ;NEW VALUE
 6250  026400  000404                          BR      9$              ;EXIT
 6251  026402  005037  001262          8$:     CLR     CFLAG           ;CLEAR THE 'CONTROL C' FLAG
 6252  026406  011603                          MOV     (SP),R3         ;RELOAD THE PARAMETER TABLE ADDRESS
 6253  026410  000733                          BR      1$              ;TRY AGAIN
 6254  026412  005726                  9$:     TST     (SP)+           ;CORRECT THE STACK POINTER
 6255  026414  000207                          RTS     PC              ;RETURN
 6256
 6257                                  ;TYPEOUT ASSIGN/DEASSIGN ERROR MESSAGE
 6258                                  ;CALL
 6259                                  ;       MOV     #MESADR,ASNMSG  ;ERROR MESSAGE ADDRESS
 6260                                  ;       JSR     PC,ASNERR
 6261                                  ;       RETURN
 6262
 6263  026416  104401  053007          ASNERR: TYPE    ,QUES           ;QUESTION MARK
 6264  026422  104401  052160                  TYPE    ,UNTMSG         ;TYPE 'DRIVE'
 6265  026426  010446                          MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
 6266                                                                  ;;TYPE DRIVE NUMBER
 6267  026430  104403                          TYPOS                   ;;GO TYPE--OCTAL ASCII
 6268  026432     002                          .BYTE   2               ;;TYPE 2 DIGIT(S)
 6269  026433     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
 6270  026434  104401                          TYPE                    ;TYPE SPECIFIC MESSAGE
 6271  026436  000000          ASNMSG: .WORD   0                       ;MESSAGE ADDRESS
 6272  026440  104401  001165            TYPE    ,$CRLF
 6273  026444  000207                    RTS     PC
 6274
 6275                                  ;DEASSIGN DRIVE IF A FATAL ERROR OCCURS
 6276                                  ;CALL
 6277                                  ;       JSR     PC,DROP
 6278                                  ;       RETURN
 6279
 6280  026446  005004          DROP:   CLR     R4              ;CLEAR R4 FOR DRIVE NUMBER
 6281  026450  111004                  MOVB    (R0),R4         ;MOVE DRIVE NUMBER TO R4
 6282  026452  146437  033320  001462        BICB    ATABIT(R4),ASNLST+  ;REMOVE DRIVE FROM ASSIGNED LIST
 6283  026460  006304                  ASL     R4              ;MAKE DRIVE NUMBER INTO A TABLE INDEX
 6284  026462  010064  001464          MOV     R0,DUNIT(R4)    ;PUT DRIVE IN DROP LIST
 6285  026466  104401  001165          TYPE    ,$CRLF
 6286  026472  104401  001165          TYPE    ,$CRLF
 6287  026476  104401  052562          TYPE    ,DROPNG         ;TYPE 'DROPPING DRIVE'
 6288  026502  104401  052673          TYPE    ,DRNUM          ;'DRIVE #'
 6289  026506  006204                  ASR     R4              ;DRIVE NUMBER
 6290  026510  010446                  MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
 6291                                                          ;;TYPE DRIVE NUMBER
 6292  026512  104403                  TYPOS                   ;;GO TYPE--OCTAL ASCII
 6293  026514     002                  .BYTE   2               ;;TYPE 2 DIGIT(S)
 6294  026515     000                  .BYTE   0               ;;SUPPRESS LEADING ZEROS
 6295  026516  104401  001165          TYPE    ,$CRLF
 6296  026522  000207                  RTS     PC
 6297
 6298                                  ;ROUTINE TO DEASSIGN DRIVE IF ERRORS BECOMES EXCESSIVE
 6299
 6300  026524  032777  000020  152406  ABNRML: BIT     #SW04,@SWR      ;SEE IF SWITCH 4 SET
 6301  026532  001006                  BNE     1$              ;BR IF IT'S SET
```

```
6302  026534  023760  001406  000056          CMP     MAXER,$TOTAL(R0)  ;CHECK TOTAL ERROR VALUE
6303  026542  103002                          BHIS    1$                ;BR IF ERRORS DONOT EXCEED MAX
6304  026544  000137  026446                  JMP     DROP              ;DEASSING THE DRIVE
6305  026550  000207                  1$:     RTS     PC                ;RETURN
6306
6307                                          ;ROUTINE TO CHECK FOR END OF PASS AND END OF TEST
6308
6309  026552  005737  001430          EOP:    TST     ENDET             ;END OF PASS DETERMINED BY SEEKS OR WORDS ?
6310  026556  001412                          BEQ     EOP1              ;BR IF SEEKS
6311  026560  026037  000054  001374          CMP     $READ+2(R0),ENDCON+2  ;CHECK MSW OF WORDS READ COUNT
6312  026566  101017                          BHI     EOP2              ;BR IF MSW GREATER THAN LIMIT
6313  026570  103405                          BLO     EOP1              ;BR IF MSW LESS THAN LIMIT
6314  026572  026037  000052  001372          CMP     $READ(R0),ENDCON  ;CHECK LSW AGAINST LIMIT
6315  026600  103012                          BHIS    EOP2              ;BR IF EQUAL OR GREATER
6316  026602  000510                          BR      EOPX              ;EXIT
6317  026604  026037  000044  001400  EOP1:   CMP     $POSIT+2(R0),ENDSEK+2  ;CHECK MSW OF SEEK COUNT
6318  026612  101005                          BHI     EOP2              ;BR IF MSW GREATER THAN LIMIT
6319  026614  103503                          BLO     EOPX              ;EXIT IF MSW LESS THAN LIMIT
6320  026616  026037  000042  001376          CMP     $POSIT(R0),ENDSEK  ;CHECK LSW OF SEEK COUNT
6321  026624  103477                          BLO     EOPX              ;EXIT IF LSW LESS THAN LIMIT
6322  026626  104401  001165          EOP2:   TYPE    ,$CRLF            ;CR-LF
6323  026632  104401  052616                  TYPE    ,ENDPAS           ;END OF PASS FOR THE DRIVE
6324  026636  016046  000070                  MOV     $PASSC(R0),-(SP)  ;PUT PASS COUNT ON THE STACK
6325  026642  104405                          TYPDS                     ;CONVERT PASS COUNT TO DECIMAL AND TYPE IT
6326  026644  111037  001246                  MOVB    (R0),UNIT         ;STORE THE DRIVE NUMBER
6327  026650  032777  000020  152262          BIT     #SW04,@SWR        ;SWITCH 4 SET ?
6328  026656  001017                          BNE     1$                ;BR IF SET
6329  026660  026037  000070  001402          CMP     $PASSC(R0),PASCNT  ;SEE IF AT END OF TEST
6330  026666  103413                          BLO     1$                ;BR IF NOT
6331  026670  104401  052632                  TYPE    ,ENDTST           ;TYPE 'END OF TEST'
6332  026674  104401  052673                  TYPE    ,DRNUM            ;'DRIVE #'
6333  026700  013746  001246                  MOV     UNIT,-(SP)        ;;SAVE UNIT FOR TYPEOUT
6334                                                                    ;;TYPE DRIVE NUMBER
6335  026704  104403                          TYPOS                     ;;GO TYPE--OCTAL ASCII
6336  026706     002                          .BYTE   2                 ;;TYPE 2 DIGIT(S)
6337  026707     000                          .BYTE   0                 ;;SUPPRESS LEADING ZEROS
6338  026710  104401  001165                  TYPE    ,$CRLF            ;CR-LF
6339  026714  000431                          BR      3$                ;DEASSIGN THE DRIVE
6340  026716  104401  052673          1$:     TYPE    ,DRNUM            ;'DRIVE #'
6341  026722  013746  001246                  MOV     UNIT,-(SP)        ;;SAVE UNIT FOR TYPEOUT
6342                                                                    ;;TYPE DRIVE NUMBER
6343  026726  104403                          TYPOS                     ;;GO TYPE--OCTAL ASCII
6344  026730     002                          .BYTE   2                 ;;TYPE 2 DIGIT(S)
6345  026731     000                          .BYTE   0                 ;;SUPPRESS LEADING ZEROS
6346  026732  104401  001165                  TYPE    ,$CRLF            ;CR-LF
6347  026736  004737  022676                  JSR     PC,TYPEST         ;TYPE THE DRIVE'S STATISTICS
6348  026742  010346                          MOV     R3,-(SP)          ;SAVE R3
6349  026744  010446                          MOV     R4,-(SP)          ;SAVE R4
6350  026746  010004                          MOV     R0,R4             ;DRIVE'S BLOCK ADDRESS
6351  026750  062704  000036                  ADD     #$OPERC,R4        ;ADD THE STARTING ADDR OF SECTIONS TO CLEAR
6352  026754  012703  000010                  MOV     #8.,R3            ;NUMBER OF LOCNS TO BE CLEARED
6353                                                                    ;(ERROR COUNTERS NOT CLEARED)
6354  026760  005024                  2$:     CLR     (R4)+             ;CLEAR THE LOCN
6355  026762  005303                          DEC     R3                ;DECREMENT THE LOCATION COUNTER
6356  026764  001375                          BNE     2$                ;BR IF MORE TO GO
6357  026766  012604                          MOV     (SP)+,R4          ;RESTORE R4
```

```
6358  026770  012603                        MOV     (SP)+,R3        ;RESTORE R3
6359  026772  005260  000070                INC     $PASSC(R0)      ;INCREMENT THE PASS COUNT
6360  026776  000412                        BR      EOPX            ;EXIT
6361  027000  104401  001165        3$:     TYPE    $CRLF
6362  027004  005004                        CLR     R4              ;CLEAR R4 FOR DRIVE NUMBER
6363  027006  111004                        MOVB    (R0),R4         ;MOVE DRIVE NUMBER
6364  027010  146437  033320  001462        BICB    ATABIT(R4),ASNLST  ;DELETE DRIVE FROM ASSIGNED LIST
6365  027016  006304                        ASL     R4              ;MAKE DRIVE NUMBER INTO TABLE INDEX
6366  027020  010064  001464                MOV     R0,DUNIT(R4)    ;PUT BLOCK ADDRESS INTO DROP LIST
6367  027024  000207                EOPX:   RTS     PC              ;RETURN
6368
6369                                 ;ROUTINE TO GET THE REMAINDER OF THE RANDOM NUMBER
6370                                 ;CALL
6371                                 ;       MOV     NUMBER,R5       ;DIVISOR INTO R5
6372                                 ;       JSR     PC,GETREM
6373                                 ;       RETURN                  ;REMAINDER IS IN R5
6374
6375  027026  013746  032464        GETREM: MOV     $LONUM,-(SP)    ;STORE RANDOM NUMBER ON THE STACK FOR DIVIDE
6376  027032  013746  032462                MOV     $HINUM,-(SP)    ;UPPER PART
6377  027036  010546                        MOV     R5,-(SP)        ;PUT THE DIVISOR ONTO THE STACK
6378  027040  004737  027054                JSR     PC,LINKDV       ;DIVIDE THE RANDOM NUMBERS
6379  027044  012605                        MOV     (SP)+,R5        ;PUT THE REMAINDER INTO R5
6380  027046  005726                        TST     (SP)+           ;ADJUST THE STACK POINTER
6381  027050  000240                        NOP                     ;FOR DEBUGGING HALT
6382  027052  000207                        RTS     PC
6383
6384                                 ;LINK ROUTINE TO THE DIVISION UTILITY SUBROUTINE
6385                                 ;       THIS ROUTINE ALLOWS THE 'SYSMAC' DIVIDE ROUTINE
6386                                 ;       CALLING SEQUENCE TO BE USED
6387
6388  027054  104412                LINKDV: SAVREG                  ;STORE R0 - R5
6389  027056  016605  000026                MOV     26(SP),R5       ;DIVISOR
6390  027062  005004                        CLR     R4              ;OTHER DIVISOR WORD
6391  027064  016602  000030                MOV     30(SP),R2       ;UPPER DIVIDEND WORD
6392  027070  016603  000032                MOV     32(SP),R3       ;LOWER DIVIDEND WORD
6393  027074  005000                        CLR     R0              ;CLEAR OTHER DIVIDEND REGISTERS
6394  027076  005001                        CLR     R1
6395  027100  004737  027122                JSR     PC,M.DPID       ;GO TO THE DIVIDE ROUTINE
6396  027104  010166  000030                MOV     R1,30(SP)       ;REMAINDER ON THE STACK
6397  027110  010366  000032                MOV     R3,32(SP)       ;QUOTIENT ON THE STACK
6398  027114  104413                        RESREG                  ;RESTORE R0 - R5
6399  027116  012616                        MOV     (SP)+,(SP)      ;MOVE RETURN UP THE STACK
6400  027120  000207                        RTS     PC
6401
6402                                 ;       DIVISION UTILITY SUBROUTINE
6403                                 ;       R0-R1-R2-R3=DIVIDEND
6404                                 ;       R4-R5=DIVISOR
6405                                 ;       R0-R1=REMAINDER AFTER DIVISION
6406                                 ;       R2-R3=QUOTIENT AFTER DIVISION
6407                                 ;       ENTER WITH JSR  PC,M.DPID
6408                                 ;
6409  027122  012746  000040        M.DPID: MOV     #40,-(SP)       ;COUNTER FOR DIVISION CYCLES
6410  027126  010446                        MOV     R4,-(SP)        ;HIGH ORDER
6411  027130  010546                        MOV     R5,-(SP)        ;LOW ORDER DIVISOR TO THE STACK
6412  027132  005466  000002                NEG     2(SP)           ;FORM NEGATIVE
6413  027136  005416                        NEG     @SP             ;VERSION OF THE DIVISOR
```

```
6414  027140  005666  000002          SBC     2(SP)
6415  027144  061601                  ADD     @SP,R1
6416  027146  005500                  ADC     R0                      ;PERFORM THE INITIAL SUBTRACTION
6417  027150  066600  000002          ADD     2(SP),R0
6418  027154  103445                  BCS     M.DP50                  ;IF CARRY THEN OVERFLOW HAS OCCURRED
6419  027156  005046                  CLR     -(SP)                   ;THIS IS A LONGER LASTING CARRY BIT
6420  027160  006103          M.DP40: ROL     R3
6421  027162  006102                  ROL     R2
6422  027164  006101                  ROL     R1
6423  027166  006100                  ROL     R0
6424  027170  005716                  TST     @SP                     ;TEST "CARRY" INDICATOR
6425  027172  001410                  BEQ     M.DP41                  ;IF NO "CARRY" THEN ADD ELSE SUBTRACT
6426  027174  005016                  CLR     @SP                     ;CLEAR UP FOR NEXT TIME
6427  027176  066601  000002          ADD     2(SP),R1
6428  027202  005500                  ADC     R0                      ;ADD -(DIVISOR)
6429  027204  005516                  ADC     @SP             ;  I    ;SET "CARRY"
6430  027206  066600  000004          ADD     4(SP),R0        ;<-
6431  027212  000404                  BR      M.DP42
6432  027214  060501          M.DP41: ADD     R5,R1
6433  027216  005500                  ADC     R0                      ;ADD +(DIVISOR)
6434  027220  005516                  ADC     @SP             ;  I    ;SET "CARRY"
6435  027222  060400                  ADD     R4,R0           ;<-
6436  027224  005516          M.DP42: ADC     @SP                     ;SET "CARRY"
6437  027226  005716                  TST     @SP                     ;TEST THE UPDATE INDICATOR
6438  027230  001401                  BEQ     .+4             ;->     ;IF ZERO FORGET IT
6439  027232  005203                  INC     R3              ;  I    ;NO CARRY POSSIBLE HERE
6440  027234  005366  000006          DEC     6(SP)           ;<-     ;DECREMENT COUNTER
6441  027240  003347                  BGT     M.DP40                  ;BRANCH IF MORE TO DO
6442  027242  006003                  ROR     R3
6443  027244  103404                  BCS     M.DP44
6444  027246  060501                  ADD     R5,R1
6445  027250  005500                  ADC     R0
6446  027252  060400                  ADD     R4,R0
6447  027254  000241                  CLC
6448  027256  006103          M.DP44: ROL     R3
6449  027260  062706  000010          ADD     #10,SP                  ;ADJUST STACK BY 4 WORDS
6450  027264  000242                  CLV
6451  027266  000207                  RTS     PC
6452  027270  062706  000006  M.DP50: ADD     #6,SP
6453  027274  000262                  SEV
6454  027276  000207                  RTS     PC
6455
6456
6457                                  ;ROUTINE TO REPLACE LEADING ZEROS IN A NUMERIC STRING WITH SPACES
6458                                  ;CALL
6459                                  ;       MOV     #ADR,-(SP)      ;ADDRESS OF NUMBER (IN ASCII)
6460                                  ;       JSR     R5,REPLZ
6461                                  ;       .WORD   N               ;'N' IS NUMBER OF DIGITS TO BE TYPED
6462
6463  027300  010046          REPLZ:  MOV     R0,-(SP)                ;SAVE R0
6464  027302  012746  000012          MOV     #10.,-(SP)              ;MAXIMUM NUMBER OF DIGITS TO BE TYPED
6465  027306  162516                  SUB     (R5)+,(SP)              ;SUBTACT DIGITS TO FORM INDEX
6466  027310  016600  000006          MOV     6(SP),R0                ;ADDRESS OF NUMBER TO R0
6467  027314  122710  000060  1$:     CMPB    #'0,(R0)                ;BYTE EQUAL TO ASCII '0' ?
6468  027320  001004                  BNE     2$                      ;BR IF NOT
6469  027322  112710  000040          MOVB    #40,(R0)                ;REPLACE THE ZERO WITH A SPACE
```

```
6470  027326  005200                          INC     R0              ;INCREMENT THE BYTE ADDRESS
6471  027330  000771                          BR      1$              ;GO BACK AND LOOK FOR MORE LEADING ZEROS
6472  027332  105710                  2$:     TSTB    (R0)            ;SEE IF ZERO BYTE TERMINATOR
6473  027334  001003                          BNE     3$              ;BR IF NOT
6474  027336  005300                          DEC     R0              ;BACKUP STRING POINTER
6475  027340  112710  000060                  MOVB    #'0,(R0)        ;PUT A ZERO BACK IN
6476  027344  016637  000006  027360  3$:     MOV     6(SP),4$        ;PUT ADDRESS IN LOCATION FOR TYPEOUT
6477  027352  062637  027360                  ADD     (SP)+,4$        ;BEGINNING OF SIGNIFICANT DIGITS
6478  027356  104401                          TYPE                    ;TYPE THE NUMBER
6479  027360  000000                  4$:     .WORD   0               ;ADDRESS OF NUMBER
6480  027362  012600                          MOV     (SP)+,R0        ;RESTORE R0
6481  027364  012616                          MOV     (SP)+,(SP)      ;MOVE RETURN ADDRESS
6482  027366  000205                          RTS     R5              ;RETURN
6483
6484                                  ;TYPE NUMERICAL ASCIZ STRING SUPRESS LEADING ZEROS
6485
6486                                  ;CALL
6487                                  ;       MOV     #NUMADR,-(SP)   ;FIRST ADDRESS OF ASCIZ STRING
6488                                  ;       JSR     PC,$SUPRS
6489
6490  027370  010046                  $SUPRS: MOV     R0,-(SP)        ;SAVE R0
6491  027372  016600  000004                  MOV     4(SP),R0        ;PICKUP THE POINTER
6492  027376  105710                  1$:     TSTB    (R0)            ;TERMINATOR ?
6493  027400  001403                          BEQ     2$              ;BR IF YES
6494  027402  122720  000060                  CMPB    #'0,(R0)+       ;IS THIS AN ASCII '0' ?
6495  027406  001773                          BEQ     1$              ;BR IF YES
6496  027410  005300                  2$:     DEC     R0              ;BACKUP BY '1'
6497  027412  010037  027420                  MOV     R0,3$           ;SAVE FOR TYPING
6498  027416  104414                          DISPLY                  ;GO PRINT
6499  027420  000000                  3$:     .WORD   0               ;ASCIZ POINTER GOES HERE
6500  027422  012600                          MOV     (SP)+,R0        ;RESTORE R0
6501  027424  012616                          MOV     (SP)+,(SP)      ;RESTORE THE STACK
6502  027426  000207                          RTS     PC              ;RETURN
6503
6504                                  ;ROUTINE TO TYPE AT PRIORITY 4
6505
6506  027430  013746  177776          TYPRI4: MOV     @#PS,-(SP)      ;SAVE THE PRESENT STATUS
6507  027434  012737  000200  177776          MOV     #200,@#PS       ;CHANGE THE PRIORITY TO 4
6508  027442  012537  027452                  MOV     (R5)+,1$        ;MESSAGE ADDRESS
6509  027446  004737  031034                  JSR     PC,$TYPE        ;TYPE THE MESSAGE
6510  027452  000000                  1$:     .WORD   0               ;MESSAGE ADDRESS GOES HERE
6511  027454  000205                          RTS     R5              ;RETURN
6512
6513                                  ;ROUTINE TO TYPE ERRORS
6514                                  ;CALL
6515                                  ;       DISPLY                  ;MUST DEFINED IN 'TRAP' TABLE
6516                                  ;       MESADR                  ;ADDRESS OF MESSAGE
6517                                  ;       RETURN
6518
6519  027456  032777  020000  151454  $DSPLY: BIT     #BIT13,@SWR     ;INHIBIT ERROR TYPEOUT ?
6520  027464  001004                          BNE     1$              ;BR IF YES
6521  027466  005037  177776                  CLR     @#PS            ;SET PRIORITY TO ZERO
6522  027472  000137  031034                  JMP     $TYPE           ;TYPE THE MESSAGE
6523  027476  062716  000002          1$:     ADD     #2,(SP)         ;INCREMENT THE RETURN
6524  027502  000002                          RTI                     ;RETURN
6525
```

```
                                ;THIS ROUTINE IS USED TO CHECK IF AN
                                ;ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.
                                ;CALL
                                ;       MOV     #ADR,R1         ;ADDRESS OF ASCII CHARACTER
                                ;       JSR     R5,CK.OCT       ;CHECK THE CHARACTER
                                ;       RETURN1                 ;CHARACTER IS NOT BETWEEN 0-7
                                ;       RETURN2                 ;CHARACTER IS IN R2 AS A
                                ;                               ;OCTAL DIGIT
027504  121127  000060          CK.OCT: CMPB    (R1),#'0        ;LESS THAN ZERO?
027510  103407                  1$:     BLO     1$              ;YES -- BRANCH
027512  121127  000067                  CMPB    (R1),#'7        ;GREATER THAN SEVEN?
027516  101004                          BHI     1$              ;YES -- BRANCH
027520  111102                          MOVB    (R1),R2         ;GET THE CHARACTER
027522  042702  177770                  BIC     #^C7,R2         ;STRIP AWAY THE ASCII
027526  005725                          TST     (R5)+           ;ADJUST FOR RETURN
027530  000205                          RTS     R5              ;RETURN

                                ;THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER
                                ;AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.
                                ;CALL
                                ;       MOV     #ADR,R1         ;ADDRESS OF ASCII CHARACTER
                                ;       JSR     R5,CK.DEC       ;CHECK THE CHARACTER
                                ;       RETURN1                 ;NOT BETWEEN 0 AND 9
                                ;       RETURN2                 ;BETWEEN 0 AND 9
                                ;                               ;R2 = DIGIT

027532  121127  000060          CK.DEC: CMPB    (R1),#'0        ;LESS THAN ZERO?
027536  103407                          BLO     1$              ;YES -- BRANCH
027540  121127  000071                  CMPB    (R1),#'9        ;GREATER THAN NINE?
027544  101004                          BHI     1$              ;YES -- BRANCH
027546  111102                          MOVB    (R1),R2         ;GET THE CHARACTER
027550  042702  000060                  BIC     #'0,R2          ;STRIP AWAY THE ASCII
027554  005725                          TST     (R5)+           ;ADJUST FOR RETURN
027556  000205                  1$:     RTS     R5              ;RETURN

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

027560  105711                  CK.CHR: TSTB    (R1)            ;"CARRIAGE RETURN"?
027562  001417                          BEQ     3$              ;YES -- BRANCH
027564  121127  000054                  CMPB    (R1),#',        ;"COMMA"?
027570  001413                          BEQ     2$              ;YES -- BRANCH
027572  121127  000056                  CMPB    (R1),#'.        ;"PERIOD"?
027576  001407                          BEQ     1$              ;YES -- BRANCH
027600  004537  027532                  JSR     R5,CK.DEC       ;"DIGIT"?
```

```
 6582  027604  000410                    BR      4$              ;NO -- BRANCH
 6583  027606  004537  027504            JSR     R5,CK.OCT       ;OCTAL ?
 6584  027612  005725                    TST     (R5)+           ;DIGIT BETWEEN 8-9
 6585  027614  005725                    TST     (R5)+           ;DIGIT BETWEEN 0-7
 6586  027616  005725            1$:     TST     (R5)+           ;PERIOD
```

```
6587  027620  005725                2$:     TST     (R5)+           ;COMMA
6588  027622  005725                3$:     TST     (R5)+           ;CARRIAGE RETURN
6589  027624  005201                        INC     R1              ;MOVE POINTER TO NEXT CHARACTER
6590  027626  011505                4$:     MOV     (R5),R5         ;UNKNOWN CHARACTER
6591  027630  000205                        RTS     R5              ;RETURN
6592
6593                                ;THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL
6594                                ;CHARACTERS AND FORMS A DECIMAL VALUE BINARY NUMBER IN R2.
6595                                ;CALL
6596                                ;       MOV     #ADR,R1         ;ADDRESS OF ASCIZ STRING
6597                                ;       MOV     #NUM,R2         ;MAX. MAGNITUDE OF INPUT NUMBER
6598                                ;       JSR     R5,CK.DIG       ;CHECK DIGITS
6599                                ;       RETURN  ADR1            ;"CR" ONLY ENTERED -- R2=0
6600                                ;       RETURN  ADR2            ;"PERIOD" ONLY ENTERED -- R2=0
6601                                ;       RETURN  ADR3            ;ILLEGAL CHARACTER OR INPUT TOO LARGE -- R2=?
6602                                ;       RETURN  ADR4            ;"CR" -- R2 = NUMBER
6603                                ;       RETURN  ADR5            ;"COMMA" -- R2 = NUMBER
6604                                ;       RETURN  ADR6            ;"PERIOD" -- R2 = NUMBER
6605
6606  027632  010446                CK.DIG: MOV     R4,-(SP)        ;SAVE R4
6607  027634  010346                        MOV     R3,-(SP)        ;SAVE R3
6608  027636  010246                        MOV     R2,-(SP)        ;SAVE THE MAX. SIZE ON THE STACK
6609  027640  005002                        CLR     R2              ;START WITH 0
6610  027642  005003                        CLR     R3
6611  027644  005004                        CLR     R4
6612  027646  004537  027560              JSR     R5,CK.CHR       ;CHECK ONE CHARACTER
6613  027652  027746                      6$                      ;ILLEGAL CHARACTER
6614  027654  027754                      9$                      ;CARRIAGE RETURN
6615  027656  027746                      6$                      ;","
6616  027660  027750                      7$                      ;"'"
6617  027662  027666                      1$                      ;DIGIT 0-7
6618  027664  027666                      1$                      ;DIGIT 8-9
6619  027666  062705  000004        1$:     ADD     #4,R5           ;STEP RETURN POINTER PAST "CR" & "PERIOD" RETURNS
6620  027672  006303                2$:     ASL     R3              ;INPUT NUMBER *2
6621  027674  010346                        MOV     R3,-(SP)        ;SAVE *2
6622  027676  006303                        ASL     R3              ;*4
6623  027700  006303                        ASL     R3              ;*8
6624  027702  062603                        ADD     (SP)+,R3        ;(*2)+(*8) = *10
6625  027704  060203                        ADD     R2,R3           ;UPDATE THE INPUT NUMBER
6626  027706  004537  027560              JSR     R5,CK.CHR       ;CHECK ONE CHARACTER
6627  027712  027752                      8$                      ;ILLEGAL CHARACTER
6628  027714  027736                      5$                      ;CARRIAGE RETURN
6629  027716  027734                      4$                      ;","
6630  027720  027726                      3$                      ;"'"
6631  027722  027672                      2$                      ;DIGIT 0-7
6632  027724  027672                      2$                      ;DIGIT 8-9
6633  027726  105711                3$:     TSTB    (R1)            ;DOES A "CR" FOLLOW THE "PERIOD"
6634  027730  001010                        BNE     8$              ;BR IF NOT
6635  027732  005724                        TST     (R4)+           ;INCREMENT THE RETURN
6636  027734  005724                4$:     TST     (R4)+           ;INCREMENT THE RETURN
6637  027736  005724                5$:     TST     (R4)+           ;INCREMENT THE RETURN
6638  027740  020316                        CMP     R3,(SP)         ;CHECK THE MAGNITUDE OF THE NUMBER
6639  027742  101004                        BHI     9$              ;BR IF ENTERED NUMBER TOO LARGE
6640  027744  000402                        BR      8$              ;BYPASS INCREMENT
6641  027746  005725                6$:     TST     (R5)+           ;INCREMENT RETURN PAST INVALID RETURN
6642  027750  005725                7$:     TST     (R5)+           ;INCREMENT RETURN
```

```
                                8$:     ADD     R4,R5           ;SETUP RETURN POINTER
                                9$:     MOV     R3,R2           ;ENTERED VALUE
                                        TST     (SP)+           ;CLEAN MAX. SIZE OFF OF STACK
                                        MOV     (SP)+,R3        ;RESTORE R3
                                        MOV     (SP)+,R4        ;RESTORE R4
                                        MOV     (R5),R5         ;GET RETURN ADDRESS
                                        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

027770  016637  000002  030014  $SB2D:  MOV     2(SP),1$        ;SAVE THE BINARY NUMBER
027776  012746  030014          MOV     #1$,-(SP)       ;SET THE POINTER
030002  004737  032562          JSR     PC,$DB2D        ;CALL THE DOUBLE LENGTH CONVERT
030006  012666  000002          MOV     (SP)+,2(SP)     ;PICKUP THE POINTER
030012  000207                  RTS     PC              ;RETURN
030014  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

030020  016637  000002  030044  $SB2O:  MOV     2(SP),1$        ;SAVE THE BINARY NUMBER
030026  012746  030044          MOV     #1$,-(SP)       ;SET THE POINTER
030032  004737  032756          JSR     PC,$DB2O        ;CALL THE DOUBLE LENGTH CONVERT
030036  012666  000002          MOV     (SP)+,2(SP)     ;PICKUP THE POINTER
030042  000207                  RTS     PC              ;RETURN
030044  000000  000000  1$:     .WORD   0,0

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

030050  012737  030100  000060  $TKINT: MOV     #$TKSRV,TKVEC   ;SETUP VECTOR
030056  012737  000240  000062          MOV     #PR5,TKVEC+2    ;PRIORITY TO 5
030064  005777  151056                  TST     @$TKB           ;CLEAR THE BUFFER
030070  012777  000100  151046          MOV     #BIT06,@$TKS    ;SET INTERRUPT ENABLE
030076  000207                  RTS     PC              ;RETURN

                                ;KEYBOARD INTERRUPT SERVICE ROUTINE
                                ;CALL
                                ;       ENTER VIA INTERRUPT
```

```
                                                ;READ THE KEYBOARD
030100  104410                  $TKSRV: RDCHR
030102  112637  030230                  MOVB    (SP)+,5$        ;GET THE CHARACTER
030106  023727  030230  000003          CMP     5$,#3           ;'CONTROL C' ?
030114  001012                          BNE     1$              ;BR IF NOT
030116  104401  001165                  TYPE    ,$CRLF          ;CR-LF
030122  104401  030522                  TYPE    ,$CNTLC         ;'↑C'
030126  012737  177777  001262          MOV     #-1,CFLAG       ;SET THE 'CONTROL C' FLAG
030134  005077  151004                  CLR     @$TKS           ;CLEAR THE TTY INTERRUPT
030140  000432                          BR      4$              ;EXIT
030142  023727  001140  000176  1$:      CMP     $WR,#SWREG      ;SOFTWARE SWITCH REGISTER IN USE ?
030150  001024                          BNE     3$              ;BR IF NOT
030152  023727  030230  000007          CMP     5$,#7           ;'CONTROL G' ?
030160  001020                          BNE     3$              ;BR IF NOT
030162  104401  001165                  TYPE    ,$CRLF          ;CR-LF
030166  104401  032335                  TYPE    ,$CNTLG         ;'↑G'
030172  013746  177776                  MOV     PS,-(SP)        ;PUT THE STATUS WORD ON THE STACK
030176  012746  030212                  MOV     #2$,-(SP)       ;RETURN ADDRESS
030202  005077  150736                  CLR     @$TKS           ;CLEAR THE TTY INTERRUPT ENABLE
030206  000137  031776                  JMP     $GTSWR          ;GET THE SWITCH REGISTER ENTRY
030212  012777  000100  150724  2$:      MOV     #100,@$TKS      ;ENABLE TTY KEYBOARD INTERRUPT
030220  000402                          BR      4$              ;EXIT
030222  104401  030230          3$:     TYPE    ,5$             ;ECHO THE CHARACTER
030226  000002                  4$:     RTI                     ;RETURN

030230  000000                  5$:     .WORD   0               ;ENTERED CHARACTER

                                        ;THIS ROUTINE WILL INPUT A STRING FROM THE TTY
                                        ;CALL:
                                        ;       RDLIN                   ;INPUT A STRING FROM THE TTY
                                        ;       RETURN HERE             ;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
                                        ;                               ;TERMINATOR WILL BE A BYTE OF ALL 0'S

030232  010346                  $RDLIN: MOV     R3,-(SP)        ;SAVE R3
030234  005046                          CLR     -(SP)           ;CLEAR THE RUBOUT KEY
030236  012703  030510          1$:     MOV     #$TTYIN,R3      ;GET ADDRESS
030242  022703  030522          2$:     CMP     #$TTYIN+10.,R3  ;BUFFER FULL?
030246  101467                          BLOS    4$              ;BR IF YES
030250  104410                          RDCHR                   ;GO READ ONE CHARACTER FROM THE TTY
030252  112613                          MOVB    (SP)+,(R3)      ;GET CHARACTER
030254  122713  000177                  CMPB    #177,(R3)       ;IS IT A RUBOUT
030260  001022                          BNE     5$              ;BR IF NO
030262  005716                          TST     (SP)            ;IS THIS THE FIRST RUBOUT?
030264  001007                          BNE     6$              ;BR IF NO
030266  112737  000134  030506          MOVB    #'\,9$          ;TYPE A BACK SLASH
030274  104401  030506                  TYPE    ,9$
030300  012716  177777                  MOV     #-1,(SP)        ;SET THE RUBOUT KEY
030304  005303                  6$:     DEC     R3              ;BACKUP BY ONE
030306  020327  030510                  CMP     R3,#$TTYIN      ;STACK EMPTY?
030312  103445                          BLO     4$              ;BR IF YES
030314  111337  030506                  MOVB    (R3),9$         ;SETUP TO TYPEOUT THE DELETED CHAR.
030320  104401  030506                  TYPE    ,9$             ;GO TYPE
030324  000746                          BR      2$              ;GO READ ANOTHER CHAR.
030326  005716                  5$:     TST     (SP)            ;RUBOUT KEY SET?
030330  001406                          BEQ     7$              ;BR IF NO
030332  112737  000134  030506          MOVB    #'\,9$          ;TYPE A BACK SLASH
```

```
030340  104401  030506                  TYPE    ,9$
030344  005016                          CLR     (SP)                    ;CLEAR THE RUBOUT KEY
030346  122713  000025          7$:     CMPB    #25,(R3)                ;IS CHARACTER A CTRL U?
030352  001003                          BNE     10$                     ;BR IF NO
030354  104401  032330                  TYPE    ,$CNTLU                 ;TYPE A CONTROL "U"
030360  000726                          BR      1$                      ;GO START OVER
030362  122713  000003          10$:    CMPB    #3,(R3)                 ;IS CHARACTER A CTRL C ?
030366  001006                          BNE     8$                      ;BR IF NOT
030370  012737  177777  001262          MOV     #-1,CFLAG               ;SET CNTRL C FLAG
030376  104401  030522                  TYPE    ,$CNTLC                 ;ECHO IT
030402  000427                          BR      11$                     ;EXIT
030404  122713  000012          8$:     CMPB    #12,(R3)                ;IS CHARACTER A "LF"?
030410  001011                          BNE     3$                      ;BRANCH IF NO
030412  105013                          CLRB    (R3)                    ;CLEAR THE CHARACTER
030414  104401  001165                  TYPE    ,$CRLF                  ;TYPE A "CR" & "LF"
030420  104401  030510                  TYPE    ,$TTYIN                 ;TYPE THE INPUT STRING
030424  000706                          BR      2$                      ;GO PICKUP ANOTHER CHACTER
030426  104401  001164          4$:     TYPE    ,$QUES                  ;TYPE A '?'
030432  000701                          BR      1$                      ;CLEAR THE BUFFER AND LOOP
030434  111337  030506          3$:     MOVB    (R3),9$                 ;ECHO THE CHARACTER
030440  104401  030506                  TYPE    ,9$
030444  122723  000015                  CMPB    #15,(R3)+               ;CHECK FOR RETURN
030450  001274                          BNE     2$                      ;LOOP IF NOT RETURN
030452  105063  177777                  CLRB    -1(R3)                  ;CLEAR RETURN (THE 15)
030456  104401  001166                  TYPE    ,$LF                    ;TYPE A LINE FEED
030462  005726                  11$:    TST     (SP)+                   ;CLEAN RUBOUT KEY FROM THE STACK
030464  012603                          MOV     (SP)+,R3                ;RESTORE R3
030466  011646                          MOV     (SP),-(SP)              ;ADJUST THE STACK AND PUT ADDRESS OF THE
030470  016666  000004  000002          MOV     4(SP),2(SP)             ;       FIRST ASCII CHARACTER ON IT
030476  012766  030510  000004          MOV     #$TTYIN,4(SP)
030504  000002                          RTI                             ;RETURN
030506     000                  9$:     .BYTE   0                       ;STORAGE FOR ASCII CHAR. TO TYPE
030507     000                          .BYTE   0                       ;TERMINATOR
030510  000012                  $TTYIN: .BLKB   10.                     ;RESERVE 10 BYTES FOR TTY INPUT
030522  041536  005015     000  $CNTLC: .ASCIZ  /↑C/<CR><LF>            ;CONTROL "C"

        030530                  .EVEN

                                ;;*************************************************************

                                .SBTTL  MACRO ROUTINES

                                .SBTTL  ERROR HANDLER ROUTINE

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

030530                          $ERROR:
```

```
6811  030530  104407                  CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
6812  030532  010337  001244          MOV     R3,ATTN         ;SAVE THE ATTENTION REGISTER CONTENTS
6813  030536  010137  001242          MOV     R1,DRIVE        ;DRIVE NUMBER
6814  030542  032777  020000  150370  BIT     #SW13,@SWR      ;INHIBIT PRINTOUTS ?
6815  030550  001002                  BNE     .+6             ;BR IF YES
6816  030552  004737  023450          JSR     PC,$TIME        ;TYPE THE TIME
6817  030556  105237  001103  7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
6818  030562  001775                  BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
6819  030564  013777  001102  150350  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
6820  030572  032777  002000  150340  BIT     #BIT10,@SWR     ;;BELL ON ERROR?
6821  030600  001402                  BEQ     1$              ;;NO - SKIP
6822  030602  104401  001160          TYPE    ,$BELL          ;;RING BELL
6823  030606  005237  001112  1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
6824  030612  011637  001116          MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
6825  030616  162737  000002  001116  SUB     #2,$ERRPC
6826  030624  117737  150266  001114  MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
6827  030632  032777  020000  150300  BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
6828  030640  001004                  BNE     20$             ;;SKIP TYPEOUTS
6829  030642  004737  030700          JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
6830  030646  104401  001165          TYPE    ,$CRLF
6831  030652                  20$:
6832  030652  005777  150262  2$:     TST     @SWR            ;;HALT ON ERROR
6833  030656  100002                  BPL     3$              ;;SKIP IF CONTINUE
6834  030660  000000                  HALT                    ;;HALT ON ERROR!
6835  030662  104407                  CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
6836  030664                  3$:
6837  030664  023737  000042  000046  CMP     @#42,@#46       ;ARE WE IN ACT-11 AUTO MODE?
6838  030672  001001                  BNE     .+4             ;BRANCH IF NOT
6839  030674  000000                  HALT                    ;HALT ON ERROR IF ACT AUTO MODE
6840  030676  000002                  RTI                     ;RETURN
6841
6842                          .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
6843
6844                          ;;**************************************************************
6845                          ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
6846                          ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
6847                          ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
6848
6849  030700                  $ERRTYP:
6850  030700  104401  001165          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
6851  030704  010046                  MOV     R0,-(SP)        ;;SAVE R0
6852  030706  005000                  CLR     R0              ;;PICKUP THE ITEM INDEX
6853  030710  153700  001114          BISB    @#$ITEMB,R0
6854  030714  001004                  BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
6855                                                          ;;TYPE THE PC OF THE ERROR
6856  030716  013746  001116          MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
6857                                                          ;;ERROR ADDRESS
6858  030722  104402                  TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
6859  030724  000426                  BR      6$              ;;GET OUT
6860  030726  005300          1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
6861  030730  006300                  ASL     R0              ;;      WORK FOR THE ERROR TABLE
6862  030732  006300                  ASL     R0
6863  030734  006300                  ASL     R0
6864  030736  062700  004026          ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
6865  030742  012037  030752          MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
6866  030746  001404                  BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
```

```
                        TYPE                    ;;TYPE THE "ERROR MESSAGE"
                        .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
                        TYPE    $CRLF           ;;"CARRIAGE RETURN" & "LINE FEED"
                        MOV     @(R0)+,4$       ;;PICKUP "DATA HEADER" POINTER
                        BEQ     5$              ;;SKIP TYPEOUT IF 0
                        TYPE                    ;;TYPE THE "DATA HEADER"
                        .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
                        TYPE    $CRLF           ;;"CARRIAGE RETURN" & "LINE FEED"
                        MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
                        BNE     7$              ;;GO TYPE THE DATA
                        MOV     (SP)+,R0        ;;RESTORE R0
                        TYPE    $CRLF           ;;"CARRIAGE RETURN" & "LINE FEED"
                        RTS     PC              ;;RETURN
                        MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
                        TST     (R0)            ;;IS THERE ANOTHER NUMBER?
                        BEQ     6$              ;;BR IF NO
                        TYPE    ,8$             ;;TYPE TWO(2) SPACES
                        BR      7$              ;;LOOP
                        .ASCIZ  /  /            ;;TWO(2) SPACES
                        .EVEN

                .SBTTL  TYPE ROUTINE

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

$TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
                BPL     1$      ;;BR IF YES
                HALT            ;;HALT HERE IF NO TERMINAL
                BR      3$      ;;LEAVE
1$:             MOV     R0,-(SP)        ;;SAVE R0
                MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
2$:             MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
                BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
                TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
60$:            MOV     (SP)+,R0        ;;RESTORE R0
3$:             ADD     #2,(SP)         ;;ADJUST RETURN PC
                RTI                     ;;RETURN
4$:             CMPB    #HT,(SP)        ;;BRANCH IF <HT>
                BEQ     8$
                CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
                BNE     5$
```

```
                        TST     (SP)+           ;;POP  <CR><LF> EQUIV
                        TYPE                    ;;TYPE A CR AND LF
                        $CRLF
                        CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
                        BR      2$              ;;GET NEXT CHARACTER
                5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
                6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
                        BNE     2$              ;;IF NO GO GET NEXT CHAR.
                        MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                ;;AND THE NULL CHAR.
                7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
                        BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
                        JSR     PC,$TYPEC       ;;GO TYPE A NULL
                        DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
                        BR      7$              ;;LOOP

                ;HORIZONTAL TAB PROCESSOR

                8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
                9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
                        BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
                        BNE     9$              ;;TAB STOP
                        TST     (SP)+           ;;POP SPACE OFF STACK
                        BR      2$              ;;GET NEXT CHARACTER
$TYPEC:         TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
                        BPL     $TYPEC
                        MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
                        CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
                        BNE     1$              ;;BRANCH IF NO
                        CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
                        BR      $TYPEX          ;;EXIT
                1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
                        BEQ     $TYPEX          ;;BRANCH IF YES
                        INCB    (PC)+           ;;COUNT THE CHARACTER
$CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
$TYPEX: RTS     PC


                .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE

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

```
 6979                                   ;*       TYPON                    ;;CALL FOR TYPEOUT
 6980                                   ;*
 6981                                   ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
 6982                                   ;*CALL:
 6983                                   ;*       MOV     NUM,-(SP)        ;;NUMBER TO BE TYPED
 6984                                   ;*       TYPOC                    ;;CALL FOR TYPEOUT
 6985
 6986  031254  017646  000000           $TYPOS: MOV     @(SP),-(SP)      ;;PICKUP THE MODE
 6987  031260  116637  000001  031477           MOVB    1(SP),$OFILL     ;;LOAD ZERO FILL SWITCH
 6988  031266  112637  031501                   MOVB    (SP)+,$OMODE+1   ;;NUMBER OF DIGITS TO TYPE
 6989  031272  062716  000002                   ADD     #2,(SP)          ;;ADJUST RETURN ADDRESS
 6990  031276  000406                           BR      $TYPON
 6991  031300  112737  000001  031477  $TYPOC: MOVB    #1,$OFILL        ;;SET THE ZERO FILL SWITCH
 6992  031306  112737  000006  031501          MOVB    #6,$OMODE+1      ;;SET FOR SIX(6) DIGITS
 6993  031314  112737  000005  031476  $TYPON: MOVB    #5,$OCNT         ;;SET THE ITERATION COUNT
 6994  031322  010346                           MOV     R3,-(SP)         ;;SAVE R3
 6995  031324  010446                           MOV     R4,-(SP)         ;;SAVE R4
 6996  031326  010546                           MOV     R5,-(SP)         ;;SAVE R5
 6997  031330  113704  031501                   MOVB    $OMODE+1,R4      ;;GET THE NUMBER OF DIGITS TO TYPE
 6998  031334  005404                           NEG     R4
 6999  031336  062704  000006                   ADD     #6,R4            ;;SUBTRACT IT FOR MAX. ALLOWED
 7000  031342  110437  031500                   MOVB    R4,$OMODE        ;;SAVE IT FOR USE
 7001  031346  113704  031477                   MOVB    $OFILL,R4        ;;GET THE ZERO FILL SWITCH
 7002  031352  016605  000012                   MOV     12(SP),R5        ;;PICKUP THE INPUT NUMBER
 7003  031356  005003                           CLR     R3               ;;CLEAR THE OUTPUT WORD
 7004  031360  006105                  1$:      ROL     R5               ;;ROTATE MSB INTO "C"
 7005  031362  000404                           BR      3$               ;;GO DO MSB
 7006  031364  006105                  2$:      ROL     R5               ;;FORM THIS DIGIT
 7007  031366  006105                           ROL     R5
 7008  031370  006105                           ROL     R5
 7009  031372  010503                           MOV     R5,R3
 7010  031374  006103                  3$:      ROL     R3               ;;GET LSB OF THIS DIGIT
 7011  031376  105337  031500                   DECB    $OMODE           ;;TYPE THIS DIGIT?
 7012  031402  100016                           BPL     7$               ;;BR IF NO
 7013  031404  042703  177770                   BIC     #177770,R3       ;;GET RID OF JUNK
 7014  031410  001002                           BNE     4$               ;;TEST FOR 0
 7015  031412  005704                           TST     R4               ;;SUPPRESS THIS 0?
 7016  031414  001403                           BEQ     5$               ;;BR IF YES
 7017  031416  005204                  4$:      INC     R4               ;;DON'T SUPPRESS ANYMORE 0'S
 7018  031420  052703  000060                   BIS     #'0,R3           ;;MAKE THIS DIGIT ASCII
 7019  031424  052703  000040          5$:      BIS     #' ,R3           ;;MAKE ASCII IF NOT ALREADY
 7020  031430  110337  031474                   MOVB    R3,8$            ;;SAVE FOR TYPING
 7021  031434  104401  031474                   TYPE    ,8$              ;;GO TYPE THIS DIGIT
 7022  031440  105337  031476          7$:      DECB    $OCNT            ;;COUNT BY 1
 7023  031444  003347                           BGT     2$               ;;BR IF MORE TO DO
 7024  031446  002402                           BLT     6$               ;;BR IF DONE
 7025  031450  005204                           INC     R4               ;;INSURE LAST DIGIT ISN'T A BLANK
 7026  031452  000744                           BR      2$               ;;GO DO THE LAST DIGIT
 7027  031454  012605                  6$:      MOV     (SP)+,R5         ;;RESTORE R5
 7028  031456  012604                           MOV     (SP)+,R4         ;;RESTORE R4
 7029  031460  012603                           MOV     (SP)+,R3         ;;RESTORE R3
 7030  031462  016666  000002  000004           MOV     2(SP),4(SP)      ;;SET THE STACK FOR RETURNING
 7031  031470  012616                           MOV     (SP)+,(SP)
 7032  031472  000002                           RTI                      ;;RETURN
 7033  031474     000                  8$:      .BYTE   0                ;;STORAGE FOR ASCII DIGIT
 7034  031475     000                           .BYTE   0                ;;TERMINATOR FOR TYPE ROUTINE
```

```
7035  031476  000                  $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
7036  031477  000                  $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
7037  031500  000000               $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
7038
7039                               .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
7040
7041                               ;;*****************************************************************
7042                               ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
7043                               ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
7044                               ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
7045                               ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
7046                               ;*REPLACED WITH SPACES.
7047                               ;*CALL:
7048                               ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
7049                               ;*      TYPDS                   ;;GO TO THE ROUTINE
7050
7051  031502                       $TYPDS:
7052  031502  010046                       MOV     R0,-(SP)        ;;PUSH R0 ON STACK
7053  031504  010146                       MOV     R1,-(SP)        ;;PUSH R1 ON STACK
7054  031506  010246                       MOV     R2,-(SP)        ;;PUSH R2 ON STACK
7055  031510  010346                       MOV     R3,-(SP)        ;;PUSH R3 ON STACK
7056  031512  010546                       MOV     R5,-(SP)        ;;PUSH R5 ON STACK
7057  031514  012746  020200               MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
7058  031520  016605  000020               MOV     20(SP),R5       ;;GET THE INPUT NUMBER
7059  031524  100004                       BPL     1$              ;;BR IF INPUT IS POS.
7060  031526  005405                       NEG     R5              ;;MAKE THE BINARY NUMBER POS.
7061  031530  112766  000055  000001       MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
7062  031536  005000               1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
7063  031540  012703  031716               MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
7064  031544  112723  000040               MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
7065  031550  005002               2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
7066  031552  016001  031706               MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
7067  031556  160105               3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
7068  031560  002402                       BLT     4$              ;;BR IF DONE
7069  031562  005202                       INC     R2              ;;INCREASE THE BCD DIGIT BY 1
7070  031564  000774                       BR      3$
7071  031566  060105               4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
7072  031570  005702                       TST     R2              ;;CHECK IF BCD DIGIT=0
7073  031572  001002                       BNE     5$              ;;FALL THROUGH IF 0
7074  031574  105716                       TSTB    (SP)            ;;STILL DOING LEADING 0'S?
7075  031576  100407                       BMI     7$              ;;BR IF YES
7076  031600  106316               5$:     ASLB    (SP)            ;;MSD?
7077  031602  103003                       BCC     6$              ;;BR IF NO
7078  031604  116663  000001  177777       MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
7079  031612  052702  000060         6$:   BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
7080  031616  052702  000040         7$:   BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
7081  031622  110223                       MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
7082  031624  005720                       TST     (R0)+           ;;JUST INCREMENTING
7083  031626  020027  000010               CMP     R0,#10          ;;CHECK THE TABLE INDEX
7084  031632  002746                       BLT     2$              ;;GO DO THE NEXT DIGIT
7085  031634  003002                       BGT     8$              ;;GO TO EXIT
7086  031636  010502                       MOV     R5,R2           ;;GET THE LSD
7087  031640  000764                       BR      6$              ;;GO CHANGE TO ASCII
7088  031642  105726               8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
7089  031644  100003                       BPL     9$              ;;BR IF NO
7090  031646  116663  177777  177776       MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
```

```
7091  031654  105013                  9$:     CLRB    -(R3)           ;;SET THE TERMINATOR
7092  031656  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
7093  031660  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
7094  031662  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
7095  031664  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
7096  031666  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
7097  031670  104401  031716                  TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
7098  031674  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
7099  031702  012616                          MOV     (SP)+,(SP)
7100  031704  000002                          RTI                     ;;RETURN TO USER
7101  031706  023420                  $DTBL:  10000.
7102  031710  001750                          1000.
7103  031712  000144                          100.
7104  031714  000012                          10.
7105  031716  000004                  $DBLK:  .BLKW   4
7106
7107                                  .SBTTL  TTY INPUT ROUTINE
7108
7109                                  ;;**********************************************************************
7110                                  .ENABL  LSB
7111
7112                                  ;;**********************************************************************
7113                                  ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
7114                                  ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
7115                                  ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
7116                                  ;*WHEN OPERATING IN TTY FLAG MODE.
7117  031726  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
7118  031734  001074                          BNE     15$             ;;BRANCH IF NO
7119  031736  105777  147202                  TSTB    @$TKS           ;;CHAR THERE?
7120  031742  100071                          BPL     15$             ;;IF NO, DON'T WAIT AROUND
7121  031744  117746  147176                  MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
7122  031750  042716  177600                  BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
7123  031754  022726  000007                  CMP     #7,(SP)+        ;;IS IT A CONTROL G?
7124  031760  001062                          BNE     15$             ;;NO, RETURN TO USER
7125  031762  123727  001134  000001          CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
7126  031770  001456                          BEQ     15$             ;;BRANCH IF YES
7127
7128  031772  104401  032335                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
7129  031776  104401  032342          $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
7130  032002  013746  000176                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
7131  032006  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
7132  032010  104401  032353                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
7133  032014  005046                  19$:    CLR     -(SP)           ;;CLEAR COUNTER
7134  032016  005046                          CLR     -(SP)           ;;THE NEW SWR
7135  032020  105777  147120          7$:     TSTB    @$TKS           ;;CHAR THERE?
7136  032024  100375                          BPL     7$              ;;IF NOT TRY AGAIN
7137
7138  032026  117746  147114                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
7139  032032  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
7140
7141
7142
7143  032036  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
7144  032042  001005                          BNE     10$             ;;BRANCH IF NOT
7145  032044  104401  032330                  TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
7146  032050  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
```

```
032054  000757                          BR      19$                     ;;LET'S TRY IT AGAIN


032056  021627  000015          10$:    CMP     (SP),#15                ;;IS IT A <CR>?
032062  001022                          BNE     16$                     ;;BRANCH IF NO
032064  005766  000004                  TST     4(SP)                   ;;YES, IS IT THE FIRST CHAR?
032070  001403                          BEQ     11$                     ;;BRANCH IF YES
032072  016677  000002  147040          MOV     2(SP),@SWR              ;;SAVE NEW SWR
032100  062706  000006          11$:    ADD     #6,SP                   ;;CLEAR UP STACK
032104  104401  001165          14$:    TYPE    ,$CRLF                  ;;ECHO <CR> AND <LF>
032110  123727  001135  000001          CMPB    $INTAG,#1               ;;RE-ENABLE TTY KBD INTERRUPTS?
032116  001003                          BNE     15$                     ;;BRANCH IF NOT
032120  012777  000100  147016          MOV     #100,@$TKS              ;;RE-ENABLE TTY KBD INTERRUPTS
032126  000002                  15$:    RTI                             ;;RETURN
032130  004737  031204          16$:    JSR     PC,$TYPEC               ;;ECHO CHAR
032134  021627  000060                  CMP     (SP),#60                ;;CHAR < 0?
032140  002420                          BLT     18$                     ;;BRANCH IF YES
032142  021627  000067                  CMP     (SP),#67                ;;CHAR > 7?
032146  003015                          BGT     18$                     ;;BRANCH IF YES
032150  042726  000060                  BIC     #60,(SP)+               ;;STRIP-OFF ASCII
032154  005766  000002                  TST     2(SP)                   ;;IS THIS THE FIRST CHAR
032160  001403                          BEQ     17$                     ;;BRANCH IF YES
032162  006316                          ASL     (SP)                    ;;NO, SHIFT PRESENT
032164  006316                          ASL     (SP)                    ;;    CHAR OVER TO MAKE
032166  006316                          ASL     (SP)                    ;;    ROOM FOR NEW ONE.
032170  005266  000002          17$:    INC     2(SP)                   ;;KEEP COUNT OF CHAR
032174  056616  177776                  BIS     -2(SP),(SP)             ;;SET IN NEW CHAR
032200  000707                          BR      7$                      ;;GET THE NEXT ONE
032202  104401  001164          18$:    TYPE    ,$QUES                  ;;TYPE ?<CR><LF>
032206  000720                          BR      20$                     ;;SIMULATE CONTROL-U
                                .DSABL  LSB


                                ;;*************************************************************
                                ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
                                ;*CALL:
                                ;*       RDCHR                          ;;INPUT A SINGLE CHARACTER FROM THE TTY
                                ;*       RETURN HERE                    ;;CHARACTER IS ON THE STACK
                                ;*                                      ;;WITH PARITY BIT STRIPPED OFF
                                ;

032210  011646                  $RDCHR: MOV     (SP),-(SP)              ;;PUSH DOWN THE PC
032212  016666  000004  000002          MOV     4(SP),2(SP)             ;;SAVE THE PS
032220  105777  146720          1$:     TSTB    @$TKS                   ;;WAIT FOR
032224  100375                          BPL     1$                      ;;A CHARACTER
032226  117766  146714  000004          MOVB    @$TKB,4(SP)             ;;READ THE TTY
032234  042766  177600  000004          BIC     #↑C<177>,4(SP)          ;;GET RID OF JUNK IF ANY
032242  026627  000004  000023          CMP     4(SP),#23               ;;IS IT A CONTROL-S?
032250  001013                          BNE     3$                      ;;BRANCH IF NO
032252  105777  146666          2$:     TSTB    @$TKS                   ;;WAIT FOR A CHARACTER
032256  100375                          BPL     2$                      ;;LOOP UNTIL ITS THERE
032260  117746  146662                  MOVB    @$TKB,-(SP)             ;;GET CHARACTER
032264  042716  177600                  BIC     #↑C177,(SP)             ;;MAKE IT 7-BIT ASCII
032270  022627  000021                  CMP     (SP)+,#21               ;;IS IT A CONTROL-Q?
032274  001366                          BNE     2$                      ;;IF NOT DISCARD IT
032276  000750                          BR      1$                      ;;YES, RESUME
```

```
 7203  032300  026627  000004  000140  3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
 7204  032306  002407                          BLT     4$              ;;BRANCH IF YES
 7205  032310  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
 7206  032316  003003                          BGT     4$              ;;BRANCH IF YES
 7207  032320  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
 7208  032326  000002                  4$:     RTI                     ;;GO BACK TO USER
 7209  032330  052536  005015     000 $CNTLU:  .ASCIZ   /↑U/<15><12>   ;;CONTROL "U"
 7210  032335     136  006507  000012 $CNTLG:  .ASCIZ   /↑G/<15><12>   ;;CONTROL "G"
 7211  032342  005015  053523  020122 $MSWR:   .ASCIZ   <15><12>/SWR = /
 7212  032350  020075     000
 7213  032353     040  047040  053505 $MNEW:   .ASCIZ   /  NEW = /
 7214  032360  036440  000040
 7215
 7216                                  .SBTTL  RANDOM NUMBER GENERATOR ROUTINE
 7217
 7218                                  ;;*****************************************************************
 7219                                  ;*THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
 7220                                  ;*WITH A RANGE OF 0 TO 2(+33)-1.
 7221                                  ;*CALL:
 7222                                  ;*      JSR     PC,$RAND        ;;CALL THE ROUTINE
 7223                                  ;*      RETURN                  ;;RETURN HERE THE RANDOM
 7224                                  ;*                              ;;NUMBER WILL BE IN
 7225                                  ;*                              ;;$HINUM,$LONUM
 7226
 7227  032364                  $RAND:
 7228  032364  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 7229  032366  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 7230  032370  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
 7231  032372  013700  032464                  MOV     $LONUM,R0       ;;SET R0 WITH LOW
 7232  032376  013701  032462                  MOV     $HINUM,R1       ;;SET R1 WITH HIGH
 7233  032402  012702  177771                  MOV     #-7,R2          ;;SET SHIFT COUNT
 7234  032406  006300                  1$:     ASL     R0              ;;SHIFT R0 LEFT AND
 7235  032410  006101                          ROL     R1              ;;ROTATE CARRY INTO R1 AND
 7236  032412  005202                          INC     R2              ;;CHECK FOR DONE
 7237  032414  001374                          BNE     1$              ;;CONTINUE SHIFT LOOP
 7238  032416  063700  032464                  ADD     $LONUM,R0       ;;ADD NUMBER TO MAKE X 129
 7239  032422  005501                          ADC     R1              ;;PROPOGATE CARRY
 7240  032424  063701  032462                  ADD     $HINUM,R1       ;;ADD NUMBER TO MAKE X 129
 7241  032430  062700  001057                  ADD     #1057,R0        ;;ADD LOW CONSTANT
 7242  032434  005501                          ADC     R1              ;;PROPOGATE CARRY
 7243  032436  062701  047401                  ADD     #47401,R1       ;;ADD HIGH CONSTANT
 7244  032442  010037  032464                  MOV     R0,$LONUM       ;;SAVE R0
 7245  032446  010137  032462                  MOV     R1,$HINUM       ;;SAVE R1
 7246  032452  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
 7247  032454  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
 7248  032456  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
 7249  032460  000207                          RTS     PC              ;;RETURN
 7250  032462  176543          $HINUM: .WORD   176543
 7251  032464  123456          $LONUM: .WORD   123456
 7252
 7253                                  .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES
 7254
 7255                                  ;;*****************************************************************
 7256                                  ;*SAVE R0-R5
 7257                                  ;*CALL:
 7258                                  ;*      SAVREG
```

```
                                ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
                                ;*
                                ;*TOP---(+16)
                                ;*  +2---(+18)
                                ;*  +4---R5
                                ;*  +6---R4
                                ;*  +8---R3
                                ;*+10---R2
                                ;*+12---R1
                                ;*+14---R0
                                ;
032466                          $SAVREG:
032466  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
032470  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
032472  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
032474  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
032476  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
032500  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
032502  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
032506  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
032512  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
032516  016646  000022                  MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
032522  000002                          RTI
                                ;*RESTORE R0-R5
                                ;*CALL:
                                ;*      RESREG
032524                          $RESREG:
032524  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
032530  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
032534  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
032540  012666  000022                  MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
032544  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
032546  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
032550  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
032552  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
032554  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
032556  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
032560  000002                          RTI

                                .SBTTL  DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

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

032562  104412                  $DB2D:  SAVREG                  ;;SAVE REGISTERS
032564  016602  000002                  MOV     2(SP),R2        ;;PICKUP THE DATA POINTER
032570  012700  032742                  MOV     #$DECVL,R0      ;;GET ADDRESS OF "$DECVL" STRING
```

```
 7315  032574  010066  000002                  MOV     R0,2(SP)        ;;PUT ADDRESS OF ASCIZ STRING ON STACK
 7316  032600  012201                          MOV     (R2)+,R1        ;;PICKUP THE BINARY NUMBER
 7317  032602  012202                          MOV     (R2)+,R2
 7318  032604  012737  000012  032660          MOV     #10.,4$         ;;SET UP TO DO 10 CONVERSIONS
 7319  032612  012704  032672                  MOV     #$TNPWR,R4      ;;ADDRESS OF TEN POWER
 7320  032616  012705  032674                  MOV     #$TNPWR+2,R5
 7321  032622  005003                  1$:     CLR     R3              ;;CLEAR PARTIAL
 7322  032624  161401                  2$:     SUB     (R4),R1         ;;SUBTRACT TEN POWER
 7323  032626  005602                          SBC     R2
 7324  032630  161502                          SUB     (R5),R2
 7325  032632  002402                          BLT     3$              ;;BR IF TEN POWER TO LARGE
 7326  032634  005203                          INC     R3              ;;ADD 1 TO PARTIAL
 7327  032636  000772                          BR      2$              ;;LOOP
 7328  032640  062401                  3$:     ADD     (R4)+,R1        ;;RESTORE SUBTRACTED VALUE
 7329  032642  005502                          ADC     R2
 7330  032644  062402                          ADD     (R4)+,R2
 7331  032646  022525                          CMP     (R5)+,(R5)+     ;;MOVE TO NEXT TEN POWER
 7332  032650  052703  000060                  BIS     #'0,R3          ;;CHANGE PARTIAL TO ASCII
 7333  032654  110320                          MOVB    R3,(R0)+        ;;SAVE IT
 7334  032656  005327                          DEC     (PC)+           ;;DONE?
 7335  032660  000000                  4$:     .WORD   0
 7336  032662  001357                          BNE     1$              ;;BR IF NO
 7337  032664  105020                          CLRB    (R0)+           ;;TERMINATOR
 7338  032666  104413                          RESREG                  ;;RESTORE REGISTERS
 7339  032670  000207                          RTS     PC              ;;RETURN
 7340  032672  145000                  $TNPWR: 145000                  ;;1.0E09
 7341  032674  035632                          35632
 7342  032676  160400                          160400                  ;;1.0E08
 7343  032700  002765                          2765
 7344  032702  113200                          113200                  ;;1.0E07
 7345  032704  000230                          230
 7346  032706  041100                          041100                  ;;1.0E06
 7347  032710  000017                          17
 7348  032712  103240                          103240                  ;;1.0E05
 7349  032714  000001                          1
 7350  032716  023420                          23420                   ;;1.0E04
 7351  032720  000000                          0
 7352  032722  001750                          1750                    ;;1.0E03
 7353  032724  000000                          0
 7354  032726  000144                          144                     ;;1.0E02
 7355  032730  000000                          0
 7356  032732  000012                          12                      ;;1.0E01
 7357  032734  000000                          0
 7358  032736  000001                          1                       ;;1.0E00
 7359  032740  000000                          0
 7360  032742  000014  $DECVL: .BLKB   12.                             ;;RESERVE STORAGE FOR ASCIZ STRING
 7361
 7362                  .SBTTL  DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
 7363
 7364                  ;;******************************************************************
 7365                  ;*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
 7366                  ;*UNSIGNED OCTAL ASCIZ NUMBER.
 7367                  ;*CALL
 7368                  ;*      MOV     #PNTR,-(SP)     ;;POINTER TO LOW WORD OF BINARY NUMBER
 7369                  ;*      JSR     PC,@#$DB2O      ;;CALL THE ROUTINE
 7370                  ;*      RETURN                  ;;THE ADDRESS OF THE FIRST ASCIZ CHAR. IS ON THE STACK
```

```
7371
7372
7373  032756  104412                $DB20:  SAVREG                  ;;SAVE ALL REGISTERS
7374  032760  016601  000002                MOV     2(SP),R1        ;;PICKUP THE POINTER TO LOW WORD
7375  032764  012705  033075                MOV     #$OCTVL+13.,R5  ;;POINTER TO DATA TABLE
7376  032770  012704  000014                MOV     #12.,R4         ;;DO ELEVEN CHARACTERS
7377  032774  012703  177770                MOV     #^C7,R3         ;;MASK
7378  033000  012100                       MOV     (R1)+,R0        ;;LOWER WORD
7379  033002  012101                       MOV     (R1)+,R1        ;;HIGH WORD
7380  033004  005002                       CLR     R2              ;;TERMINATOR
7381  033006  110245                1$:    MOVB    R2,-(R5)        ;;PUT CHARACTER IN DATA TABLE
7382  033010  010002                       MOV     R0,R2           ;;GET THIS DIGIT
7383  033012  005304                       DEC     R4              ;;COUNT THIS CHARACTER
7384  033014  003007                       BGT     3$              ;;BR IF NOT THE LAST DIGIT
7385  033016  001405                       BEQ     2$              ;;BR IF IT IS THE LAST DIGIT
7386  033020  005205                       INC     R5              ;;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
7387  033022  010566  000002              MOV     R5,2(SP)        ;;ASCIZ CHAR. & PUT IT ON THE STACK
7388  033026  104413                       RESREG                  ;;RESTORE ALL REGISTERS
7389  033030  000207                       RTS     PC              ;;RETURN TO USER
7390  033032  006203                2$:    ASR     R3              ;;POSITION THE MASK FOR THE LAST DIGIT
7391  033034  006001                3$:    ROR     R1              ;;POSITION THE BINARY NUMBER FOR
7392  033036  006000                       ROR     R0              ;;       THE NEXT OCTAL DIGIT
7393  033040  006001                       ROR     R1
7394  033042  006000                       ROR     R0
7395  033044  006001                       ROR     R1
7396  033046  006000                       ROR     R0
7397  033050  040302                       BIC     R3,R2           ;;MASK OUT ALL JUNK
7398  033052  062702  000060              ADD     #'0,R2          ;;MAKE THIS CHAR. ASCII
7399  033056  000753                       BR      1$              ;;GO PUT IT IN THE DATA TABLE
7400  033060  000016                $OCTVL: .BLKB   14.              ;;RESERVE DATA TABLE
7401
7402                                  .SBTTL  TRAP DECODER
7403
7404                                  ;;**************************************************************
7405                                  ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
7406                                  ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
7407                                  ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
7408                                  ;*GO TO THAT ROUTINE.
7409
7410  033076  010046                $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
7411  033100  016600  000002               MOV     2(SP),R0        ;;GET TRAP ADDRESS
7412  033104  005740                       TST     -(R0)           ;;BACKUP BY 2
7413  033106  111000                       MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
7414  033110  006300                       ASL     R0              ;;POSITION FOR INDEXING
7415  033112  016000  033132               MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
7416  033116  000200                       RTS     R0              ;;GO TO ROUTINE
7417
7418
7419                                  ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
7420
7421  033120  011646                $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
7422  033122  016666  000004  000002       MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
7423  033130  000002                       RTI                     ;;RESTORE THE PSW
7424
7425                                  .SBTTL  TRAP TABLE
7426
```

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

                                ;       ROUTINE
                                ;       -------
033132  033120                  $TRPAD: .WORD   $TRAP2
033134  031034                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
033136  031300                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
033140  031254                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
033142  031314                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
033144  031502                          $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)

033146  031776                          $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING

033150  031726                          $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
033152  032210                          $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
033154  030232                          $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
033156  032466                          $SAVREG ;;CALL=SAVREG   TRAP+12(104412) SAVE R0-R5 ROUTINE
033160  032524                          $RESREG ;;CALL=RESREG   TRAP+13(104413) RESTORE R0-R5 ROUTINE
033162  027456                          $DSPLY  ;;CALL=DISPLY   TRAP+14(104414) ROUTINE TO TYPE ERROR MESSAGES
        000032                  $TERM=.-$TRPAD

                                ;;*****************************************************************

                                        .

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

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

                                ;;*****************************************************************

                                ;STORAGE FOR RPDS1, RPER1, RPER2, AND RPER3 ON AN ERROR "2"
                                        ;RPERRS   = RPDS1
                                        ;RPERRS+2 = RPER1
                                        ;RPERRS+4 = RPER2
                                        ;RPERRS+6 = RPER3

033164  000000  000000  000000  RPERRS: .WORD   0,0,0,0
033172  000000

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

033174      000                 DRVACT: .BYTE   0               ;DRIVE 0
033175      000                         .BYTE   0               ;DRIVE 1
033176      000                         .BYTE   0               ;DRIVE 2
033177      000                         .BYTE   0               ;DRIVE 3
033200      000                         .BYTE   0               ;DRIVE 4
033201      000                         .BYTE   0               ;DRIVE 5
033202      000                         .BYTE   0               ;DRIVE 6
```

```
                                        .BYTE   0               ;DRIVE 7

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

                                DRVSTA: .BYTE   0               ;DRIVE 0
                                        .BYTE   0               ;DRIVE 1
                                        .BYTE   0               ;DRIVE 2
                                        .BYTE   0               ;DRIVE 3
                                        .BYTE   0               ;DRIVE 4
                                        .BYTE   0               ;DRIVE 5
                                        .BYTE   0               ;DRIVE 6
                                        .BYTE   0               ;DRIVE 7

                                ;TABLE OF DRIVE TYPES (DRVTYP=8 BYTES)
                                        ;DRVTYP=0 IF DRIVE IS NONEXISTENT (DRVSTA=0, ALSO)
                                        ;DRVTYP=1 IF DRIVE IS RP04
                                        ;DRVTYP=2 IF DRIVE IS RP05
                                        ;DRVTYP=4 IF DRIVE IS RP06
                                        ;DRVTYP=-1 IF NOT RP04/5/6

                                DRVTYP: .BYTE   0               ;DRIVE 0
                                        .BYTE   0               ;DRIVE 1
                                        .BYTE   0               ;DRIVE 2
                                        .BYTE   0               ;DRIVE 3
                                        .BYTE   0               ;DRIVE 4
                                        .BYTE   0               ;DRIVE 5
                                        .BYTE   0               ;DRIVE 6
                                        .BYTE   0               ;DRIVE 7

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

                                DPINT:  .BYTE   0               ;DRIVE 0
                                        .BYTE   0               ;DRIVE 1
                                        .BYTE   0               ;DRIVE 2
                                        .BYTE   0               ;DRIVE 3
                                        .BYTE   0               ;DRIVE 4
                                        .BYTE   0               ;DRIVE 5
                                        .BYTE   0               ;DRIVE 6
                                        .BYTE   0               ;DRIVE 7

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

                                DPRQS:  .BYTE   0               ;DRIVE 0
                                        .BYTE   0               ;DRIVE 1
                                        .BYTE   0               ;DRIVE 2
                                        .BYTE   0               ;DRIVE 3
                                        .BYTE   0               ;DRIVE 4
                                        .BYTE   0               ;DRIVE 5
                                        .BYTE   0               ;DRIVE 6
```

```
7539  033243  000                       .BYTE   0               ;DRIVE 7
7540
7541                                  ;TRANSFER WAIT FLAG (TRNSWT=1 WORD)
7542                                          ;THIS IS A ONE WORD QUEUE. IT WILL CONTAIN THE ADDRESS OF
7543                                          ;"DPB" OF THE I/O OPERATION.
7544
7545  033244  000000                  TRNSWT: .WORD   0
7546
7547                                  ;SEARCH WAIT KEYS (SRCHWT=1 WORD)
7548                                          ;THIS IS A ONE WORD QUEUE THAT WILL CONTAIN A KEY FOR EACH OF
7549                                          ;THE DRIVES THAT ARE PERFORMING A SEARCH COMMAND FOR THE I/O
7550                                          ;REQUEST THAT IS AT THE TOP OF THEIR REQUEST QUEUE.
7551                                          ;EACH DRIVE IS ASSIGNED ONE BIT, STARTING AT BIT00 FOR DRIVE 0.
7552
7553  033246  000000                  SRCHWT: .WORD   0
7554
7555                                  ;RP04/5/6 DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
7556                                          ;ACTDRV=0 IF DRIVER IS INACTIVE
7557                                          ;ACTDRV>0 IF DRIVER IS ACTIVE
7558
7559  033250  000                       ACTDRV: .BYTE   0
7560
7561                                  ;SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)
7562                                          ;ACTSTR=0 IF SOFTWARE TIMER ROUTINE IS INACTIVE
7563                                          ;ACTSTR>0 IF SOFTWARE TIMER ROUTINE IS ACTIVE
7564
7565  033251  000                       ACTSTR: .BYTE   0
7566
7567                                  ;UNLOAD FLAG (ULDFLG=8 BYTES)
7568                                          ;ULDFLG=0 IF NO UNLOAD COMMAND
7569                                          ;ULDFLG>0 IF UNLOAD COMMAND IN PROGRESS
7570                                          ;ULDFLG<0 IF UNLOAD COMMAND IN WAIT QUEUE
7571
7572  033252  000                       ULDFLG: .BYTE   0               ;DRIVE 0
7573  033253  000                               .BYTE   0               ;DRIVE 1
7574  033254  000                               .BYTE   0               ;DRIVE 2
7575  033255  000                               .BYTE   0               ;DRIVE 3
7576  033256  000                               .BYTE   0               ;DRIVE 4
7577  033257  000                               .BYTE   0               ;DRIVE 5
7578  033260  000                               .BYTE   0               ;DRIVE 6
7579  033261  000                               .BYTE   0               ;DRIVE 7
7580
7581                                  ;LOOK AHEAD COUNT (LACNT=8 BYTES)
7582                                          ;LACNT WILL INDICATE THE NUMBER OF LOOK AHEADS PERFORMED
7583
7584  033262  000                       LACNT:  .BYTE   0               ;DRIVE 0
7585  033263  000                               .BYTE   0               ;DRIVE 1
7586  033264  000                               .BYTE   0               ;DRIVE 2
7587  033265  000                               .BYTE   0               ;DRIVE 3
7588  033266  000                               .BYTE   0               ;DRIVE 4
7589  033267  000                               .BYTE   0               ;DRIVE 5
7590  033270  000                               .BYTE   0               ;DRIVE 6
7591  033271  000                               .BYTE   0               ;DRIVE 7
7592
7593                                  ;SAVE REGISTERS FLAG (SAVEFG =1 WORD)
7594                                          ;SAVEFG <0 IF SAVE THE RH11/RP04/5/6 REGISTERS WHEN THE
```

```
                                        ;OPERATION IS COMPLETED AS PER (DPB+14).
                                        ;SAVEFG=0 IF SAVE THE RH11/RP04/5/6 REGISTERS, AS PER
                                        ;(DPB+14), AFTER AN ERROR.

033272  000000                  SAVEFG: .WORD   0

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

033274  000000                  SEEKFG: .WORD   0

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

033276  177777                  TIMER:  .WORD   -1       ;DRIVE 0
033300  177777                          .WORD   -1       ;DRIVE 1
033302  177777                          .WORD   -1       ;DRIVE 2
033304  177777                          .WORD   -1       ;DRIVE 3
033306  177777                          .WORD   -1       ;DRIVE 4
033310  177777                          .WORD   -1       ;DRIVE 5
033312  177777                          .WORD   -1       ;DRIVE 6
033314  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

033316  177777                  DTUW:   .WORD   -1

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

033320      001                 ATABIT: .BYTE   1        ;DRIVE 0
033321      002                         .BYTE   2        ;DRIVE 1
033322      004                         .BYTE   4        ;DRIVE 2
033323      010                         .BYTE   10       ;DRIVE 3
033324      020                         .BYTE   20       ;DRIVE 4
033325      040                         .BYTE   40       ;DRIVE 5
033326      100                         .BYTE   100      ;DRIVE 6
033327      200                         .BYTE   200      ;DRIVE 7

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

033330  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)).

033332  176700                  RPADR:  .WORD   176700
033334  000254  000240          RPVEC:  .WORD   254,5*32.
```

```
                        ;MAXIMUM NUMBER OF LOOK AHEADS ALLOWED IS 4 (MXLACT=1 WORD)
                        
033340  000004          MXLACT: .WORD   4
                        ;MAXIMUM DELTA DELAY IS 8 SECTORS (MXDLTA=1 WORD)
                        
033342  001000          MXDLTA: .WORD   8.*64.
                        ;MINIMUM DELTA DELAY IS 2 SECTORS (MNDLTA=1 WORD)
                        
033344  000200          MNDLTA: .WORD   2*64.
                        ;MAXIMUM SEARCH FOR I/O WINDOW IS 5 SECTORS (MXWNDW=1 WORD)
                        
033346  000005          MXWNDW: .WORD   5
                        
                        ;DEFINITIONS OF THE RH11/RP04/5/6 ADDRESS INDEXES
                        
000000                  RPCS1=0                 ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00)
000002                  RPWC=2                  ;WORD COUNT REGISTER (NOT A DRIVE REG)
000004                  RPBA=4                  ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
000006                  RPDA=6                  ;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 05)
000010                  RPCS2=10                ;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
000012                  RPDS1=12                ;DRIVE STATUS REGISTER (DRIVE REG 01)
000014                  RPER1=14                ;ERROR REGISTER #1 (DRIVE REG. 02)
000016                  RPAS=16                 ;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 04)
000020                  RPLA=20                 ;LOOK AHEAD REGISTER (DRIVE REG. 07)
000022                  RPDB=22                 ;DATA BUFFER REGISTER (NOT A DRIVE REG.)
000024                  RPMR=24                 ;MAINTAINABILITY REGISTER (DRIVE REG. 03)
000026                  RPDT=26                 ;DRIVE TYPE REGISTER (DRIVE REG. 06)
000030                  RPSN=30                 ;SERIAL NUMBER REGISTER (DRIVE REG. 10)
000032                  RPOF=32                 ;OFFSET REGISTER (DRIVE REG. 11)
000034                  RPCA=34                 ;DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
000036                  RPCC=36                 ;CURRENT CYLINDER ADDRESS REGISTER (DRIVE REG. 13)
000040                  RPER2=40                ;ERROR REGISTER #2 (DRIVE REG. 14)
000042                  RPER3=42                ;ERROR REGISTER #3 (DRIVE REG. 15)
000044                  RPEC1=44                ;ECC POSITION REGISTER (DRIVE REG. 16)
000046                  RPEC2=46                ;ECC PATTERN REGISTER (DRIVE REG.  17)
                        
                        ;RH11/RP04/5/6 DRIVER INITIALIZATION CODE
                                ;THIS ROUTINE WILL DETERMINE WHICH RP04/5/6 DRIVES ARE
                                ;AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
                                ;TO THE PROPER STATE FOR EACH DRIVE.
                                ;NOTE: THIS ROUTINE CALLS DRVINT
                        ;
                        ;CALL
                        ;
                        ;       JSR     PC,RPINIT
                        ;       RETURN
                        ;
                        ;NOTE: THE 'P' OR 'L' CLOCK MUST BE STARTED
                        ;
033350  104412          RPINIT: SAVREG                  ;SAVE R0 - R5
033352  013746  177776          MOV     @#PS,-(SP)      ;SAVE THE PRESENT PROCESSOR STATUS
033356  012737  000240  177776  MOV     #<5*32.>,@#PS   ;CHANGE THE PRIORITY TO 5
033364  004737  041432          JSR     PC,CLRQUE       ;CLEAR ALL REQUEST QUEUES
033370  012701  033164          MOV     #RPERRS,R1      ;FIRST ADDRESS TO BE CLEARED
033374  012702  033274          MOV     #SEEKFG,R2      ;LAST ADDRESS TO BE CLEARED
033400  005021                  1$:     CLR     (R1)+           ;CLEAR
```

```
7707  033402  020102                        CMP     R1,R2           ;ARE WE DONE?
7708  033404  101775                        BLOS    1$              ;BRANCH IF NO
7709  033406  012702  033316                MOV     #DTUW,R2        ;LAST ADDRESS
7710  033412  012721  177777        2$:     MOV     #-1,(R1)+       ;INITIALIZE
7711  033416  020102                        CMP     R1,R2           ;DONE?
7712  033420  101774                        BLOS    2$              ;LOOP IF NO
7713  033422  005037  033204                CLR     DRVSTA          ;SET ALL DRIVES TO OFFLINE
7714  033426  005037  033206                CLR     DRVSTA+2
7715  033432  005037  033210                CLR     DRVSTA+4
7716  033436  005037  033212                CLR     DRVSTA+6
7717  033442  013703  033334                MOV     RPVEC,R3        ;SETUP THE RH11/RP04/5/6 VECTOR
7718  033446  012723  036214                MOV     #ISR,(R3)+
7719  033452  013713  033336                MOV     RPVEC+2,(R3)
7720  033456  013704  033332                MOV     RPADR,R4        ;FIRST ADDRESS OF RH11/RP04
7721  033462  012764  000040  000010        MOV     #BIT05,RPCS2(R4)  ;MASSBUS INIT
7722  033470  005001                        CLR     R1              ;START WITH DRIVE 0
7723  033472  004037  033562        3$:     JSR     R0,DRVINT       ;INIT THE DRIVE
7724  033476  000401                        BR      4$              ;'DVA' NOT SET OR PARITY ERROR
7725  033500  000402                        BR      5$              ;NORMAL RETURN
7726  033502  105061  033204        4$:     CLRB    DRVSTA(R1)      ;SET DRIVE STATUS TO OFFLINE
7727  033506  005201                5$:     INC     R1              ;GO TO NEXT DRIVE
7728  033510  042701  177770                BIC     #^C7,R1         ;MASK OUT UNUSED BITS
7729  033514  001366                        BNE     3$              ;BR IF MORE DRIVES TO GO
7730  033516  012701  000007                MOV     #7,R1           ;START WITH DRIVE 7
7731  033522  005037  177776                CLR     @#PS            ;CLEAR THE PROCESSOR STATUS
7732  033526  105761  033224        6$:     TSTB    DPINT(R1)       ;WAITING FOR DRIVE TO SWITCH PORTS ?
7733  033532  001405                        BEQ     8$              ;BR NOT WAITING
7734  033534  004737  041066                JSR     PC,SET.IE       ;SET INTERRUPT
7735  033540  105761  033224        7$:     TSTB    DPINT(R1)       ;DRIVE SWITCHED PORTS ?
7736  033544  001375                        BNE     7$              ;BR IF NOT
7737  033546  005301                8$:     DEC     R1              ;GO TO THE NEXT DRIVE
7738  033550  100366                        BPL     6$              ;CHECK NEXT DRIVE
7739  033552  012637  177776                MOV     (SP)+,@#PS      ;RESTORE THE PROCESSOR STATUS
7740  033556  104413                        RESREG                  ;RESTORE R0 - R5
7741  033560  000207                        RTS     PC              ;BYE-BYE
7742
7743                                ;DRIVE INITILIZATION ROUTINE
7744                                 ;THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
7745                                 ;AN RP04/5/6. IF IT IS, A "READ-IN PRESET" IS ISSUED AND FMT22
7746                                 ;IS SET TO A "1". THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
7747                                 ;INSURE THEY ARE ALL ON A "1". AND DEPENDING ON THEIR STATE,
7748                                 ;DRVSTA IS SET TO THE PROPER CONDITION.
7749                                ;CALL
7750                                ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
7751                                ;       MOV     RPADR,R4        ;UNIBUS ADDRESS OF RH11/RP04/5/6 (RPCS1)
7752                                ;       JSR     R0,DRVINT       ;CALLED BY A JSR
7753                                ;       RETURN1                 ;ERROR OCCURRED (PARITY)
7754                                ;       RETURN2                 ;NORMAL RETURN
7755                                ;
7756
7757  033562  010546                DRVINT: MOV     R5,-(SP)        ;SAVE R5
7758  033564  105061  033204                CLRB    DRVSTA(R1)      ;START DRIVE STATUS AS OFFLINE
7759  033570  105061  033214                CLRB    DRVTYP(R1)      ;CLEAR THE DRIVE TYPE INDICATOR
7760  033574  105061  033252                CLRB    ULDFLG(R1)      ;CLEAR THE UNLOAD FLAG
7761  033600  010164  000010                MOV     R1,RPCS2(R4)    ;SELECT A DRIVE
7762  033604  112764  000111  000000        MOVB    #111,RPCS1(R4)  ;DO A DRIVE CLEAR COMMAND (& SEIZE DRIVE)
```

```
7763  033612  032764  010000  000010          BIT     #BIT12,RPCS2(R4) ;NONEXISTENT DRIVE?
7764  033620  001403                          BEQ     1$              ;NO---BRANCH
7765  033622  004737  041066                  JSR     PC,SET.IE       ;GO SET "IE" WITHOUT A "TRE"
7766  033626  000520                          BR      6$              ;LEAVE THIS ROUTINE
7767  033630  105061  033204          1$:     CLRB    DRVSTA(R1)      ;SET DRIVE STATUS TO OFFLINE
7768  033634  032764  004000  000000          BIT     #BIT11,RPCS1(R4)  ;SEE IF DRIVE AVAILABLE
7769  033642  001514                          BEQ     7$              ;BR IF DRIVE NOT AVAILABLE
7770  033644  004037  040406                  JSR     R0,RD.RP        ;READ THE DRIVE TYPE REG.
7771  033650  000026                          RPDT
7772  033652  034114                          8$                      ;ERROR RETURN ADDRESS
7773  033654  012605                          MOV     (SP)+,R5        ;PUT DRIVE TYPE IN R5
7774  033656  112761  000001  033214          MOVB    #1,DRVTYP(R1)   ;SET RP04 INDICATOR
7775  033664  022705  020020                  CMP     #20020,R5       ;IS IT A SINGLE PORT RP04?
7776  033670  001431                          BEQ     2$              ;BRANCH IF YES
7777  033672  022705  024020                  CMP     #24020,R5       ;IS IT A DUAL PORT RP04?
7778  033676  001426                          BEQ     2$              ;BR IF YES
7779  033700  112761  000002  033214          MOVB    #2,DRVTYP(R1)   ;SET RP05 INDICATOR
7780  033706  022705  020021                  CMP     #20021,R5       ;SINGLE PORT RP05 ?
7781  033712  001420                          BEQ     2$              ;BR IF YES
7782  033714  022705  024021                  CMP     #24021,R5       ;DUAL PORT RP05 ?
7783  033720  001415                          BEQ     2$              ;BR IF YES
7784  033722  112761  000004  033214          MOVB    #4,DRVTYP(R1)   ;SET RP06 INDICATOR
7785  033730  022705  020022                  CMP     #20022,R5       ;SINGLE PORT RP06 ?
7786  033734  001407                          BEQ     2$              ;BR IF YES
7787  033736  022705  024022                  CMP     #24022,R5       ;DUAL PORT RP06 ?
7788  033742  001404                          BEQ     2$              ;BR IF YES
7789  033744  112761  177777  033214          MOVB    #-1,DRVTYP(R1)  ;SET INDICATOR TO 'OTHER'
7790  033752  000446                          BR      6$              ;EXIT
7791  033754  012746  000121          2$:     MOV     #121,-(SP)      ;DO A "READ-IN PRESET"
7792  033760  004037  040562                  JSR     R0,WRT.RP
7793  033764  000000                          RPCS1
7794  033766  034114                          8$
7795  033770  012746  010000                  MOV     #BIT12,-(SP)    ;SET FMT22=1
7796  033774  004037  040562                  JSR     R0,WRT.RP
7797  034000  000032                          RPOF
7798  034002  034114                          8$
7799  034004  004037  040406                  JSR     R0,RD.RP        ;READ RPDS1
7800  034010  000012                          RPDS1
7801  034012  034114                          8$
7802  034014  012605                          MOV     (SP)+,R5        ;AND SAVE IT IN R5
7803  034016  100015                          BPL     4$              ;BRANCH IF ATA=0
7804  034020  116164  033320  000016          MOVB    ATABIT(R1),RPAS(R4) ;CLEAR ATTENTION BIT
7805  034026  004037  040406                  JSR     R0,RD.RP        ;FIND OUT WHY ATA=1
7806  034032  000014                          RPER1
7807  034034  034114                          8$
7808  034036  006126                          ROL     (SP)+           ;IS IT UNSAFE?
7809  034040  100004                          BPL     4$              ;BR IF NOT
7810  034042  112761  177777  033204          MOVB    #-1,DRVSTA(R1)  ;SET UNSAFE INDICATOR
7811  034050  000407                          BR      6$              ;EXIT
7812  034052  005105                  4$:     COM     R5              ;CHECK MOL, DPR, DRY, AND VV
7813  034054  042705  167077                  BIC     #^C<BIT12!BIT08!BIT07!BIT06>,R5
7814  034060  001003                          BNE     6$              ;BRANCH IF MOL, DPR, DRY, OR VV IS CLEAR
7815  034062  112761  000001  033204          MOVB    #1,DRVSTA(R1)   ;SET DRIVE STATUS TO ONLINE
7816  034070  005720                  6$:     TST     (R0)+           ;STEP OVER THE ERROR RETURN
7817  034072  000410                          BR      8$              ;EXIT
7818  034074  006301                  7$:     ASL     R1              ;CHANGE INDEX TO ADDRESS WORDS
```

```
                                                MOV     #2000.,TIMER(R1)  ;START 2 SEC TIMER
                                                ASR     R1                ;RESTORE R1
                                                COMB    DPINT(R1)         ;SET PORT INITIALIZE INIDICATOR
                                                TST     (R0)+
                                        8$:     MOV     (SP)+,R5          ;RESTORE R5
                                                RTS     R0                ;EXIT

                                        ;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

                                        RP04:   MOV     @#PS,-(SP)        ;SAVE THE CALLING STATUS
                                                MOV     RPVEC+2,@#PS      ;DON'T ALLOW ANY RP04/5/6 INTERRUPTS
                                                MOVB    #1,ACTDRV         ;SET "ACTIVE DRIVER" FLAG
                                                SAVREG                    ;SAVE R0 - R5
                                                MOV     (R0),R2           ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
                                                CLR     16(R2)            ;CLEAR THE STATUS/ERROR INDICATOR
                                                MOVB    (R2),R1           ;PICKUP THE DRIVE NUMBER
                                                MOV     RPADR,R4          ;UNIBUS ADDRESS OF RPCS1
                                                TSTB    DRVSTA(R1)        ;CHECK DRIVES STATUS
                                                BGT     1$                ;BRANCH IF ONLINE
                                                TSTB    ULDFLG(R1)        ;UNLOAD COMMAND IN QUEUE?
                                                BNE     3$                ;BRANCH IF YES
                                                TSTB    DPINT(R1)         ;TRYING TO INIT THE DRIVE
                                                BNE     5$                ;BR IF YES
                                                JSR     R0,DRVINT         ;GO INIT. THE DRIVE
                                                BR      4$                ;ERROR RETURN
                                        1$:     TSTB    DRVSTA(R1)        ;IS DRIVE STATUS ONLINE?
                                                BLE     6$                ;BR IF NOT
                                                TSTB    DPRQS(R1)         ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
                                                BNE     5$                ;BR IF YES
                                                MOV     R1,RPCS2(R4)      ;SELECT THE DRIVE
                                                JSR     R0,DRVQUE         ;PUT THIS REQUEST IN QUEUE
                                                BR      9$                ;QUEUE IS FULL
                                                CMPB    #103,2(R2)        ;IS THIS REQ. FOR AN UNLOAD?
                                                BNE     2$                ;BR IF NO
                                                MOVB    #-1,ULDFLG(R1)    ;SET THE "UNLOAD IN QUEUE" FLAG
                                        2$:     TSTB    DRVACT(R1)        ;IS THIS DRIVE ACTIVE?
                                                BNE     8$                ;BR IF YES
                                                JSR     PC,OPT            ;CALL THE OPTIMIZER
                                                BR      8$
                                        3$:     MOV     #BIT15!BIT13,16(R2) ;SET THE "UNLOAD IN QUEUE" ERROR FLAG
                                                BR      8$                ;EXIT
                                        4$:     JSR     PC,CI7            ;GO HANDLE THE PARITY ERROR
                                                BR      8$
                                        5$:     JSR     R0,DRVQUE         ;PUT REQUEST IN QUEUE
                                                BR      9$                ;QUEUE IS FULL
                                                BIT     #BIT06,(R4)       ;IS 'IE' SET ALREADY ?
                                                BNE     8$                ;BR IF IT IS
                                                JSR     PC,SET.IE         ;SET INTERRUPT
```

```
7875  034324  000420                        BR      8$                ;RETURN, REQUEST IN QUEUE
7876  034326  105761  033204          6$:   TSTB    DRVSTA(R1)        ;SEE IF DRIVE OFFLINE OR UNSAFE
7877  034332  002412                        BLT     7$                ;BR IF UNSAFE
7878  034334  012762  140000  000016        MOV     #BIT15!BIT14,16(R2)  ;SET OFFLINE ERROR INDICATOR
7879  034342  105761  033214                TSTB    DRVTYP(R1)        ;SEE IF OFFLINE OR NONEXISTENT
7880  034346  001007                        BNE     8$                ;BR IF OFFLINE
7881  034350  012762  100002  000016        MOV     #BIT15!BIT01,16(R2)  ;REPORT DRIVE NONEXISTENT
7882  034356  000403                        BR      8$                ;GO TO EXIT
7883  034360  012762  110000  000016  7$:   MOV     #BIT15!BIT12,16(R2)  ;DRIVE IS UNSAFE
7884  034366  104413                  8$:   RESREG                    ;RESTORE R0 - R5
7885  034370  005720                        TST     (R0)+             ;SETUP FOR NORMAL RETURN
7886  034372  000401                        BR      10$               ;FINISH UP, THEN EXIT
7887  034374  104413                  9$:   RESREG                    ;RESTORE R0 - R5
7888  034376  005720                  10$:  TST     (R0)+             ;CORRECT THE RETURN ADDRESS
7889  034400  105037  033250                CLRB    ACTDRV            ;CLEAR "ACTIVE DRIVER" FLAG
7890  034404  012637  177776                MOV     (SP)+,@#PS        ;RETURN "PS" TO USER LEVEL
7891  034410  000200                        RTS     R0                ;RETURN TO CALLER
7892
7893                                  ;OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
7894                                  ;
7895                                  ;CALL
7896                                  ;       MOV     #DRVNUM,R1        ;DRIVE NUMBER TO R1
7897                                  ;       JSR     PC,OPT            ;SETUP A COMMAND
7898                                  ;
7899  034412  104412                  OPT:   SAVREG                    ;SAVE R0 - R5
7900  034414  013746  177776                MOV     @#PS,-(SP)        ;SAVE PROC. STATUS
7901  034420  146137  033320  033246        BICB    ATABIT(R1),SRCHWT       ;CLEAR "SEARCH WAIT" KEY
7902  034426  004737  041604                JSR     PC,GETREQ         ;GET "DPB" POINTER OF REQUEST
7903  034432  005702                        TST     R2                ;IS THERE A REQUEST IN QUEUE?
7904  034434  001505                        BEQ     7$                ;NO--BRANCH TO EXIT
7905  034436  032764  004000  000000        BIT     #BIT11,RPCS1(R4)        ;IS DVA SET?
7906  034444  001407                        BEQ     10$               ;BRANCH IF NOT
7907  034446  032764  000100  000012        BIT     #BIT6,RPDS1(R4)   ;IS VV SET ?
7908  034454  001003                        BNE     10$               ;BR IF IT IS
7909  034456  004037  033562            9$:   JSR     R0,DRVINT         ;SEE IF DRIVE STILL ONLINE ?
7910  034462  000470                        BR      6$                ;PARITY OR 'DVA' NOT SET
7911  034464  105761  033204          10$:  TSTB    DRVSTA(R1)        ;IS DRIVE ONLINE?
7912  034470  003014                        BGT     1$                ;YES--BRANCH
7913  034472  004737  041626                JSR     PC,POPQUE         ;NO--REMOVE REQUEST FROM QUEUE
7914  034476  012762  140000  000016        MOV     #BIT15!BIT14,16(R2)  ;SET OFFLINE STATUS/ERROR INDICATOR
7915  034504  105761  033204                TSTB    DRVSTA(R1)        ;IS DRIVE UNSAFE ?
7916  034510  100064                        BPL     8$                ;BR TO EXIT IF NOT
7917  034512  012762  110000  000016        MOV     #BIT15!BIT12,16(R2)  ;SET UNSAFE STATUS/ERROR INDICATOR
7918  034520  000460                        BR      8$                ;BRANCH TO EXIT
7919  034522  012746  000111          1$:   MOV     #111,-(SP)        ;LOAD COMMAND ONTO THE STACK
7920  034526  004037  040562                JSR     R0,WRT.RP         ;LOAD THE REGISTER
7921  034532  000000                        RPCS1                     ;REGISTER INCREMENT
7922  034534  034644                        6$                        ;ERROR RETURN ADDRESS
7923  034536  032714  004000                BIT     #BIT11,(R4)       ;DRIVE AVAILABLE ?
7924  034542  001427                        BEQ     5$                ;BR IF NOT
7925  034544  122762  000150  000002        CMPB    #150,2(R2)        ;IS THE REQUEST FOR I/O?
7926  034552  002403                        BLT     2$                ;YES--BRANCH
7927  034554  004737  035106                JSR     PC,CI4            ;CALL THE COMMAND INITIATOR
7928  034560  000440                        BR      8$                ;BRANCH TO EXIT
7929  034562  005737  033316          2$:   TST     DTUW              ;DATA TRANSFER UNDERWAY?
7930  034566  002012                        BGE     4$                ;YES--GO START A SEARCH
```

```
034570  005737  033274                   TST     SEEKFG                  ;DO IMPLIED SEEKS?
034574  100404                           BMI     3$                      ;YES---BRANCH
034576  004037  036056                   JSR     R0,LA                   ;NO--DO LOOK AHEAD
034602  000427                           BR      8$                      ;RETURN HERE ON A PARITY ERROR
034604  000403                           BR      4$                      ;GO START A SEARCH
034606  004737  034672          3$:      JSR     PC,CI1                  ;START A DATA TRANSFER
034612  000423                           BR      8$
034614  004737  035000          4$:      JSR     PC,CI3                  ;START A SEARCH
034620  000420                           BR      8$                      ;GO TO THE EXIT
034622  112761  177777  033234  5$:      MOVB    #-1,DPRQS(R1)           ;SET PORT REQUEST INDICATOR
034630  010103                           MOV     R1,R3                   ;SET UP TO ADDRESS WORDS
034632  006303                           ASL     R3                      ;CONVERT TO WORD INDEX
034634  012763  023420  033276           MOV     #10000.,TIMER(R3)       ;START 10 SEC TIMER
034642  000402                           BR      7$                      ;EXIT
034644  004737  035522          6$:      JSR     PC,CI7                  ;PROCESS THE PARITY ERROR
034650  032714  000100          7$:      BIT     #BIT06,(R4)             ;SEE IF 'IE' ALREADY SET
034654  001002                           BNE     8$                      ;BR IF SET
034656  004737  041066                   JSR     PC,SET.IE               ;SET "IE" WITHOUT A "TRE"
034662  012637  177776          8$:      MOV     (SP)+,@#PS              ;RESTORE PROC. STATUS
034666  104413                           RESREG                          ;RESTORE R0 - R5
034670  000207                           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

034672  004737  041626          CI1:     JSR     PC,POPQUE               ;REMOVE REQUEST FROM "DRIVES WAIT" QUEUE
034676  010237  033244                   MOV     R2,TRNSWT               ;PUT REQ. IN TRANSFER WAIT QUEUE
034702  010203                           MOV     R2,R3                   ;DPB ADDRESS TO R3
034704  013704  033332                   MOV     RPADR,R4                ;RPCS1 ADDRESS
034710  010164  000010                   MOV     R1,RPCS2(R4)            ;SELECT DRIVE
034714  062703  000004                   ADD     #4,R3                   ;DESIRED WORD COUNT
034720  062704  000002                   ADD     #2,R4                   ;RPWC ADDRESS
034724  012324                           MOV     (R3)+,(R4)+             ;LOAD WORD COUNT
034726  012324                           MOV     (R3)+,(R4)+             ;LOAD BUFFER ADDRESS
034730  012346                           MOV     (R3)+,-(SP)             ;LOAD SECTOR AND TRACK
034732  004037  040562                   JSR     R0,WRT.RP               ;CALL THE LOAD(WRITE) ROUTINE
034736  000006                           RPDA                            ;INDEX OF REGISTER TO LOAD
034740  035522                           CI7                             ;ERROR RETURN ADDRESS
034742  012346                           MOV     (R3)+,-(SP)             ;LOAD CYLINDER ADDRESS
034744  004037  040562                   JSR     R0,WRT.RP
034750  000034                           RPCA
034752  035522                           CI7
034754  016246  000002                   MOV     2(R2),-(SP)             ;LOAD "COMMAND+GO", "A17&A16", AND "PSEL"
034760  004037  040562                   JSR     R0,WRT.RP
034764  000000                           RPCS1
034766  035522                           CI7
034770  010137  033316                   MOV     R1,DTUW                 ;SET "DATA TRANSFER UNDERWAY"
034774  000137  035464                   JMP     CI5
```

```
7987  035000  013704  033332          CI3:    MOV     RPADR,R4        ;RPCS1 ADDRESS
7988  035004  010164  000010                  MOV     R1,RPCS2(R4)    ;SELECT DRIVE
7989  035010  016246  000012                  MOV     12(R2),-(SP)    ;DESIRED CYLINDER ADDRESS
7990  035014  004037  040562                  JSR     R0,WRT.RP
7991  035020  000034                          RPCA
7992  035022  035522                          CI7
7993  035024  116203  000010                  MOVB    10(R2),R3       ;PICKUP SECTOR ADDRESS
7994  035030  163703  033346                  SUB     MXWNDW,R3       ;BACKUP BY MAX. SEARCH FOR I/O WINDOW
7995  035034  002002                          BGE     1$
7996  035036  062703  000026                  ADD     #22.,R3
7997  035042  010346                  1$:     MOV     R3,-(SP)        ;COMBINE THE ADJUSTED SECTOR WITH
7998  035044  116266  000011  000001          MOVB    11(R2),1(SP)    ;THE DESIRED TRACK
7999  035052  004037  040562                  JSR     R0,WRT.RP       ;LOAD DESIRED TRACK & SECTOR
8000  035056  000006                          RPDA
8001  035060  035522                          CI7
8002  035062  012746  000131                  MOV     #131,-(SP)      ;START A SEARCH
8003  035066  004037  040562                  JSR     R0,WRT.RP
8004  035072  000000                          RPCS1
8005  035074  035522                          CI7
8006  035076  156137  033320  033246          BISB    ATABIT(R1),SRCHWT       ;SET "SEARCH WAIT" KEY
8007  035104  000567                          BR      CI5
8008  035106  013704  033332          CI4:    MOV     RPADR,R4        ;RPCS1 ADDRESS
8009  035112  010164  000010                  MOV     R1,RPCS2(R4)    ;SELECT DRIVE
8010  035116  116203  000002                  MOVB    2(R2),R3        ;PICKUP THE REQUESTED COMMAND
8011  035122  122703  000131                  CMPB    #131,R3         ;IS IT A SEARCH COMMAND?
8012  035126  001007                          BNE     1$              ;BRANCH IF NO
8013  035130  016246  000010                  MOV     10(R2),-(SP)    ;LOAD DESIRED TRACK & SECTOR
8014  035134  004037  040562                  JSR     R0,WRT.RP
8015  035140  000006                          RPDA
8016  035142  035522                          CI7
8017  035144  000403                          BR      2$              ;GO LOAD CYLINDER
8018  035146  122703  000105          1$:     CMPB    #105,R3         ;IS IT A SEEK COMMAND
8019  035152  001007                          BNE     3$              ;BRANCH IF NO
8020  035154  016246  000012          2$:     MOV     12(R2),-(SP)    ;LOAD DESIRED CYLINDER
8021  035160  004037  040562                  JSR     R0,WRT.RP
8022  035164  000034                          RPCA
8023  035166  035522                          CI7
8024  035170  000546                          BR      CI6
8025  035172  122703  000115          3$:     CMPB    #115,R3         ;IS IT AN "OFFSET" COMMAND?
8026  035176  001013                          BNE     4$              ;BR IF NO
8027  035200  004037  040406                  JSR     R0,RD.RP        ;MERGE THE OFFSET VALUE INTO RPOF
8028  035204  000032                          RPOF                    ;BUT DON'T CHANGE THE UPPER
8029  035206  035522                          CI7
8030  035210  116216  000001                  MOVB    1(R2),(SP)      ;BYTE WHEN LOADING THE
8031  035214  004037  040562                  JSR     R0,WRT.RP       ;REGISTER (RPOF)
8032  035220  000032                          RPOF
8033  035222  035522                          CI7
8034  035224  000530                          BR      CI6             ;GO START THE COMMAND
8035  035226  122703  000107          4$:     CMPB    #107,R3         ;IS IT A "RECALIBRATE" COMMAND?
8036  035232  001525                          BEQ     CI6             ;BRANCH IF YES
8037  035234  122703  000117                  CMPB    #117,R3         ;IS IT A RETURN TO CENTER?
8038  035240  001522                          BEQ     CI6             ;BRANCH IF YES
8039  035242  122703  000103                  CMPB    #103,R3         ;IS IT AN "UNLOAD" COMMAND?
8040  035246  001016                          BNE     5$              ;BRANCH IF NO
8041  035250  112761  000001  033174          MOVB    #1,DRVACT(R1)   ;SET THE DRIVE ACTIVE INDICATOR
8042  035256  105061  033204                  CLRB    DRVSTA(R1)      ;PUT DRIVE STATUS TO OFFLINE
```

```
8043  035262  112761  000001  033252          MOVB    #1,ULDFLG(R1)   ;SET "UNLOAD IN PROGRESS" FLAG
8044  035270  010346                          MOV     R3,-(SP)        ;START THE "UNLOAD" COMMAND
8045  035272  004037  040562                  JSR     R0,WRT.RP
8046  035276  000000                          RPCS1
8047  035300  035522                          CI7
8048  035302  000207                          RTS     PC              ;RETURN TO USER
8049  035304  122703  000143          5$:     CMPB    #143,R3         ;IS IT A "SET FORMAT" COMMAND?
8050  035310  001014                          BNE     6$              ;BRANCH IF NO
8051  035312  004037  040406                  JSR     R0,RD.RP        ;READ THE OFFSET REGISTER
8052  035316  000032                          RPOF
8053  035320  035522                          CI7
8054  035322  116266  000001  000001          MOVB    1(R2),1(SP)     ;COMBINE "FMT22","ECI", AND "HCI"
8055  035330  004037  040562                  JSR     R0,WRT.RP       ;LOAD "FMT22", "ECI", AND/OR "HCI".
8056  035334  000032                          RPOF
8057  035336  035522                          CI7
8058  035340  000436                          BR      12$
8059  035342  122703  000141          6$:     CMPB    #141,R3         ;IS IT A "GET REGISTER" COMMAND?
8060  035346  001023                          BNE     10$             ;BRANCH IF NO
8061  035350  016203  000006          7$:     MOV     6(R2),R3        ;POINTS TO 1ST ADDRESS OF WHERE
8062                                                                  ;TO PUT THE REGISTER(S)
8063  035354  116237  000010  035372          MOVB    10(R2),9$       ;INIT. THE INDEX FOR THE FIRST REG.
8064  035362  116205  000011                  MOVB    11(R2),R5       ;INDEX OF LAST REG. TO MOVE
8065  035366  004037  040406          8$:     JSR     R0,RD.RP        ;READ RPO4/5/6 REGISTER
8066  035372  000000          9$:     RPCS1                           ;INDEX OF REG. TO READ
8067  035374  035522                          CI7
8068  035376  012623                          MOV     (SP)+,(R3)+     ;GET THE CONTENTS OF RH11/RPO4/5/6 REG.
8069  035400  023705  035372                  CMP     9$,R5           ;LAST REG. BEEN READ?
8070  035404  001414                          BEQ     12$             ;GET OUT IF YES
8071  035406  062737  000002  035372          ADD     #2,9$           ;INCREASE THE INDEX BY 2
8072  035414  000764                          BR      8$              ;LOOP--MORE TO READ
8073  035416  122703  000145          10$:    CMPB    #145,R3         ;IS IT A "SELECT DRIVE" COMMAND?
8074  035422  001405                          BEQ     12$             ;BRANCH IF YES
8075  035424  010346                  11$:    MOV     R3,-(SP)        ;LOAD THE COMMAND
8076  035426  004037  040562                  JSR     R0,WRT.RP
8077  035432  000000                          RPCS1
8078  035434  035522                          CI7
8079  035436  004737  041626          12$:    JSR     PC,POPQUE       ;REMOVE REQ. FROM QUEUE
8080  035442  052762  000200  000016          BIS     #BIT07,16(R2)   ;SET THE "DONE" BIT
8081  035450  005737  033272                  TST     SAVEFG          ;SAVE THE RH11/RPO4/5/6 REGISTERS?
8082  035454  100002                          BPL     13$             ;BRANCH IF NO
8083  035456  004737  040750                  JSR     PC,SVRH11       ;YES--GO SAVE THE REGISTERS
8084  035462  000207                  13$:    RTS     PC              ;RETURN TO USER
8085  035464  006301                  CI5:    ASL     R1
8086  035466  012761  001750  033276          MOV     #1000.,TIMER(R1)        ;SET A ONE SECOND TIMER
8087  035474  006201                          ASR     R1
8088  035476  112761  000001  033174          MOVB    #1,DRVACT(R1)   ;SET THE DRIVE ACTIVE
8089  035504  000207                          RTS     PC              ;RETURN TO THE USER
8090  035506  010346                  CI6:    MOV     R3,-(SP)        ;LOAD THE COMMAND
8091  035510  004037  040562                  JSR     R0,WRT.RP
8092  035514  000000                          RPCS1
8093  035516  035522                          CI7
8094  035520  000761                          BR      CI5
8095  035522  032764  010000  000010  CI7:    BIT     #BIT12,RPCS2(R4)  ;DRIVE NON-EXISTENT ?
8096  035530  001034                          BNE     CI8             ;BR IF YES
8097  035532  005702                  1$:     TST     R2              ;ANYTHING IN QUEUE ?
8098  035534  001405                          BEQ     CI7B            ;BR IF NOT
```

```
8099   035536  012762  104000  000016          MOV     #BIT15!BIT11,16(R2)     ;SET "PARITY" ERROR INDICATOR
8100   035544  004737  040750                  JSR     PC,SVRH11               ;GO SAVE THE RH11/RP04/5/6 REGISTERS
8101   035550  012746  000111          CI7B:   MOV     #111,-(SP)              ;DO A "DRIVE CLEAR"
8102   035554  004037  040562                  JSR     R0,WRT.RP
8103   035560  000000                          RPCS1
8104   035562  035622                          CI8
8105   035564  004737  041510                  JSR     PC,EMPTYQ               ;EMPTY THE QUEUE
8106   035570  105061  033252                  CLRB    ULDFLG(R1)              ;CLEAR THE UNLOAD IN QUEUE FLAG
8107   035574  105061  033174                  CLRB    DRVACT(R1)              ;DRIVE IS IDLE
8108   035600  020137  033316                  CMP     R1,DTUW                 ;IF THIS DRIVE HAD AN I/O REQUEST
8109   035604  001005                          BNE     1$                      ;IN PROGRESS CLEAR ALL OF THE FLAGS
8110   035606  005037  033244                  CLR     TRNSWT
8111   035612  012737  177777  033316          MOV     #-1,DTUW
8112   035620  000207                  1$:     RTS     PC
8113   035622  104412                  CI8:    SAVREG                          ;SAVE R0 - R5
8114   035624  032764  010000  000010          BIT     #BIT12,RPCS2(R4)        ;IS 'NED' SET ?
8115   035632  001002                          BNE     1$                      ;BR IF YES
8116   035634  005001                          CLR     R1
8117   035636  005003                          CLR     R3
8118   035640  105761  033174          1$:     TSTB    DRVACT(R1)              ;DRIVE ACTIVE?
8119   035644  001443                          BEQ     5$                      ;BRANCH IF NO
8120   035646  013702  033244                  MOV     TRNSWT,R2               ;GET THE "TRANSFER WAIT" QUEUE
8121   035652  020137  033316                  CMP     R1,DTUW                 ;DID THIS DRIVE HAVE AN I/O IN PROGRESS?
8122   035656  001402                          BEQ     2$                      ;BRANCH IF YES
8123   035660  004737  041604                  JSR     PC,GETREQ               ;GET THE DPB POINTER
8124   035664  005702                  2$:     TST     R2                      ;QUEUE ENTRY FOR DRIVE ?
8125   035666  001415                          BEQ     4$                      ;BR IF NOT
8126   035670  032764  010000  000010          BIT     #BIT12,RPCS2(R4)        ;'NED' SET ?
8127   035676  001404                          BEQ     3$                      ;BR IF NOT
8128   035700  012762  100002  000016          MOV     #BIT15!BIT01,16(R2)     ;SET 'DRIVE NON-EXISTENT' INDICATOR
8129   035706  000405                          BR      4$                      ;CONTINUE
8130   035710  012762  102000  000016  3$:     MOV     #BIT15!BIT10,16(R2)     ;SET "NON-CLEARABLE PARITY" ERROR INDICATOR
8131   035716  004737  040750                  JSR     PC,SVRH11               ;SAVE RH11/RP04/5/6 REGISTERS
8132   035722  012763  177777  033276  4$:     MOV     #-1,TIMER(R3)           ;STOP THE TIMER
8133   035730  105061  033174                  CLRB    DRVACT(R1)              ;SET "DRIVE ACTIVE" TO IDLE
8134   035734  020137  033316                  CMP     R1,DTUW                 ;IS THIS DRIVE SETUP FOR A TRANSFER
8135   035740  001005                          BNE     5$                      ;BR IF NOT
8136   035742  012737  177777  033316          MOV     #-1,DTUW                ;RESET THE INDICATOR
8137   035750  005037  033244                  CLR     TRNSWT                  ;CLEAR THE TRANSFER QUEUE
8138   035754  105061  033252          5$:     CLRB    ULDFLG(R1)              ;CLEAR UNLOAD FLAG
8139   035760  032764  010000  000010          BIT     #BIT12,RPCS2(R4)        ;'NED' SET ?
8140   035766  001021                          BNE     6$                      ;BR IF YES
8141   035770  005201                          INC     R1                      ;MOVE TO THE NEXT DRIVE
8142   035772  062703  000002                  ADD     #2,R3
8143   035776  042701  177770                  BIC     #^C7,R1
8144   036002  001316                          BNE     1$                      ;BRANCH IF MORE DRIVES
8145   036004  012737  177777  033316          MOV     #-1,DTUW                ;NO DATA TRANSFERS UNDERWAY
8146   036012  005037  033244                  CLR     TRNSWT                  ;CLEAR THE 'TRANSFER WAIT' QUEUE
8147   036016  004737  041432                  JSR     PC,CLRQUE               ;CLEAR ALL OF THE REQUEST QUEUES
8148   036022  012764  000040  000010          MOV     #BIT05,RPCS2(R4)        ;DO A MASSBUS INIT.
8149   036030  000406                          BR      7$                      ;CONTINUE
8150   036032  004737  041510          6$:     JSR     PC,EMPTYQ               ;CLEAR THE DRIVE'S QUEUE
8151   036036  105061  033204                  CLRB    DRVSTA(R1)              ;SET DRIVE TO OFFLINE
8152   036042  105061  033214                  CLRB    DRVTYP(R1)              ;CLEAR THE DRIVE TYPE INDICATOR
8153   036046  004737  041066          7$:     JSR     PC,SET.IE               ;SET "IE" WITHOUT "TRE"
8154   036052  104413                          RESREG                          ;RESTORE R0 - R5
```

```
8155  036054  000207                          RTS     PC              ;RETURN
8156
8157                                  ;LOOK AHEAD ROUTINE
8158                                  ;
8159                                  ;CALL
8160                                  ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER
8161                                  ;       MOV     #DPB,R2         ;POINT TO DPB
8162                                  ;       JSR     R0,LA           ;GO CHECK THE WINDOW
8163                                  ;       RETURN1                 ;ERROR RETURN
8164                                  ;       RETURN2                 ;START A SEARCH
8165                                  ;       RETURN3                 ;START A DATA TRANSFER
8166
8167  036056  013704  033332          LA:     MOV     RPADR,R4        ;GET RPCS1'S ADDRESS
8168  036062  010164  000010          MOV     R1,RPCS2(R4)    ;SELECT DRIVE
8169  036066  004037  040406          JSR     R0,RD.RP        ;READ CURRENT CYLINDER
8170  036072  000036                  RPCC
8171  036074  036206                  4$                      ;ERROR RETURN ADDRESS
8172  036076  022662  000012          CMP     (SP)+,12(R2)    ;IS CURRENT CYLINDER=DESIRED
8173                                                          ;CYLINDER?
8174  036102  001037                  BNE     3$              ;EXIT IF NO
8175  036104  105261  033262          INCB    LACNT(R1)       ;INCREMENT THE LOOK AHEAD COUNT
8176  036110  126137  033262  033340  CMPB    LACNT(R1),MXLACT ;EXCEED MAX?
8177  036116  003026                  BGT     2$              ;BRANCH IF YES
8178  036120  116203  000010          MOVB    10(R2),R3       ;GET DESIRED SECTOR ADDRESS AND
8179  036124  000303                  SWAB    R3              ;MULT. BY 64--ALIGN WITH
8180  036126  006203                  ASR     R3              ;LOOK AHEAD REGISTER
8181  036130  006203                  ASR     R3
8182  036132  012737  000340  177776  MOV     #340,@#PS       ;PRIORITY LEVEL "7"
8183  036140  004037  040406          JSR     R0,RD.RP        ;READ LOOK AHEAD REGISTER
8184  036144  000020                  RPLA
8185  036146  036206                  4$
8186  036150  162603                  SUB     (SP)+,R3        ;CALCULATE THE DELTA
8187  036152  002002                  BGE     1$
8188  036154  062703  002600          ADD     #<22.*64.>,R3   ;MAKE THE DELTA POSITIVE
8189  036160  023703  033342          1$:     CMP     MXDLTA,R3       ;CHECK THE DELTA TO SEE
8190  036164  002406                  BLT     3$              ;IF IT IS WITHIN THE
8191  036166  023703  033344          CMP     MNDLTA,R3       ;WINDOW---IF YES, ZERO
8192  036172  002003                  BGE     3$              ;THE LOOK AHEAD COUNT
8193  036174  105061  033262          2$:     CLRB    LACNT(R1)       ;AND TAKE THE I/O EXIT
8194  036200  005720                  TST     (R0)+
8195  036202  005720                  3$:     TST     (R0)+           ;ADJUST THE RETURN ADDRESS
8196  036204  000402                  BR      5$              ;EXIT
8197  036206  004737  035522          4$:     JSR     PC,CI7          ;PROCESS THE ERROR
8198  036212  000200                  5$:     RTS     R0              ;RETURN
8199
8200                                  ;INTERRUPT SERVICE ROUTINE
8201
8202  036214  112737  000001  033250  ISR:    MOVB    #1,ACTDRV       ;SET "ACTIVE DRIVER" FLAG
8203  036222  104412                  SAVREG                  ;SAVE R0 - R5
8204  036224  013704  033332          MOV     RPADR,R4        ;ADDRESS OF RHSCS1
8205  036230  013701  033316          MOV     DTUW,R1         ;GET "DATA TRANSFER UNDERWAY" INDICATOR
8206  036234  002403                  BLT     1$              ;BRANCH IF NO DATA TRANSFER UNDERWAY
8207  036236  004737  036260          JSR     PC,TD           ;CALL TRANSFER DONE
8208  036242  000402                  BR      2$              ;EXIT
8209  036244  004737  036542          1$:     JSR     PC,SC           ;CALL SPECIAL CONDITIONS
8210  036250  104413                  2$:     RESREG                  ;RESTORE R0 - R5
```

```
8211  036252  105037  033250          CLRB    ACTDRV          ;CLEAR "ACTIVE DRIVER" FLAG
8212  036256  000002                  RTI                     ;RETURN
8213
8214                                  ;TRANSFER DONE ROUTINE
8215
8216  036260  105061  033174  TD:     CLRB    DRVACT(R1)      ;SET DRIVE ACTIVE INDICATOR TO IDLE
8217  036264  012737  177777  033316  MOV     #-1,DTUW        ;NO DATA TRANSFERS UNDERWAY
8218  036272  006301                  ASL     R1
8219  036274  012761  177777  033276  MOV     #-1,TIMER(R1)   ;CANCEL TIMEOUT
8220  036302  006201                  ASR     R1
8221  036304  013702  033244          MOV     TRNSWT,R2       ;GET "DPB" ADDRESS FROM THE
8222  036310  005037  033244          CLR     TRNSWT          ;TRANSFER WAIT QUEUE--CLEAR QUEUE
8223  036314  052762  000200  000016  BIS     #BIT07,16(R2)   ;SET DONE
8224  036322  010164  000010          MOV     R1,RPCS2(R4)    ;SELECT THE DRIVE
8225  036326  004037  040406          JSR     R0,RD.RP        ;TRANSFER ERROR(TRE=1)?
8226  036332  000000                  RPCS1
8227  036334  035522                  CI7
8228  036336  006126                  ROL     (SP)+
8229  036340  100421                  BMI     3$              ;BR IF YES
8230  036342  005737  033272          TST     SAVEFG          ;SAVE THE RH11/RP04/5/6 REGISTERS?
8231  036346  100002                  BPL     1$              ;BRANCH IF NO
8232  036350  004737  040750          JSR     PC,SVRH11       ;YES--SAVE THE REGISTERS
8233  036354  004737  036434  1$:     JSR     PC,WC           ;SEE IF WRITE CHECK TO BE PUT IN QUEUE
8234  036360  004737  041604          JSR     PC,GETREQ       ;GET DPB POINTER
8235  036364  005702                  TST     R2              ;ENTRY FOR DRIVE ?
8236  036366  001403                  BEQ     2$              ;BR IF NOT
8237  036370  004737  034412          JSR     PC,OPT          ;CALL OPTIMIZER
8238  036374  000462                  BR      SC              ;CHECK OTHER DRIVES
8239  036376  012714  000113  2$:     MOV     #113,(R4)       ;RELEASE THE DRIVE
8240  036402  000457                  BR      SC              ;CHECK FOR OTHER DRIVES
8241  036404  052762  100100  000016  3$:     BIS     #BIT15!BIT06,16(R2) ;SET DATA ERROR FLAG
8242  036412  004737  041510          JSR     PC,EMPTYQ       ;EMPTY THE "DRIVE'S WAIT" QUEUE
8243  036416  004737  040750          JSR     PC,SVRH11       ;SAVE THE RH11/RP04/5/6 REGISTERS
8244  036422  012714  040111          MOV     #40111,(R4)     ;ISSUE A "DRIVE CLEAR"
8245  036426  012714  000113          MOV     #113,(R4)       ;ISSUE A RELEASE TO THE DRIVE
8246  036432  000443                  BR      SC              ;CHECK FOR OTHER DRIVES
8247
8248                                  ;FORCED WRITE CHECK ROUTINE
8249
8250  036434  005737  001424  WC:     TST     AUTOCK          ;AUTOMATIC WRITE CHECKS ?
8251  036440  001437                  BEQ     2$              ;BR IF NOT
8252  036442  122762  000002  000024  CMPB    #2,$CODE(R2)    ;LAST OPERATION WRITE DATA ?
8253  036450  001404                  BEQ     1$              ;BR IF IT WAS
8254  036452  122762  000003  000024  CMPB    #3,$CODE(R2)    ;LAST OPERATION WRITE HEADER & DATA ?
8255  036460  001027                  BNE     2$              ;BR IF NOT
8256  036462  004037  041530  1$:     JSR     R0,DRVQUE       ;PUT THE OPERATION IN THE QUEUE
8257  036466  000424                  BR      2$              ;QUEUE IS FULL
8258  036470  005062  000016          CLR     16(R2)          ;CLEAR 'DONE' BIT IN DPB
8259  036474  116262  000234  000027  MOVB    $RPCS1(R2),$PREVO(R2)   ;SAVE WRITE OPERATION CODE
8260  036502  016262  000012  000034  MOV     $CYL(R2),$PREVA+2(R2)   ;SAVE CYLINDER
8261  036510  016262  000010  000032  MOV     $SEC(R2),$PREVA(R2)   ;SAVE SECTOR AND TRACK ADDRESSES
8262  036516  142762  000002  000024  BICB    #2,$CODE(R2)    ;CHANGE WRITE TO CHECK
8263  036524  142762  000020  000002  BICB    #20,$COMND(R2)  ;CHANGE DRIVER CODE TO WRITE CHECK
8264  036532  152762  000010  000002  BISB    #10,$COMND(R2)  ;FINISH CHANGING CODE TO WRITE CHECK
8265  036540  000207          2$:     RTS     PC              ;EXIT
8266
```

```
                                        ;SPECIAL CONDITION ROUTINE
                                        
036542  116403  000016      SC:   MOVB    RPAS(R4),R3     ;READ "RPAS"
036546  001012                  BNE     2$              ;BRANCH IF ANY 'ATA' BITS SET
036550  004037  040406          JSR     R0,RD.RP        ;READ CONTROL AND STATUS REGISTER
036554  000000                  RPCS1
036556  035622                  CIB
036560  106126                  ROLB    (SP)+           ;IS "IE"=1?
036562  100403                  BMI     1$              ;YES, NO DRIVES TO CHECK
036564  104001                  ERROR   1               ;REPORT AN ILLEGAL INTERRUPT
036566  004737  041066          JSR     PC,SET.IE       ;SET INTERRUPT ENABLE
036572  000207              1$: RTS     PC              ;RETURN
036574  005046              2$: CLR     -(SP)           ;PROCESS ALL DRIVES THAT HAVE
036576  110316                  MOVB    R3,(SP)         ;AN "ATA"=1
036600  012703  000001          MOV     #1,R3
036604  005001                  CLR     R1
036606  030316              SC3:BIT     R3,(SP)         ;ATA=1?
036610  001005                  BNE     SC5             ;YES--BRANCH
036612  005201              SC4:INC     R1              ;MOVE TO THE NEXT DRIVE
036614  106303                  ASLB    R3
036616  001373                  BNE     SC3             ;BRANCH IF MORE TO CHECK?
036620  005726                  TST     (SP)+           ;CLEAN OFF THE STACK
036622  000207                  RTS     PC              ;RETURN TO USER
036624  105761  033224      SC5:TSTB    DPINT(R1)       ;INITIALIZING THE DRIVE ?
036630  001402                  BEQ     1$              ;BR IF NOT
036632  000137  037520          JMP     SC13            ;PROCESS THE DRIVE
036636  105761  033234      1$: TSTB    DPRQS(R1)       ;PORT REQUEST OUTSTANDING ?
036642  001402                  BEQ     2$              ;BR IF NOT
036644  000137  037520          JMP     SC13            ;START THE OUTSTANDING COMMAND
036650  105761  033204      2$: TSTB    DRVSTA(R1)      ;CHECK THE DRIVE STATUS
036654  003025                  BGT     5$              ;BRANCH IF ONLINE
036656  105761  033252          TSTB    ULDFLG(R1)      ;UNLOAD IN PROGRESS?
036662  003422                  BLE     5$              ;BRANCH IF NOT
036664  004737  041604          JSR     PC,GETREQ       ;GET DPB POINTER
036670  004737  040750          JSR     PC,SVRH11       ;SAVE THE RH11/RP04/5/6 REGISTERS
036674  004737  037450          JSR     PC,SC12         ;SAVE RPDS1, RPER1, RPER2, AND RPER3
                                                        ;ALSO DO A DRIVE INIT (DRVINT)
036700  105761  033204          TSTB    DRVSTA(R1)      ;DID DRIVE COME ONLINE?
036704  003416                  BLE     6$              ;NO---BRANCH
036706  032737  040000  033164  BIT     #BIT14,RPERRS   ;WAS THERE AN ERROR?
036714  001002                  BNE     3$              ;BR IF ERROR
036716  000137  037360          JMP     SC11            ;NO ERROR
036722  013705  033166      3$: MOV     RPERRS+2,R5     ;YES -- PICKUP RPER1 AND
036726  000476                  BR      SC6A            ;GO PROCESS THE ERROR
036730  105761  033174      5$: TSTB    DRVACT(R1)      ;DRIVE ACTIVE WITH COMMAND OR ERROR RECOVERY ?
036734  001027                  BNE     SC6             ;BR IF EITHER
036736  004737  037450          JSR     PC,SC12         ;SAVE RPDS1, RPER1, RPER2, AND RPER3
                                                        ;ALSO DO A DRVINT
036742  105761  033224      6$: TSTB    DPINT(R1)       ;TRYING TO INIT THE DRIVE ?
036746  001321                  BNE     SC4             ;BR IF YES, CHECK ON MORE DRIVES
036750  105761  033204          TSTB    DRVSTA(R1)      ;CHECK ON DRIVE'S STATUS
036754  100412                  BMI     7$              ;BR IF UNSAFE
036756  032737  020000  033172  BIT     #BIT13,RPERRS+6 ;ADDRESS PLUG CHANGED ?
036764  001011                  BNE     8$              ;BR IF YES
036766  012746  000113          MOV     #113,-(SP)      ;RELEASE COMMAND
036772  004037  040562          JSR     R0,WRT.RP       ;WRITE THE COMMAND INTO RPCS1
```

```
8323  036776  000000                        RPCS1                   ;REGISTER INDEX
8324  037000  037330                        SC8                     ;PARITY EXIT ADDRESS
8325  037002  011605              7$:       MOV     (SP),R5         ;PICKUP (RPAS) BEFORE THE ERROR CALL
8326  037004  104002                        ERROR   2               ;REPORT THE UNEXPECTED ATTENTION
8327  037006  000701                        BR      SC4             ;GO CHECK FOR MORE ATA'S
8328  037010                        8$:
8329  037010  104005                        ERROR   5               ;REPORT THE ADDRESS PLUG CHANGE
8330  037012  000677                        BR      SC4             ;CHECK FOR MORE DRIVES
8331  037014  006301              SC6:      ASL     R1              ;SETUP TO ADDRESS WORDS
8332  037016  012761  177777  033276        MOV     #-1,TIMER(R1)   ;STOP THE TIMER
8333  037024  006201                        ASR     R1              ;RESTORE THE DRIVE ADDRESS
8334  037026  004737  041604                JSR     PC,GETREQ       ;GET THE DPB POINTER FROM THE QUEUE
8335  037032  010164  000010                MOV     R1,RPCS2(R4)    ;SELECT DRIVE
8336  037036  004037  040406                JSR     R0,RD.RP        ;READ THE RP04'S STATUS REG.
8337  037042  000012                        RPDS1
8338  037044  037330                        SC8
8339  037046  011605                        MOV     (SP),R5         ;AND PUT IT IN R5
8340  037050  006126                        ROL     (SP)+           ;WAS THERE AN ERROR?
8341  037052  100407                        BMI     1$              ;BR IF ERROR
8342  037054  105761  033174                TSTB    DRVACT(R1)      ;CHECK DRIVE'S STATE
8343  037060  003137                        BGT     SC11            ;BR IF DRIVE ACTIVE WITH ORDER
8344  037062  052762  100210  000016        BIS     #BIT15!BIT07!BIT03,16(R2)  ;INFORM USER OF ERROR RECOVER COMPLETION
8345  037070  000470                        BR      SC7
8346  037072  004037  040406        1$:     JSR     R0,RD.RP        ;READ ERROR REGISTER #1
8347  037076  000014                        RPER1
8348  037100  037330                        SC8
8349  037102  012605                        MOV     (SP)+,R5        ;AND SAVE IT IN R5
8350  037104  004737  040750                JSR     PC,SVRH11       ;SAVE RH11/RP04/5/6 REGISTERS
8351  037110  012746  000111                MOV     #111,-(SP)      ;ISSUE A DRIVE CLEAR
8352  037114  004037  040562                JSR     R0,WRT.RP
8353  037120  000000                        RPCS1
8354  037122  037330                        SC8
8355  037124  006105              SC6A:     ROL     R5              ;WAS "UNSAFE" CONDITION =1?
8356  037126  100406                        BMI     1$              ;BRANCH IF YES
8357  037130  005702                        TST     R2              ;ANYTHING IN QUEUE ?
8358  037132  001447                        BEQ     SC7             ;BR IF NOT
8359  037134  052762  100240  000016        BIS     #BIT15!BIT07!BIT05,16(R2)       ;INFORM USER OF ERROR
8360  037142  000443                        BR      SC7
8361  037144  004037  040406        1$:     JSR     R0,RD.RP        ;READ DRIVE STATUS REG. #1
8362  037150  000012                        RPDS1
8363  037152  037330                        SC8
8364  037154  011605                        MOV     (SP),R5         ;SAVE RPDS1 IN R5
8365  037156  006126                        ROL     (SP)+           ;"ERR"=1?
8366  037160  100011                        BPL     2$              ;BR IF NO--UNSAFE CLEARED
8367  037162  112761  177777  033204        MOVB    #-1,DRVSTA(R1)  ;DRIVE IS UNSAFE
8368  037170  004737  040750                JSR     PC,SVRH11       ;SAVE RH11/RP04/5/6 REGISTERS
8369  037174  052762  110000  000016        BIS     #BIT15!BIT12,16(R2)     ;INFORM USER OF UNSAFE ERROR
8370  037202  000423                        BR      SC7
8371  037204  032705  010000        2$:     BIT     #BIT12,R5       ;"MOL" = 1 ?
8372  037210  001015                        BNE     3$              ;BR IF YES
8373  037212  112761  177777  033174        MOVB    #-1,DRVACT(R1)  ;ACTIVE ERROR RECOVER
8374  037220  112761  000001  033204        MOVB    #1,DRVSTA(R1)   ;ONLINE
8375  037226  006301                        ASL     R1
8376  037230  012761  072460  033276        MOV     #30000.,TIMER(R1) ;START 30 SECOND TIMER
8377  037236  006201                        ASR     R1
8378  037240  000137  036612                JMP     SC4
```

```
 8379  037244  052762  100220  000016  3$:     BIS     #BIT15!BIT07!BIT04,16(R2)       ;INFORM USER OF ERROR
 8380  037252  105061  033174          SC7:    CLRB    DRVACT(R1)              ;DRIVE IS IDLE
 8381  037256  004737  041510                  JSR     PC,EMPTYQ               ;DUMP THE QUEUE
 8382  037262  105761  033252                  TSTB    ULDFLG(R1)              ;UNLOAD IN PROGRESS OR QUEUE?
 8383  037266  003002                          BGT     1$                      ;BR IF NOT
 8384  037270  105061  033252                  CLRB    ULDFLG(R1)              ;CLEAR UNLOAD FLAG
 8385  037274  116164  033320  000016  1$:     MOVB    ATA3IT(R1),RPAS(R4)     ;CLEAR ATTENTION BIT
 8386  037302  105761  033204                  TSTB    DRVSTA(R1)              ;IS THE DRIVE UNSAFE ?
 8387  037306  100406                          BMI     2$                      ;BR IF IT IS
 8388  037310  012746  000113                  MOV     #113,-(SP)              ;RELEASE COMMAND
 8389  037314  004037  040562                  JSR     R0,WRT.RP               ;WRITE THE COMMAND INTO RPCS1
 8390  037320  000000                          RPCS1                           ;REGISTER INDEX
 8391  037322  037330                          SC8                             ;PARITY EXIT ADDRESS
 8392  037324  000137  036612          2$:     JMP     SC4                     ;CHECK FOR MORE DRIVES
 8393  037330  105761  033174          SC8:    TSTB    DRVACT(R1)              ;IS DRIVE IDLE?
 8394  037334  001405                          BEQ     1$                      ;YES--BRANCH
 8395  037336  004737  041604                  JSR     PC,GETREQ               ;GET DPB POINTER
 8396  037342  004737  035522                  JSR     PC,CI7                  ;PROCESS THE PARITY ERROR
 8397  037346  000402                          BR      2$                      ;CONTINUE
 8398  037350  004737  035550          1$:     JSR     PC,CI7B                 ;PROCESS THE UNCORRECTABLE PARITY ERROR
 8399  037354  000137  036612          2$:     JMP     SC4                     ;CHECK MORE DRIVES
 8400  037360  105761  033252          SC11:   TSTB    ULDFLG(R1)              ;"UNLOAD IN PROGRESS"?
 8401  037364  003402                          BLE     1$                      ;BRANCH IF NO
 8402  037366  105061  033252                  CLRB    ULDFLG(R1)              ;CLEAR UNLOAD FLAG
 8403  037372  105061  033174          1$:     CLRB    DRVACT(R1)              ;SET DRIVE IDLE
 8404  037376  136137  033320  033246          BITB    ATABIT(R1),SRCHWT       ;DOING A SEARCH OPERATION FOR
 8405                                                                          ;AN I/O COMMAND?
 8406  037404  001012                          BNE     2$                      ;BRANCH IF YES
 8407  037406  004737  041626                  JSR     PC,POPQUE               ;REMOVE REQUEST FROM QUEUE
 8408  037412  052762  000200  000016          BIS     #BIT07,16(R2)           ;SET "DONE" BIT
 8409  037420  005737  033272                  TST     SAVEFG                  ;SAVE THE REGISTERS?
 8410  037424  100002                          BPL     2$                      ;BRANCH IF NO
 8411  037426  004737  040750                  JSR     PC,SVRH11               ;YES--SAVE ALL OF THE RH11/RPO4/5/6 REG'S
 8412  037432  116164  033320  000016  2$:     MOVB    ATABIT(R1),RPAS(R4)     ;CLEAR ATTENTION BIT
 8413  037440  004737  034412                  JSR     PC,OPT                  ;START A REQUEST
 8414  037444  000137  036612                  JMP     SC4                     ;CHECK FOR MORE DRIVES
 8415  037450  010164  000010          SC12:   MOV     R1,RPCS2(R4)            ;SELECT DRIVE
 8416  037454  016437  000012  033164          MOV     RPOS1(R4),RPERRS        ;SAVE THE FOUR REGISTERS THAT
 8417  037462  016437  000014  033166          MOV     RPER1(R4),RPERRS+2      ;WILL TELL US SOMETHING
 8418  037470  016437  000040  033170          MOV     RPER2(R4),RPERRS+4
 8419  037476  016437  000042  033172          MOV     RPER3(R4),RPERRS+6
 8420  037504  004037  033562                  JSR     R0,DRVINT               ;INIT. THE STATE OF THE DRIVE
 8421  037510  000401                          BR      1$                      ;TAKE ERROR EXIT
 8422  037512  000207                          RTS     PC                      ;RETURN
 8423  037514  005726                  1$:     TST     (SP)+                   ;POP PC OFF OF THE STACK
 8424  037516  000704                          BR      SC8                     ;PROCESS THE PARITY ERROR
 8425  037520  006301                  SC13:   ASL     R1                      ;SETUP TO ADDRESS WORDS
 8426  037522  012761  177777  033276          MOV     #-1,TIMER(R1)           ;STOP THE TIMER
 8427  037530  006201                          ASR     R1                      ;
 8428  037532  010164  000010                  MOV     R1,RPCS2(R4)            ;SELECT THE DRIVE
 8429  037536  116164  033320  000016          MOVB    ATABIT(R1),RPAS(R4)     ;CLEAR THE ATTENTION BIT
 8430  037544  032714  004000                  BIT     #BIT11,(R4)             ;DRIVE AVAILABLE ?
 8431  037550  001006                          BNE     1$                      ;BR IF AVAILABLE
 8432  037552  006301                          ASL     R1
 8433
 8434  037554  012761  023420  033276          MOV     #10000.,TIMER(R1)       ;START 10 SEC TIMER AGAIN
```

```
8435
8436  037562  006201                          ASR     R1
8437  037564  000433                          BR      3$                      ;EXIT
8438  037566  105761  033224          1$:     TSTB    DPINT(R1)               ;INITIALIZING THE DRIVE ?
8439  037572  001424                          BEQ     2$                      ;BR IF NOT
8440  037574  105061  033224                  CLRB    DPINT(R1)               ;CLEAR THE INIT INDICATOR
8441  037600  004037  033562                  JSR     R0,DRVINT               ;GO INIT THE DRIVE
8442  037604  000240                          NOP                             ;DUMMY PARITY ERROR RETURN
8443  037606  105761  033204                  TSTB    DRVSTA(R1)              ;DRIVE ONLINE ?
8444  037612  003014                          BGT     2$                      ;BR IF YES -- START ORDER
8445  037614  005702                          TST     R2                      ;QUEUE ENTRY FOR THE DRIVE
8446  037616  001416                          BEQ     3$                      ;BR IF NOT
8447  037620  004737  041604                  JSR     PC,GETREQ               ;GET DPB ADDRESS
8448  037624  052762  140000  000016          BIS     #BIT15!BIT14,16(R2)     ;INFORM USER THAT DRIVE OFFLINE
8449  037632  004737  040750                  JSR     PC,SVRH11               ;SAVE THE REGISTERS
8450  037636  004737  041510                  JSR     PC,EMPTYQ               ;EMPTY THE REQUEST QUEUE
8451  037642  000404                          BR      3$
8452  037644  105061  033234          2$:     CLRB    DPRQS(R1)               ;CLEAR THE PORT REQUEST INDICATOR
8453  037650  004737  034412                  JSR     PC,OPT                  ;START THE PENDING REQUEST
8454  037654  000137  036612          3$:     JMP     SC4                     ;PROCESS OTHER DRIVES
8455
8456                                  ;RP04/5/6 TIMER ROUTINE
8457                                  ;CALL
8458                                  ;       MOV     #TIME,-(SP)             ;ELASPED TIME IN MILLISECONDS ON THE STACK
8459                                  ;       JSR     PC,RPTMR                ;CALL RP04/5/6 TIME ROUTINE
8460
8461  037660  005737  033250          RPTMR:  TST     ACTDRV                  ;CHECK "ACTDRV & ACTSTR"
8462  037664  001030                          BNE     4$                      ;IF NON ZERO EXIT
8463  037666  112737  000001  033251          MOVB    #1,ACTSTR               ;SET "ACTSTR"
8464  037674  104412                          SAVREG                          ;SAVE R0 - R5
8465  037676  005001                          CLR     R1                      ;START WITH DRIVE 0
8466  037700  005003                          CLR     R3
8467  037702  005763  033276          1$:     TST     TIMER(R3)               ;IS THE TIMER RUNNING?
8468  037706  002407                          BLT     2$                      ;BRANCH IF NO
8469  037710  166663  000002  033276          SUB     2(SP),TIMER(R3)         ;COUNT THE INTERVAL
8470  037716  003003                          BGT     2$                      ;BR IF NO SOFTWARE TIMEOUT
8471  037720  004737  037752                  JSR     PC,STO                  ;CALL SOFTWARE TIMEOUT ROUTINE
8472  037724  000405                          BR      3$                      ;GO TO THE EXIT
8473  037726  005201                  2$:     INC     R1                      ;MOVE TO NEXT DRIVE
8474  037730  005723                          TST     (R3)+
8475  037732  022701  000010                  CMP     #8.,R1                  ;OUT OF DRIVES?
8476  037736  003361                          BGT     1$                      ;BRANCH IF NO
8477  037740  104413                  3$:     RESREG                          ;RESTORE R0 - R5
8478  037742  105037  033251                  CLRB    ACTSTR                  ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
8479  037746  012616                  4$:     MOV     (SP)+,(SP)              ;ADJUST THE STACK
8480  037750  000207                          RTS     PC                      ;RETURN
8481
8482                                  ;SOFTWARE TIMEOUT ROUTINE
8483                                  ;
8484                                  ;NOTE:  THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
8485                                  ;       OR GREATER
8486                                  ;
8487                                  ;CALL:  STO
8488                                  ;       MOV     #DRVNUM,R1              ;DRIVE NUMBER
8489                                  ;       JSR     PC,STO                  ;CALL
8490                                  ;       RETURN
```

```
8491
8492  037752  010146                  STO:    MOV     R1,-(SP)        ;SAVE R1
8493  037754  010346                          MOV     R3,-(SP)        ;SAVE R3
8494  037756  013704  033332                  MOV     RPADR,R4        ;GET ADDRESS OF "RPCS1"
8495  037762  010164  000010                  MOV     R1,RPCS2(R4)    ;SELECT THE DRIVE
8496  037766  004037  040406                  JSR     R0,RD.RP        ;READ "DRIVE STATUS REG"
8497  037772  000012                          RPDS1
8498  037774  040274                          STO5
8499  037776  105726                          TSTB    (SP)+           ;IS "DRY"=1?
8500  040000  100477                          BMI     STO2            ;BR IF YES
8501  040002  105761  033224          STO1:   TSTB    DPINT(R1)       ;TRYING TO INTIALIZE THE DRIVE ?
8502  040006  001074                          BNE     STO2            ;BR IF YES
8503  040010  105761  033234                  TSTB    DPRQS(R1)       ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
8504  040014  001071                          BNE     STO2            ;BR IF YES
8505  040016  013702  033244                  MOV     TRNSWT,R2       ;PICKUP TRANSFER WAIT QUEUE
8506  040022  020137  033316                  CMP     R1,DTUW         ;TRANSFER UNDERWAY ON THIS DRIVE?
8507  040026  001402                          BEQ     1$              ;BRANCH IF YES
8508  040030  004737  041604                  JSR     PC,GETREQ       ;GET DPB ADDRESS
8509  040034  052762  101000  000016  1$:     BIS     #BIT15!BIT09,16(R2) ;SET THE ERROR FLAGS
8510  040042  004737  040750                  JSR     PC,SVRH11       ;SAVE RH11/RP04/5/6 REGISTERS
8511  040046  012764  000040  000010          MOV     #BIT05,RPCS2(R4) ;"INIT" THE MASS BUS
8512  040054  105061  033174                  CLRB    DRVACT(R1)      ;DRIVE IS IDLE
8513  040060  105061  033252                  CLRB    ULDFLG(R1)      ;CLEAR THE UNLOAD FLAG
8514  040064  005001                          CLR     R1              ;START WITH DRIVE 0
8515  040066  005003                          CLR     R3
8516  040070  004037  033562          2$:     JSR     R0,DRVINT       ;INIT. THIS DRIVE
8517  040074  000477                          BR      STO5            ;PARITY ERROR RETURN
8518  040076  105761  033174                  TSTB    DRVACT(R1)      ;DRIVE IDLE BEFORE THE INIT.?
8519  040102  001414                          BEQ     4$              ;YES--BRANCH
8520  040104  013702  033244                  MOV     TRNSWT,R2       ;GET TRANSFER WAIT QUEUE
8521  040110  023701  033316                  CMP     DTUW,R1         ;WAS THERE I/O ON THIS DRIVE?
8522  040114  001402                          BEQ     3$              ;YES--BRANCH
8523  040116  004737  041604                  JSR     PC,GETREQ       ;GET THE DPB POINTER FROM QUEUE
8524  040122  052762  100400  000016  3$:     BIS     #BIT15!BIT08,16(R2) ;INFORM USER OF INIT.
8525  040130  105061  033174                  CLRB    DRVACT(R1)      ;SET DRIVE ACTIVE TO IDLE
8526  040134  105061  033252          4$:     CLRB    ULDFLG(R1)      ;NO UNLOAD
8527  040140  012763  177777  033276          MOV     #-1,TIMER(R3)   ;STOP THE TIMER
8528  040146  005723                          TST     (R3)+           ;UPDATE THE INDEX
8529  040150  005201                          INC     R1              ;INCREMENT THE DRIVE NUMBER
8530  040152  022701  000010                  CMP     #8.,R1          ;LAST DRIVE BEEN CHECKED?
8531  040156  003344                          BGT     2$              ;NO--LOOP
8532  040160  012737  177777  033316          MOV     #-1,DTUW        ;NO DATA TRANSFERS UNDERWAY
8533  040166  005037  033244                  CLR     TRNSWT          ;CLEAR TRANSFER WAIT QUEUE
8534  040172  004737  041432                  JSR     PC,CLRQUE       ;CLEAR ALL REQUEST QUEUES
8535  040176  000500                          BR      STO9            ;EXIT
8536  040200  116405  000016          STO2:   MOVB    RPAS(R4),R5     ;READ ATTENTION REG
8537  040204  136105  033320                  BITB    ATABIT(R1),R5   ;IS ATTENTION FOR THIS DRIVE UP ?
8538  040210  001017                          BNE     STO3            ;YES--BRANCH
8539  040212  105761  033224                  TSTB    DPINT(R1)       ;TRYING TO INTIALIZE THE DRIVE ?
8540  040216  001031                          BNE     STO6            ;BR IF YES - DRIVE NOT ONLINE
8541  040220  105761  033234                  TSTB    DPRQS(R1)       ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
8542  040224  001045                          BNE     STO7            ;BR IF YES - NO RESPONSE TO REQUEST
8543  040226  020137  033316                  CMP     R1,DTUW         ;DATA TRANSFER UNDERWAY FOR THIS DRIVE
8544  040232  001263                          BNE     STO1            ;BR IF NO
8545  040234  004037  040406                  JSR     R0,RD.RP        ;YES--CHECK "RDY"
8546  040240  000000                          RPCS1
```

```
                                                ST05
040242  040274                                  TSTB    (SP)+
040244  105726                                  BPL     ST01                    ;BR IF "RDY"=0
040246  100255                                  TSTB    DPINT(R1)               ;INITIALIZING THE DRIVE ?
040250  105761  033224          ST03:           BNE     1$                      ;BR IF INIT PENDING
040254  001003                                  TSTB    DPRQS(R1)               ;PORT REQUEST PENDING ?
040256  105761  033234                          BEQ     ST09                    ;BR IF NOT
040262  001446                                  MOV     #-1,TIMER(R3)           ;STOP THE TIMER
040264  012763  177777  033276  1$:             BR      ST09                    ;EXIT
040272  000442                                  JSR     PC,CI8                  ;GO HANDLE THE PARITY ERROR
040274  004737  035622          ST05:           BR      ST09
040300  000437                          ST06:   CLRB    DPINT(R1)               ;CLEAR THE INITIALIZE INDICATOR
040302  105061  033224                          CLRB    DRVSTA(R1)              ;SET UNIT OFFLINE
040306  105061  033204                          MOV     #-1,TIMER(R3)   ;STOP THE TIMER
040312  012763  177777  033276                  JSR     PC,GETREQ               ;GET THE DPB ADDRESS
040320  004737  041604                          TST     R2                      ;REQUEST IN QUEUE ?
040324  005702                                  BEQ     ST09                    ;BR IF NOT
040326  001424                                  BIS     #BIT15!BIT14,16(R2)     ;INFORM THE USER DRIVE NOT AVAILABLE
040330  052762  140000  000016                  BR      ST08                    ;FINISH
040336  000414                          ST07:   MOV     #-1,TIMER(R3)           ;STOP THE TIMER
040340  012763  177777  033276                  CLRB    DPRQS(R1)               ;CLEAR PORT REQUEST INDICATOR
040346  105061  033234                          JSR     PC,GETREQ               ;GET DPB ADDRESS
040352  004737  041604                          TST     R2                      ;QUEUE ENTRY FOR DRIVE ?
040356  005702                                  BEQ     ST09                    ;BR IF NONE
040360  001407                                  MOV     #BIT15!BIT2,16(R2)      ;INFORM USER OF PORT REQUEST ERROR
040362  012762  100004  000016          ST08:   JSR     PC,EMPTYQ               ;CLEAR THE QUEUE FOR THE DRIVE
040370  004737  041510                          JSR     PC,SVRH11               ;SAVE THE REGISTERS
040374  004737  040750          ST09:           MOV     (SP)+,R3                ;RESTORE R3
040400  012603                                  MOV     (SP)+,R1                ;RESTORE R1
040402  012601                                  RTS     PC                      ;RETURN
040404  000207
                                        ;ROUTINE TO READ A RH11/RP04/5/6 REGISTER
                                        ;
                                        ;CALL
                                        ;       JSR     R0,RD.RP                ;GO READ A REGISTER
                                        ;       INDEX                           ;REG. INDEX FROM BASE
                                        ;       ERRADR                          ;ERROR ADDRESS--PROCESS ERROR STARTING
                                        ;                                       ;AT THIS ADDRESS
                                        ;       RETURN                          ;CONTENTS OF REG. IS ON THE STACK
                                        ;
040406  013737  033330  040550  RD.RP:   MOV     MCPEMX,RD.RP2           ;MAX. RETRYS ALLOWED
040414  011646                          MOV     (SP),-(SP)              ;SAVE R0 FOR RETURN
040416  013737  033332  040432          MOV     RPADR,RD.ADR            ;FORM THE DESIRED ADDRESS
040424  062037  040432                  ADD     (R0)+,RD.ADR            ;USING THE BASE AND THE INDEX
040430  013727                  RD.RP1: MOV     @(PC)+,(PC)+            ;READ THE DESIRED REGISTER OF THE RP04
040432  000000                  RD.ADR: .WORD   0                       ;ADDRESS IS FORMED HERE
040434  000000                  RD.WRD: .WORD   0                       ;REG. CONTENTS PUT HERE
040436  013766  040434  000002          MOV     RD.WRD,2(SP)            ;RETURN IT TO THE USER
040444  013746  033332                  MOV     RPADR,-(SP)             ;PUT THE ADDRESS ON THE STACK
040450  062716  000010                  ADD     #RPCS2,(SP)             ;FORM THE ADDRESS OF RPCS2
040454  032736  010000                  BIT     #BIT12,@(SP)+           ;CHECK THE 'NED' BIT
040460  001035                          BNE     RD.RP3                  ;BR IF DRIVE NON-EXISTENT
040462  017746  172644                  MOV     @RPADR,-(SP)            ;READ RPCS1
040466  032716  020000                  BIT     #BIT13,(SP)             ;DID MCPE SET?
040472  001002                          BNE     1$                      ;BRANCH IF YES
040474  022620                          CMP     (SP)+,(R0)+             ;ADJUST FOR RETURN
```

```
8603   040476   000430                          BR      RD.RP4          ;EXIT
8604   040500                           1$:
8605   040500   104003                          ERROR   3               ;REPORT "MCPE" ERROR
8606   040502   005737   033316                 TST     DTUW            ;DATA TRANSFER UNDERWAY?
8607   040506   100405                          BMI     2$              ;NO--BRANCH
8608   040510   032716   040000                 BIT     #BIT14,(SP)     ;NO--"TRE"=1?
8609   040514   001402                          BEQ     2$              ;NO--BRANCH
8610   040516   005726                          TST     (SP)+           ;YES--CLEAN OFF THE STACK AND
8611   040520   000415                          BR      RD.RP3          ;TAKE THE FATAL ERROR EXIT
8612   040522   052716   040000          2$:    BIS     #BIT14,(SP)     ;CLEAR "MCPE" BY SENDING A "1" TO "TRE"
8613   040526   000316                          SWAB    (SP)            ;POSITION BEFORE WRITING
8614   040530   013737   033332   040544        MOV     RPADR,3$        ;FORM ADDRESS OF HIGH BYTE
8615   040536   005237   040544                 INC     3$
8616   040542   112637                          MOVB    (SP)+,@(PC)+    ;WRITE THE HIGH BYTE OF RPCS1
8617   040544   000000           3$:    .WORD   0               ;ADDRESS STORAGE
8618   040546   005327                          DEC     (PC)+           ;EXCEEDED MAX. RETRYS
8619   040550   000003           RD.RP2: .WORD   3
8620   040552   002326                          BGE     RD.RP1          ;BRANCH IF NO
8621   040554   011000           RD.RP3: MOV     (R0),R0         ;FATAL ERROR EXIT
8622   040556   012616                          MOV     (SP)+,(SP)
8623   040560   000200           RD.RP4: RTS     R0
8624
8625                             ;ROUTINE TO WRITE A REGISTER
8626                             ;
8627                             ;CALL
8628                             ;       MOV     DATA,-(SP)      ;DATA TO BE LOADED ON THE STACK
8629                             ;       JSR     R0,WRT.RP       ;CALL THE ROUTINE TO LOAD(WRITE) THE REG.
8630                             ;       INDEX                   ;INDEX OF THE REGISTER TO BE LOADED
8631                             ;       ERRADR                  ;ADDRESS TO RETURN TO ON AN ERROR
8632                             ;       RETURN                  ;ERROR FREE RETURN
8633
8634   040562   013737   033330   040732  WRT.RP: MOV     MCPEMX,WRT.R2   ;MAX RETRYS ALLOWED
8635   040570   016637   000002   040650          MOV     2(SP),WRT.WD    ;SAVE THE WORD TO WRITE
8636   040576   012616                          MOV     (SP)+,(SP)      ;ADJUST THE STACK
8637   040600   012037   040652                 MOV     (R0)+,WRT.AD    ;GET INDEX OF REGISTER TO BE WRITTEN
8638   040604   001015                          BNE     1$              ;BRANCH IF NOT RPCS1
8639   040606   122737   000150   040650        CMPB    #150,WRT.WD     ;IS THE COMMAND FOR DATA TRANSFERS?
8640   040614   002411                          BLT     1$              ;YES--DON'T GET THE OLD A16 & A17, & PSEL
8641   040616   004037   040406                 JSR     R0,RD.RP        ;NO---COMBINE A16&A17, & PSEL WITH
8642   040622   000000                          RPCS1                   ;THE COMMAND BEFORE SENDING IT TO
8643   040624   040740                          WRT.R3                  ;THE RH11/RP04
8644   040626   000316                          SWAB    (SP)
8645   040630   042716   177770                 BIC     #^C7,(SP)
8646   040634   112637   040651                 MOVB    (SP)+,WRT.WD+1
8647   040640   063737   033332   040652  1$:    ADD     RPADR,WRT.AD    ;FORM THE ADDRESS OF THE DISK REG.
8648   040646   012737           WRT.R1: MOV     (PC)+,@(PC)+    ;LOAD THE DESIRED REG.
8649   040650   000000           WRT.WD: .WORD   0               ;WORD TO WRITE GOES HERE
8650   040652   000000           WRT.AD: .WORD   0               ;ADDRESS IS FORMED HERE
8651   040654   013746   033332                 MOV     RPADR,-(SP)     ;PUT THE ADDRESS ON THE STACK
8652   040660   062716   000010                 ADD     #RPCS2,(SP)     ;FORM THE ADDRESS OF RPCS2
8653   040664   032736   010000                 BIT     #BIT12,@(SP)+   ;CHECK THE 'NED' BIT
8654   040670   001023                          BNE     WRT.R3          ;BR IF DRIVE NON-EXISTENT
8655   040672   004037   040406                 JSR     R0,RD.RP        ;CHECK FOR PARITY ERROR ON WRITE
8656   040676   000014                          RPER1
8657   040700   040740                          WRT.R3
8658   040702   032726   000010                 BIT     #BIT03,(SP)+
```

```
8659  040706  001416                        BEQ     WRT.R4          ;BRANCH IF "PAR=0"
8660  040710  016037  177776  040722        MOV     -2(R0),1$       ;PICKUP THE INDEX
8661  040716  004037  040406                JSR     R0,RD.RP        ;READ THE REG.
8662  040722  000000                1$:     .WORD   0               ;REG. INDEX
8663  040724  040740                        WRT.R3                  ;RETURN TO THIS ADDRESS ON ERROR
8664  040726  104004                        ERROR   4               ;REPORT THE PARITY ON WRITE ERROR
8665  040730  005327                        DEC     (PC)+           ;DECREMENT THE ERROR COUNT
8666  040732  000003                WRT.R2: .WORD   3               ;RETRY COUNTER
8667  040734  002344                        BGE     WRT.R1          ;TRY AGAIN IF NOT FINISHED
8668  040736  005726                        TST     (SP)+           ;CLEAN OFF THE STACK
8669  040740  011000                WRT.R3: MOV     (R0),R0         ;TAKE THE "PARITY ON WRITE" ERROR EXIT
8670  040742  000401                        BR      WRT.R5          ;EXIT
8671  040744  005720                WRT.R4: TST     (R0)+           ;ADJUST FOR ERROR FREE EXIT
8672  040746  000200                WRT.R5: RTS     R0
8673
8674                                ;ROUTINE TO SAVE THE RH11/RP04/5/6 REGISTERS AS PER DPB+14
8675                                ;
8676                                ;CALL
8677                                ;       MOV     #DPBNUM,R2      ;DPB POINTER TO R2
8678                                ;       JSR     PC,SVRH11       ;SAVE THE DRIVES REG'S
8679
8680  040750  104412                SVRH11: SAVREG                  ;SAVE R0 - R5
8681  040752  005702                        TST     R2              ;QUEUE ENTRY FOR THE DRIVE ?
8682  040754  001430                        BEQ     4$              ;BR IF NONE
8683  040756  013704  033332                MOV     RPADR,R4
8684  040762  111264  000010                MOVB    (R2),RPCS2(R4)  ;SELECT DRIVE
8685  040766  016203  000014                MOV     14(R2),R3       ;GET THE ERROR TABLE POINTER
8686  040772  001433                        BEQ     6$              ;EXIT IF NO ADDRESS
8687  040774  005037  041030                CLR     3$              ;COUNTER & POINTER
8688  041000  023727  041030  000022  1$:   CMP     3$,#RPDB        ;REACHED THE BUFFER REGISTER ?
8689  041006  001006                        BNE     2$              ;BR IF NOT
8690  041010  032764  000200  000010        BIT     #BIT07,RPCS2(R4)  ;'OR' SET ?
8691  041016  001002                        BNE     2$              ;BR IF SET
8692  041020  005023                        CLR     (R3)+           ;STORE RPDB AS ZEROES
8693  041022  000405                        BR      4$              ;CONTINUE
8694  041024  004037  040406        2$:     JSR     R0,RD.RP        ;READ THE SELECTED REGISTER
8695  041030  000000                3$:     .WORD   0               ;REGISTER INDEX
8696  041032  041056                        5$                      ;ERROR RETURN ADDRESS
8697  041034  012623                        MOV     (SP)+,(R3)+     ;STORE THE REGISTER CONTENTS
8698  041036  023727  041030  000046  4$:   CMP     3$,#RPEC2       ;REACHED THE END ?
8699  041044  001406                        BEQ     6$              ;BR IF YES
8700  041046  062737  000002  041030        ADD     #2,3$           ;INCREMENT THE REGISTER INDEX
8701  041054  000751                        BR      1$              ;CONTINUE READING THE REGISTERS
8702  041056  004737  035522        5$:     JSR     PC,CI7          ;PROCESS THE UNCORRECTABLE PARITY ERROR
8703  041062  104413                6$:     RESREG                  ;RESTORE R0 - R5
8704  041064  000207                        RTS     PC              ;RETURN
8705
8706                                ;ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A "TRE"
8707                                ;CALL
8708                                ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
8709                                ;       JSR     PC,SET.IE       ;SET "IE"
8710                                ;       RETURN
8711
8712  041066  010446                SET.IE: MOV     R4,-(SP)        ;SAVE R4
8713  041070  013704  033332                MOV     RPADR,R4        ;PICKUP ADDRESS OF RPCS1
8714  041074  010164  000010                MOV     R1,RPCS2(R4)    ;SELECT DRIVE
```

```
8715  041100  011446                          MOV     (R4),-(SP)      ;READ RPCS1
8716  041102  052716  040000                  BIS     #BIT14,(SP)     ;SET THE "TRE" BIT OF THE WORD READ
8717  041106  000316                          SWAB    (SP)            ;ADJUST FOR DATO
8718  041110  112714  000100                  MOVB    #BIT06,(R4)     ;SET "IE"
8719  041114  032764  010000  000010          BIT     #BIT12,RPCS2(R4) ;IS "NED"=1?
8720  041122  001002                          BNE     1$              ;YES--CLEAR "TRE"
8721  041124  005726                          TST     (SP)+           ;CLEAN OFF THE STACK
8722  041126  000402                          BR      2$
8723  041130  112664  000001          1$:     MOVB    (SP)+,1(R4)     ;CLEAR "TRE"
8724  041134  012604                  2$:     MOV     (SP)+,R4        ;RESTORE R4
8725  041136  000207                          RTS     PC              ;RETURN TO CALLER
8726
8727                                  ;QUEUE COUNT
8728  041140      000                 QCNT:   .BYTE   0               ;DRIVE 0
8729  041141      000                         .BYTE   0               ;DRIVE 1
8730  041142      000                         .BYTE   0               ;DRIVE 2
8731  041143      000                         .BYTE   0               ;DRIVE 3
8732  041144      000                         .BYTE   0               ;DRIVE 4
8733  041145      000                         .BYTE   0               ;DRIVE 5
8734  041146      000                         .BYTE   0               ;DRIVE 6
8735  041147      000                         .BYTE   0               ;DRIVE 7
8736
8737                                  ;QUEUE INPUT POINTERS
8738
8739  041150  041232                  QINPT:  .WORD   QDRV0            ;DRIVE 0
8740  041152  041252                          .WORD   QDRV1           ;DRIVE 1
8741  041154  041272                          .WORD   QDRV2           ;DRIVE 2
8742  041156  041312                          .WORD   QDRV3           ;DRIVE 3
8743  041160  041332                          .WORD   QDRV4           ;DRIVE 4
8744  041162  041352                          .WORD   QDRV5           ;DRIVE 5
8745  041164  041372                          .WORD   QDRV6           ;DRIVE 6
8746  041166  041412                          .WORD   QDRV7           ;DRIVE 7
8747
8748                                  ;QUEUE OUTPUT POINTERS
8749
8750  041170  041232                  QOUTPT: .WORD   QDRV0            ;DRIVE 0
8751  041172  041252                          .WORD   QDRV1           ;DRIVE 1
8752  041174  041272                          .WORD   QDRV2           ;DRIVE 2
8753  041176  041312                          .WORD   QDRV3           ;DRIVE 3
8754  041200  041332                          .WORD   QDRV4           ;DRIVE 4
8755  041202  041352                          .WORD   QDRV5           ;DRIVE 5
8756  041204  041372                          .WORD   QDRV6           ;DRIVE 6
8757  041206  041412                          .WORD   QDRV7           ;DRIVE 7
8758
8759  041210  041232                  QSTART: .WORD   QDRV0            ;DRIVE 0 START ADDRESS
8760  041212  041252                  QSTOP:  .WORD   QDRV1            ;DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
8761  041214  041272                          .WORD   QDRV2           ;STOP DRIVE 1--START DRIVE 2
8762  041216  041312                          .WORD   QDRV3           ;STOP DRIVE 2--START DRIVE 3
8763  041220  041332                          .WORD   QDRV4           ;STOP DRIVE 3--START DRIVE 4
8764  041222  041352                          .WORD   QDRV5           ;STOP DRIVE 4--START DRIVE 5
8765  041224  041372                          .WORD   QDRV6           ;STOP DRIVE 5--START DRIVE 6
8766  041226  041412                          .WORD   QDRV7           ;STOP DRIVE 6--START DRIVE 7
8767  041230  041432                          .WORD   QTERM           ;STOP DRIVE 7
8768
8769                                  ;DRIVE REQUEST QUEUES
8770
```

```
8771   041232  000010                  QDRV0:  .BLKW   10
8772   041252  000010                  QDRV1:  .BLKW   10
8773   041272  000010                  QDRV2:  .BLKW   10
8774   041312  000010                  QDRV3:  .BLKW   10
8775   041332  000010                  QDRV4:  .BLKW   10
8776   041352  000010                  QDRV5:  .BLKW   10
8777   041372  000010                  QDRV6:  .BLKW   10
8778   041412  000010                  QDRV7:  .BLKW   10
8779           041432                  QTERM=.
8780
8781                                   ;ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
8782                                   ;
8783                                   ;CALL
8784                                   ;       JSR     PC,CLRQUE
8785
8786   041432  104412                  CLRQUE: SAVREG                  ;SAVE R0 - R5
8787   041434  012702  041140                  MOV     #QCNT,R2        ;ZERO THE QUEUE COUNTS
8788   041440  005022                          CLR     (R2)+           ;DRIVES 0 & 1
8789   041442  005022                          CLR     (R2)+           ;DRIVES 2 & 3
8790   041444  005022                          CLR     (R2)+           ;DRIVES 4 & 5
8791   041446  005022                          CLR     (R2)+           ;DRIVES 6 & 7
8792   041450  012703  000010                  MOV     #8.,R3          ;MOVE THE STARTING
8793   041454  012701  041210                  MOV     #QSTART,R1      ;ADDRESS OF THE QUEUE INTO
8794   041460  012122                  1$:     MOV     (R1)+,(R2)+     ;THE QUEUE INPUT POINTER
8795   041462  005303                          DEC     R3
8796   041464  001375                          BNE     1$
8797   041466  012703  000010                  MOV     #8.,R3          ;MOVE THE STARTING ADDRESS
8798   041472  012701  041210                  MOV     #QSTART,R1      ;OF THE QUEUE INTO THE
8799   041476  012122                  2$:     MOV     (R1)+,(R2)+     ;QUEUE OUTPUT POINTER
8800   041500  005303                          DEC     R3
8801   041502  001375                          BNE     2$
8802   041504  104413                          RESREG                  ;RESTORE R0 - R5
8803   041506  000207                          RTS     PC
8804
8805                                   ;EMPTY THE QUEUE SPECIFIED BY R1
8806                                   ;
8807                                   ;CALL
8808                                   ;       MOV     DRVNUM,R1       ;DRIVE NUMBER TO R1
8809                                   ;       JSR     PC,EMPTYQ
8810
8811   041510  105061  041140          EMPTYQ: CLRB    QCNT(R1)        ;CLEAR NUMBER OF ITEMS IN QUEUE
8812   041514  006301                          ASL     R1
8813   041516  016161  041150  041170          MOV     QINPT(R1),QOUTPT(R1) ;SET OUTPUT QUEUE POINTER=INPUT POINTER
8814   041524  006201                          ASR     R1
8815   041526  000207                          RTS     PC
8816
8817                                   ;ROUTINE TO PUT A REQUEST IN QUEUE
8818                                   ;
8819                                   ;CALL
8820                                   ;       MOV     #DRVNUM,R1      ;DRIVE NUMBER
8821                                   ;       MOV     #DPB,R2         ;ADDRESS OF PARAMETER BLOCK
8822                                   ;       JSR     R0,DRVQUE       ;GO PUT REQUEST IN QUEUE
8823                                   ;       RETURN1                 ;RETURN HERE IF QUEUE IS FULL
8824                                   ;       RETURN2                 ;RETURN HERE IF REQUEST IS IN QUEUE
8825
8826   041530  122761  000010  041140  DRVQUE: CMPB    #10,QCNT(R1)    ;IS QUEUE FULL?
```

```
8827  041536  001421                        BEQ     2$                  ;BR IF YES-TAKE RETURN1
8828  041540  105261  041140                INCB    QCNT(R1)            ;INCREMENT QUEUE COUNT
8829  041544  006301                        ASL     R1
8830  041546  010271  041150                MOV     R2,@QINPT(R1)       ;PUT THIS REQUEST IN QUEUE
8831  041552  062761  000002  041150        ADD     #2,QINPT(R1)        ;UPDATE THE QUEUE POINTER
8832  041560  026161  041150  041212        CMP     QINPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER
8833  041566  001003                        BNE     1$                  ;BRANCH IF NO
8834  041570  016161  041210  041150        MOV     QSTART(R1),QINPT(R1) ;YES--RESET POINTER
8835  041576  006201                1$:     ASR     R1
8836  041600  005720                        TST     (R0)+               ;TAKE RETURN 2
8837  041602  000200                2$:     RTS     R0                  ;RETURN TO USER
8838
8839                                ;ROUTINE TO GET THE "DPB" ADDRESS OF NEXT REQUEST IN QUEUE
8840                                ;
8841                                ;CALL
8842                                ;       MOV     #DRVNUM,R1          ;DRIVE NUMBER TO R1
8843                                ;       JSR     PC,GETREQ           ;GO GET THE REQUEST
8844                                ;       RETURN                      ;R2="DPB" ADDRESS OF THE REQUEST
8845                                ;                                   ;R2=0 IF NO REQUEST IN QUEUE
8846
8847  041604  005002                GETREQ: CLR     R2
8848  041606  105761  041140                TSTB    QCNT(R1)            ;IS THERE ANY REQUEST IN QUEUE?
8849  041612  001404                        BEQ     2$                  ;NO---BRANCH
8850  041614  006301                1$:     ASL     R1
8851  041616  017102  041170                MOV     @QOUTPT(R1),R2      ;PICKUP "DPB" POINTER FOR THIS DRIVE
8852  041622  006201                        ASR     R1
8853  041624  000207                2$:     RTS     PC                  ;RETURN TO USER
8854
8855                                ;ROUTINE TO "POP" THE REQUEST FROM QUEUE
8856                                ;
8857                                ;CALL
8858                                ;       MOV     #DRVNUM,R1          ;DRIVE NUMBER TO R1
8859                                ;       JSR     PC,POPQUE           ;CALL TO REMOVE REQUEST
8860                                ;       RETURN                      ;R2=ADDRESS OF DPB REMOVED
8861
8862  041626  105361  041140        POPQUE: DECB    QCNT(R1)            ;DECREMENT QUEUE COUNT
8863  041632  006301                        ASL     R1
8864  041634  017102  041170                MOV     @QOUTPT(R1),R2      ;GET THE "DPB" POINTER
8865  041640  062761  000002  041170        ADD     #2,QOUTPT(R1)       ;UPDATE THE QUEUE POINTER
8866  041646  026161  041170  041212        CMP     QOUTPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER?
8867  041654  001003                        BNE     1$                  ;NO--BRANCH TO EXIT
8868  041656  016161  041210  041170        MOV     QSTART(R1),QOUTPT(R1) ;YES--RESET THE POINTER
8869  041664  006201                1$:     ASR     R1
8870  041666  000207                        RTS     PC                  ;RETURN TO USER
8871
8872
8873                                ;;*****************************************************************
8874
8875                                .SBTTL  DATA, CONTROL, & STATUS BLOCKS
8876
8877                                ;;*****************************************************************
8878
8879                                ;BLOCK LOCATION EQUATE STATEMENTS
8880
8881          000001                $FMT    =       1                   ;FMT,HCI,ECI OR OFFSET CODE
8882          000002                $COMND  =       $FMT+1              ;OPERATION CODE
```

```
                        $PSEL   =       $FMT+2          ;PORT SELECT & BITS A16, A17
                        $WRDM   =       $FMT+3          ;WORD COUNT (2'S COMP)
                        $BUF    =       $FMT+5          ;BUFFER ADDR OR REGISTER TABLE POINTER
                        $SEC    =       $FMT+7          ;SECTOR ADDRESS OR 1ST REG ADDR
                        $TRK    =       $FMT+10         ;TRACK ADDRESS OF LAST REG ADDR
                        $CYL    =       $FMT+11         ;CYLINDER ADDR
                        $REG    =       $FMT+13         ;REGISTER STORAGE (IF ERROR)
                        $STATUS =       $FMT+15         ;STATUS WORD (SET BY DRIVER)

                ;DRIVE'S HISTORY AND CURRENT INDICATOR STORAGE EQUATES

                        $WRDL   =       $FMT+17         ;WORD COUNT (NOT 2'S COMP)
                        $SSEC   =       $WRDL+2         ;SECTOR SIZE FOR CURRENT OPERATION
                        $CODE   =       $WRDL+4         ;PRESENT COMMAND SELECTION CODE
                        $PACK   =       $WRDL+6         ;WRITE DATA PACK INDICATOR
                        $PREVO  =       $WRDL+7         ;PREVIOUS COMMAND SELECTION CODE
                        $PATTC  =       $WRDL+10        ;PATTERN CODE
                        $PREVA  =       $WRDL+12        ;PREVIOUS ADDRESS - TRACK, SECTOR, CYLINDER
                        $OPERC  =       $WRDL+16        ;OPERATION COUNT
                        $POSIT  =       $WRDL+22        ;SEEK COUNT
                        $TRANS  =       $WRDL+26        ;TOTAL BITS XFERED COUNT (R & W)
                        $READ   =       $WRDL+32        ;TOTAL BITS READ COUNT
                        $TOTAL  =       $WRDL+36        ;TOTAL ERRORS (ALL TYPES) COUNT
                        $SOFT   =       $WRDL+40        ;'SOFT' ERROR COUNT
                        $HARD   =       $WRDL+42        ;'HARD' ERROR COUNT
                        $SKI    =       $WRDL+44        ;'SKI' OR 'OCYL' ERROR COUNT
                        $MISPO  =       $WRDL+46        ;PROG DETECTED MISPOSITIONING ERROR S COUNT
                        $PASSC  =       $WRDL+50        ;PASS COUNTER
                        $FAIR   =       $WRDL+52        ;OPERATION QUEUE 'FAIRNESS' COUNT

                ;INDEX EQUATES TO THE NEXT OPERATION PARAMETERS

                        $NCODE  =       $WRDL+54        ;NEXT OPERATION CODE
                        $NPATC  =       $NCODE+1        ;NEXT PATTERN
                        $NSEC   =       $NCODE+2        ;NEXT SECTOR
                        $NTRK   =       $NCODE+3        ;NEXT TRACK
                        $NCYL   =       $NCODE+4        ;NEXT CYLINDER
                        $NWRDL  =       $NCODE+6        ;NEXT BUFFER SIZE
                        $NEXT   =       $NCODE+10       ;PARAMETER SELECTION INDICATOR

                ;INDEX EQUATES FOR MAXIMUM/MINIMUM ADDRESSES

                        MAXCYL  =       $NCODE+12       ;MAXIMUM CYLINDER ADDRESS
                        MINCYL  =       MAXCYL+2        ;MINIMUM CYLINDER ADDRESS
                        MAXTRK  =       MAXCYL+4        ;MAXIMUM TRACK ADDRESS
                        MINTRK  =       MAXCYL+6        ;MINIMUM TRACK ADDRESS
                        MAXSEC  =       MAXCYL+10       ;MAXIMUM SECTOR ADDRESS
                        MINSEC  =       MAXCYL+12       ;MINIMUM SECTOR ADDRESS
                        $FIRST  =       MAXCYL+14       ;FIRST OPERATION INDICATOR

                ;BAD SECTOR/TRACK ADDRESS STORAGE AREA INDEX EQUATE

                        $BDSEC  =       MAXCYL+16       ;BAD SECTOR STORAGE TABLE

                ;DRIVE ID AREA INDEX EQUATE
```

```
                        000224                  $DRVID  =       $BDSEC+100      ;DRIVE ID
                                                ;RH11/RP04/5/6 REGISTER EQUATES
                        000234                  $RPCS1  =       $DRVID+10       ;RP04 REGISTER STORAGE
                        000236                  $RPWC   =       $RPCS1+2
                        000240                  $RPBA   =       $RPCS1+4
                        000242                  $RPDA   =       $RPCS1+6
                        000244                  $RPCS2  =       $RPCS1+10
                        000246                  $RPDS1  =       $RPCS1+12
                        000250                  $RPER1  =       $RPCS1+14
                        000252                  $RPAS   =       $RPCS1+16
                        000254                  $RPLA   =       $RPCS1+20
                        000256                  $RPDB   =       $RPCS1+22
                        000260                  $RPMR   =       $RPCS1+24
                        000262                  $RPDT   =       $RPCS1+26
                        000264                  $RPSN   =       $RPCS1+30
                        000266                  $RPOF   =       $RPCS1+32
                        000270                  $RPCA   =       $RPCS1+34
                        000272                  $RPCC   =       $RPCS1+36
                        000274                  $RPER2  =       $RPCS1+40
                        000276                  $RPER3  =       $RPCS1+42
                        000300                  $RPEC1  =       $RPCS1+44
                        000302                  $RPEC2  =       $RPCS1+46
                                                ;BLOCK FOR DRIVE 0
041670      000     000                         DRIVE0: .BYTE   0,0             ;DRIVE NUMBER
041672  000005                                          .BLKW   5
041704  042124                                          .WORD   .+$RPCS1-$REG
041706  000266                                          .BLKB   $RPEC2-$REG
                                                ;BLOCK FOR DRIVE 1
042174      001     000                         DRIVE1: .BYTE   1,0             ;DRIVE NUMBER
042176  000005                                          .BLKW   5
042210  042430                                          .WORD   .+$RPCS1-$REG
042212  000266                                          .BLKB   $RPEC2-$REG
                                                ;BLOCK FOR DRIVE 2
042500      002     000                         DRIVE2: .BYTE   2,0             ;DRIVE NUMBER
042502  000005                                          .BLKW   5
042514  042734                                          .WORD   .+$RPCS1-$REG
042516  000266                                          .BLKB   $RPEC2-$REG
                                                ;BLOCK FOR DRIVE 3
043004      003     000                         DRIVE3: .BYTE   3,0             ;DRIVE NUMBER
043006  000005                                          .BLKW   5
043020  043240                                          .WORD   .+$RPCS1-$REG
043022  000266                                          .BLKB   $RPEC2-$REG
```

```
                                                ;BLOCK FOR DRIVE 4

043310     004      000                 DRIVE4: .BYTE   4,0             ;DRIVE NUMBER
043312  000005                                  .BLKW   5
043324  043544                                  .WORD   .+$RPCS1-$REG
043326  000266                                  .BLKB   $RPEC2-$REG


                                                ;BLOCK FOR DRIVE 5

043614     005      000                 DRIVE5: .BYTE   5,0             ;DRIVE NUMBER
043616  000005                                  .BLKW   5
043630  044050                                  .WORD   .+$RPCS1-$REG
043632  000266                                  .BLKB   $RPEC2-$REG


                                                ;BLOCK FOR DRIVE 6

044120     006      000                 DRIVE6: .BYTE   6,0             ;DRIVE NUMBER
044122  000005                                  .BLKW   5
044134  044354                                  .WORD   .+$RPCS1-$REG
044136  000266                                  .BLKB   $RPEC2-$REG


                                                ;BLOCK FOR DRIVE 7

044424     007      000                 DRIVE7: .BYTE   7,0             ;DRIVE NUMBER
044426  000005                                  .BLKW   5
044440  044660                                  .WORD   .+$RPCS1-$REG
044442  000266                                  .BLKB   $RPEC2-$REG


                                        ;GENERAL PURPOSE DPB - USED BY 'READHD', 'RECALT', 'OFFSET',&'RTNCTR'

044730  000000  000000  177774          GENDPB: .WORD   0,0,-4,CYLDER
044736  054456
044740  000000  000000  044750                  .WORD   0,0,GENREG,0
044746  000000
044750  000024                          GENREG: .BLKW   24              ;REGISTER STORAGE IF ERROR

                                        ;;*************************************************************

                                        .SBTTL  ERROR MESSAGES

                                        ;;*************************************************************

045020  044122  030461  044440          EM1:    .ASCIZ  /RH11 INTERRUPT OCCURRED (RPAS = 0)/
045026  052116  051105  052522
045034  052120  047440  041503
045042  051125  042522  020104
045050  051050  040520  020123
045056  020075  024460     000

045063     125  042516  050130          EM2:    .ASCIZ  /UNEXPECTED ATTENTION OCCURRED/
```

```
 9051  045070  041505  042524  020104
 9052  045076  052101  042524  052116
 9053  045104  047511  020116  041517
 9054  045112  052503  051122  042105
 9055  045120     000
 9056
 9057  045121     115  051501  041123  EM3:    .ASCIZ  /MASSBUS PARITY ERROR (MCPE=1)/
 9058  045126  051525  050040  051101
 9059  045134  052111  020131  051105
 9060  045142  047522  020122  046450
 9061  045150  050103  036505  024461
 9062  045156     000
 9063
 9064  045157     115  051501  041123  EM4:    .ASCIZ  /MASSBUS PARITY ERROR (PAR=1)/
 9065  045164  051525  050040  051101
 9066  045172  052111  020131  051105
 9067  045200  047522  020122  050050
 9068  045206  051101  030475  000051
 9069
 9070  045214  042101  051104  051505  EM5:    .ASCIZ  /ADDRESS PLUG CHANGE BIT SET/
 9071  045222  020123  046120  043525
 9072  045230  041440  040510  043516
 9073  045236  020105  044502  020124
 9074  045244  042523  000124
 9075
 9076  045250  044122  030461  042040  EM6:    .ASCIZ  /RH11 DIDN'T RESPOND TO ADDRESSING/
 9077  045256  042111  023516  020124
 9078  045264  042522  050123  047117
 9079  045272  020104  047524  040440
 9080  045300  042104  042522  051523
 9081  045306  047111  000107
 9082
 9083  045312  047125  047503  051122  EM10:   .ASCIZ  /UNCORRECTABLE MASSBUS PARITY ERROR/
 9084  045320  041505  040524  046102
 9085  045326  020105  040515  051523
 9086  045334  052502  020123  040520
 9087  045342  044522  054524  042440
 9088  045350  051122  051117     000
 9089
 9090  045355     106  052101  046101  EM11:   .ASCIZ  /FATAL MASSBUS PARITY ERROR/
 9091  045362  046440  051501  041123
 9092  045370  051525  050040  051101
 9093  045376  052111  020131  051105
 9094  045404  047522  000122
 9095
 9096  045410  042520  051522  051511  EM12:   .ASCIZ  /PERSISTENT DEVICE UNSAFE/
 9097  045416  042524  052116  042040
 9098  045424  053105  041511  020105
 9099  045432  047125  040523  042506
 9100  045440     000
 9101
 9102  045441     117  042520  040522  EM13:   .ASCIZ  /OPERATION NOT COMPLETED WITHIN TIME LIMIT/
 9103  045446  044524  047117  047040
 9104  045454  052117  041440  046517
 9105  045462  046120  052105  042105
 9106  045470  053440  052111  044510
```

```
9107  045476  020116  044524  042515
9108  045504  046040  046511  052111
9109  045512     000
9110
9111  045513     104  044522  042526  EM14:   .ASCIZ  /DRIVE WENT OFFLINE/
9112  045520  053440  047105  020124
9113  045526  043117  046106  047111
9114  045534  000105
9115
9116  045536  047516  051040  051505  EM15:   .ASCIZ  /NO RESPONSE TO PORT REQUEST/
9117  045544  047520  051516  020105
9118  045552  047524  050040  051117
9119  045560  020124  042522  052521
9120  045566  051505  000124
9121
9122  045572  042510  042101  051105  EM20:   .ASCIZ  /HEADER CRC ERROR/
9123  045600  041440  041522  042440
9124  045606  051122  051117     000
9125
9126  045613     104  052101  020101  EM21:   .ASCIZ  /DATA CHECK ('DCK') ERROR/
9127  045620  044103  041505  020113
9128  045626  023450  041504  023513
9129  045634  020051  051105  047522
9130  045642  000122
9131
9132  045644  051127  052111  020105  EM22:   .ASCIZ  /WRITE CHECK ERROR - DATA CHECK ('DCK') SET/
9133  045652  044103  041505  020113
9134  045660  051105  047522  020122
9135  045666  020055  040504  040524
9136  045674  041440  042510  045503
9137  045702  024040  042047  045503
9138  045710  024447  051440  052105
9139  045716     000
9140
9141  045717     127  044522  042524  EM23:   .ASCIZ  /WRITE CHECK ERROR - DATA CHECK ('DCK') NOT SET/
9142  045724  041440  042510  045503
9143  045732  042440  051122  051117
9144  045740  026440  042040  052101
9145  045746  020101  044103  041505
9146  045754  020113  023450  041504
9147  045762  023513  020051  047516
9148  045770  020124  042523  000124
9149
9150  045776  042510  042101  051105  EM24:   .ASCIZ  /HEADER READ ERROR - 'FMT' BIT DROPPED/
9151  046004  051040  040505  020104
9152  046012  051105  047522  020122
9153  046020  020055  043047  052115
9154  046026  020047  044502  020124
9155  046034  051104  050117  042520
9156  046042  000104
9157
9158  046044  042510  042101  051105  EM25:   .ASCIZ  /HEADER READ ERROR - HEADER COMPARE ('HCE') ERROR/
9159  046052  051040  040505  020104
9160  046060  051105  047522  020122
9161  046066  020055  042510  042101
9162  046074  051105  041440  046517
```

```
9163  046102  040520  042522  024040
9164  046110  044047  042503  024447
9165  046116  042440  051122  051117
9166  046124     000
9167
9168  046125     106  051117  040515  EM26:   .ASCIZ  /FORMAT ERROR ('FER')/
9169  046132  020124  051105  047522
9170  046140  020122  023450  042506
9171  046146  023522  000051
9172
9173  046152  042510  042101  051105  EM27:   .ASCIZ  /HEADER COMPARE ('HCE') ERROR/
9174  046160  041440  046517  040520
9175  046166  042522  024040  044047
9176  046174  042503  024447  042440
9177  046202  051122  051117     000
9178
9179  046207     115  051511  042503  EM30:   .ASCIZ  /MISCELLANEOUS DRIVE ERROR/
9180  046214  046114  047101  047505
9181  046222  051525  042040  044522
9182  046230  042526  042440  051122
9183  046236  051117     000
9184
9185  046241     117  042520  040522  EM31:   .ASCIZ  /OPERATION INCOMPLETE ('OPI') ERROR/
9186  046246  044524  047117  044440
9187  046254  041516  046517  046120
9188  046262  052105  020105  023450
9189  046270  050117  023511  020051
9190  046276  051105  047522  000122
9191
9192  046304  051104  053111  020105  EM32:   .ASCIZ  /DRIVE TIMING ('DTE') ERROR/
9193  046312  044524  044515  043516
9194  046320  024040  042047  042524
9195  046326  024447  042440  051122
9196  046334  051117     000
9197
9198  046337     120  051101  052111  EM33:   .ASCIZ  /PARITY ('PAR') ERROR AFTER OPERATION STARTED/
9199  046344  020131  023450  040520
9200  046352  023522  020051  051105
9201  046360  047522  020122  043101
9202  046366  042524  020122  050117
9203  046374  051105  052101  047511
9204  046402  020116  052123  051101
9205  046410  042524  000104
9206
9207  046414  051127  052111  020105  EM34:   .ASCIZ  /WRITE CLOCK FAILURE ('WCF') ERROR/
9208  046422  046103  041517  020113
9209  046430  040506  046111  051125
9210  046436  020105  023450  041527
9211  046444  023506  020051  051105
9212  046452  047522  000122
9213
9214  046456  047111  040526  044514  EM35:   .ASCIZ  /INVALID ADDRESS ('IAE') ERROR/
9215  046464  020104  042101  051104
9216  046472  051505  020123  023450
9217  046500  040511  023505  020051
9218  046506  051105  047522  000122
```

```
9219
9220  046514  051127  052111  020105  EM36:   .ASCIZ  /WRITE LOCK ('WLE') ERROR/
9221  046522  047514  045503  024040
9222  046530  053447  042514  024447
9223  046536  042440  051122  051117
9224  046544     000
9225
9226  046545     104  052101  020101  EM37:   .ASCIZ  /DATA CHECK ('DCK') SET DURING WRITE CHECK COMMAND/
9227  046552  044103  041505  020113
9228  046560  023450  041504  023513
9229  046566  020051  042523  020124
9230  046574  052504  044522  043516
9231  046602  053440  044522  042524
9232  046610  041440  042510  045503
9233  046616  041440  046517  040515
9234  046624  042116     000
9235
9236  046627     122  030510  020061  EM40:   .ASCIZ  /RH11 OR UNIBUS TRANSFER ERROR/
9237  046634  051117  052440  044516
9238  046642  052502  020123  051124
9239  046650  047101  043123  051105
9240  046656  042440  051122  051117
9241  046664     000
9242
9243  046665     102  051525  040440  EM41:   .ASCIZ  /BUS ADDRESS OR WORD COUNT INCORRECT/
9244  046672  042104  042522  051523
9245  046700  047440  020122  047527
9246  046706  042122  041440  052517
9247  046714  052116  044440  041516
9248  046722  051117  042522  052103
9249  046730     000
9250
9251  046731     104  052101  020101  EM42:   .ASCIZ  /DATA COMPARE ERRORS - NO OTHER ERROR(S) DETECTED/
9252  046736  047503  050115  051101
9253  046744  020105  051105  047522
9254  046752  051522  026440  047040
9255  046760  020117  052117  042510
9256  046766  020122  051105  047522
9257  046774  024122  024523  042040
9258  047002  052105  041505  042524
9259  047010  000104
9260
9261  047012  040503  023516  020124  EM43:   .ASCIZ  /CAN'T MATCH DATA READ WITH A PATTERN/
9262  047020  040515  041524  020110
9263  047026  040504  040524  051040
9264  047034  040505  020104  044527
9265  047042  044124  040440  050040
9266  047050  052101  042524  047122
9267  047056     000
9268
9269  047057     105  051122  051117  EM44:   .ASCIZ  /ERROR BIT(S) SET, BUT NO ERROR SIGNALED BY THE RH11/
9270  047064  041040  052111  051450
9271  047072  020051  042523  026124
9272  047100  041040  052125  047040
9273  047106  020117  051105  047522
9274  047114  020122  044523  047107
```

```
9275  047122  046101  042105  041040
9276  047130  020131  044124  020105
9277  047136  044122  030461     000
9278
9279  047143     105  041503  046040  EM45:   .ASCIZ  /ECC LOGIC FAILURE - POSITION REGISTER VALUE TOO LARGE/
9280  047150  043517  041511  043040
9281  047156  044501  052514  042522
9282  047164  026440  050040  051517
9283  047172  052111  047511  020116
9284  047200  042522  044507  052123
9285  047206  051105  053040  046101
9286  047214  042525  052040  047517
9287  047222  046040  051101  042507
9288  047230     000
9289
9290  047231     102  051525  040440  EM46:   .ASCIZ  /BUS ADDRESS AND WORD COUNT NOT CONSISTENT/
9291  047236  042104  042522  051523
9292  047244  040440  042116  053440
9293  047252  051117  020104  047503
9294  047260  047125  020124  047516
9295  047266  020124  047503  051516
9296  047274  051511  042524  052116
9297  047302     000
9298
9299  047303     123  042505  020113  EM50:   .ASCIZ  /SEEK INCOMPLETE ('SKI') OR OFF CYLINDER ('OCYL') ERROR/
9300  047310  047111  047503  050115
9301  047316  042514  042524  024040
9302  047324  051447  044513  024447
9303  047332  047440  020122  043117
9304  047340  020106  054503  044514
9305  047346  042116  051105  024040
9306  047354  047447  054503  023514
9307  047362  020051  051105  047522
9308  047370  000122
9309
9310  047372  051120  043517  040522  EM51:   .ASCIZ  /PROGRAM DETECTED POSITIONING ERROR/
9311  047400  020115  042504  042524
9312  047406  052103  042105  050040
9313  047414  051517  052111  047511
9314  047422  044516  043516  042440
9315  047430  051122  051117     000
9316
9317  047435     104  044522  042526  EM60:   .ASCIZ  /DRIVE UNSAFE ERROR/
9318  047442  052440  051516  043101
9319  047450  020105  051105  047522
9320  047456  000122
9321
9322  047460  050122  051501     000  DH1:    .ASCIZ  /RPAS/
9323
9324  047465     104  044522  042526  DH2:    .ASCIZ  /DRIVE   RPDS1   RPER1   RPER2   RPER3   RPAS/
9325  047472  020040  051040  042120
9326  047500  030523  020040  051040
9327  047506  042520  030522  020040
9328  047514  051040  042520  031122
9329  047522  020040  051040  042520
9330  047530  031522  020040  051040
```

```
 9331  047536  040520  000123
 9332
 9333  047542  051104  053111  020105  DH3:    .ASCIZ  /DRIVE   REG ADR  DATA/
 9334  047550  020040  042522  020107
 9335  047556  042101  020122  042040
 9336  047564  052101  000101
 9337
 9338  047570  051104  053111  020105  DH4:    .ASCIZ  /DRIVE   REG ADR    GOOD    BAD/
 9339  047576  020040  042522  020107
 9340  047604  042101  020122  020040
 9341  047612  043440  047517  020104
 9342  047620  020040  041040  042101
 9343  047626     000
 9344
 9345  047627     044  050122  042101  DH6:    .ASCIZ  /$RPADR/
 9346  047634  000122
 9347
 9348  047636  051104  020126  050122  DH14:   .ASCII  /DRV RPCS1   RPCS2   RPDS1   RPER1   /
 9349  047644  051503  020061  020040
 9350  047652  050122  051503  020062
 9351  047660  020040  050122  051504
 9352  047666  020061  020040  050122
 9353  047674  051105  020061  020040
 9354  047702  050122  051105  020062          .ASCIZ  /RPER2   RPER3   RPEC1   RPEC2/<CR><LF>
 9355  047710  020040  050122  051105
 9356  047716  020063  020040  050122
 9357  047724  041505  020061  020040
 9358  047732  050122  041505  006462
 9359  047740  000012
 9360
 9361  047742  050122  041527  020040  DH15:   .ASCII  /RPWC    RPBA    RPDA    RPAS    RPLA    /
 9362  047750  020040  050122  040502
 9363  047756  020040  020040  050122
 9364  047764  040504  020040  020040
 9365  047772  050122  051501  020040
 9366  050000  020040  050122  040514
 9367  050006  020040  020040
 9368  050012  050122  041104  020040          .ASCIZ  /RPDB    RPMR    RPDT/<CR><LF>
 9369  050020  020040  050122  051115
 9370  050026  020040  020040  050122
 9371  050034  052104  005015     000
 9372
 9373  050041     122  051520  020116  DH16:   .ASCIZ  /RPSN    RPOF    RPCA    RPCC    STATUS/<CR><LF>
 9374  050046  020040  051040  047520
 9375  050054  020106  020040  051040
 9376  050062  041520  020101  020040
 9377  050070  051040  041520  020103
 9378  050076  020040  051440  040524
 9379  050104  052524  006523  000012
 9380
 9381                                          .EVEN
 9382
 9383  050112  001244  000000          DT1:    .WORD   ATTN,0
 9384
 9385  050116  001242  033164  033166  DT2:    .WORD   DRIVE,RPERRS,RPERRS+2,RPERRS+4,RPERRS+6,ATTN,0
 9386  050124  033170  033172  001244
```

```
 9387  050132  000000
 9388
 9389  050134  001242  040432  040434  DT3:    .WORD   DRIVE,RD.ADR,RD.WRD,0
 9390  050142  000000
 9391
 9392  050144  001242  040652  040650  DT4:    .WORD   DRIVE,WRT.AD,WRT.WD,RD.WRD,0
 9393  050152  040434  000000
 9394
 9395  050156  001170  000000          DT6:    .WORD   $RPADR,0
 9396
 9397  050162  000234  000244  000246  DT14:   .WORD   $RPCS1,$RPCS2,$RPDS1,$RPER1,$RPER2,$RPER3,$RPEC1,$RPEC2,0
 9398  050170  000250  000274  000276
 9399  050176  000300  000302  000000
 9400
 9401  050204  000236  000240  000242  DT15:   .WORD   $RPWC,$RPBA,$RPDA,$RPAS,$RPLA,$RPDB,$RPMR,$RPDT,0
 9402  050212  000252  000254  000256
 9403  050220  000260  000262  000000
 9404
 9405  050226  000264  000266  000270  DT16:   .WORD   $RPSN,$RPOF,$RPCA,$RPCC,$TATUS,0
 9406  050234  000272  000016  000000
 9407
 9408  050242  051120  051505  047105  LIN2C:  .ASCIZ  /PRESENT ORDER = /
 9409  050250  020124  051117  042504
 9410  050256  020122  020075     000
 9411  050263     040  050040  042522  LIN2P:  .ASCIZ  /  PREVIOUS ORDER = /
 9412  050270  044526  052517  020123
 9413  050276  051117  042504  020122
 9414  050304  020075     000
 9415  050307     052  042440  051122  LIN2S:  .ASCIZ  @* ERROR AT BAD TRACK/SECTOR@
 9416  050314  051117  040440  020124
 9417  050322  040502  020104  051124
 9418  050330  041501  027513  042523
 9419  050336  052103  051117     000
 9420  050343     105  051122  051117  LINM3:  .ASCIZ  /ERROR AT C/
 9421  050350  040440  020124  000103
 9422  050356  052040     000          T:      .ASCIZ  / T/
 9423  050361     120  042522  042523  LINN3:  .ASCIZ  /PRESENT ADDR = C/
 9424  050366  052116  040440  042104
 9425  050374  020122  020075  000103
 9426  050402  051440     000          S:      .ASCIZ  / S/
 9427  050405     040  020040  051120  LINP3:  .ASCIZ  /   PREV ADDR = C/
 9428  050412  053105  040440  042104
 9429  050420  020122  020075  000103
 9430  050426  052123  051101  020124  LINS3:  .ASCIZ  /START CYL = /
 9431  050434  054503  020114  020075
 9432  050442     000
 9433  050443     040  042440  042116  LINEN3: .ASCIZ  /  END CYL = /
 9434  050450  041440  046131  036440
 9435  050456  000040
 9436  050460  020040  041501  052524  LINA3:  .ASCIZ  /  ACTUAL CYL = /
 9437  050466  046101  041440  046131
 9438  050474  036440  000040
 9439  050500  020040  051124  020113  LINT3:  .ASCIZ  /  TRK = /
 9440  050506  020075     000
 9441  050511     040  050122  040503  LINCA3: .ASCIZ  / RPCA = /
 9442  050516  036440  000040
```

```
 9443  050522  050122  040504  036440  LINDA3: .ASCIZ  /RPDA = /
 9444  050530  000040
 9445  050532  050122  040502  036440  LINB3:  .ASCIZ  /RPBA = /
 9446  050540  000040
 9447  050542  020040  050122  041527  LINW3:  .ASCIZ  /  RPWC = /
 9448  050550  036440  000040
 9449  050554  052123  051101  020124  LINST3: .ASCIZ  /START TRK = /
 9450  050562  051124  020113  020075
 9451  050570     000
 9452  050571     123  040524  052122  LINSS3: .ASCIZ  /START SEC = /
 9453  050576  051440  041505  036440
 9454  050604  000040
 9455  050606  052502  043106  051105  LINM4:  .ASCIZ  /BUFFER ADDR = /
 9456  050614  040440  042104  020122
 9457  050622  020075     000
 9458  050625     040  051440  055111  LINS4:  .ASCIZ  /  SIZE = /
 9459  050632  020105  020075     000
 9460  050637     040  040440  052103  LINX4:  .ASCIZ  /  ACTUAL NMBR WRDS XFRD = /
 9461  050644  040525  020114  046516
 9462  050652  051102  053440  042122
 9463  050660  020123  043130  042122
 9464  050666  036440  000040
 9465  050672  047507  042117  042040  LIND5:  .ASCIZ  /GOOD DATA = /
 9466  050700  052101  020101  020075
 9467  050706     000
 9468  050707     040  041040  042101  LINB5:  .ASCIZ  /  BAD DATA = /
 9469  050714  042040  052101  020101
 9470  050722  020075     000
 9471  050725     040  051440  041505  LINP5:  .ASCIZ  /  SECT POS = /
 9472  050732  020124  047520  020123
 9473  050740  020075     000
 9474  050743     110  040505  042504  LINS5:  .ASCIZ  /HEADER FROM ERROR SECTOR = /
 9475  050750  020122  051106  046517
 9476  050756  042440  051122  051117
 9477  050764  051440  041505  047524
 9478  050772  020122  020075     000
 9479  050777     122  042520  030503  LINEP5: .ASCIZ  /RPEC1 = /
 9480  051004  036440  000040
 9481  051010  051040  042520  031103  LINE05: .ASCIZ  / RPEC2 = /
 9482  051016  036440  000040
 9483  051022  042523  052103  051117  LINB6:  .ASCIZ  /SECTOR IS ECC CORRECTABLE /
 9484  051030  044440  020123  041505
 9485  051036  020103  047503  051122
 9486  051044  041505  040524  046102
 9487  051052  020105     000
 9488  051055     123  041505  047524  LINC6:  .ASCIZ  /SECTOR READ CORRECTLY /
 9489  051062  020122  042522  042101
 9490  051070  041440  051117  042522
 9491  051076  052103  054514  000040
 9492  051104  047503  051122  041505  LING6:  .ASCIZ  /CORRECTED ON /
 9493  051112  042524  020104  047117
 9494  051120  000040
 9495  051122  051040  052105  044522  LINR6:  .ASCIZ  / RETRIES/
 9496  051130  051505     000
 9497  051133     125  041516  051117  LINU06: .ASCIZ  /UNCORRECTABLE AFTER /
 9498  051140  042522  052103  041101
```

```
 9499  051146  042514  040440  052106
 9500  051154  051105  000040
 9501  051160  020040  047524  040524  LIN7M:  .ASCIZ  /  TOTAL MISPOS ERR = /
 9502  051166  020114  044515  050123
 9503  051174  051517  042440  051122
 9504  051202  036440  000040
 9505  051206  051117  042504  051522  LIN7O:  .ASCIZ  /ORDERS:/
 9506  051214  000072
 9507  051216  052040  052117  046101  LIN7P:  .ASCIZ  / TOTAL SEEKS = /
 9508  051224  051440  042505  051513
 9509  051232  036440  000040
 9510  051236  052040  052117  046101  LIN7S:  .ASCIZ  / TOTAL SKI,OCYL ERR = /
 9511  051244  051440  044513  047454
 9512  051252  054503  020114  051105
 9513  051260  020122  020075     000
 9514  051265     040  042440  051122  LIN7T:  .ASCIZ  /  ERRORS:/
 9515  051272  051117  035123     000
 9516  051277     040  053440  042122  LIN7X:  .ASCIZ  /  WRDS XFR:/
 9517  051304  020123  043130  035122
 9518  051312     000
 9519  051313     040  053440  042122  LIN7R:  .ASCIZ  /  WRDS READ:/
 9520  051320  020123  042522  042101
 9521  051326  000072
 9522  051330  044504  043106  051105  LIN8M:  .ASCIZ  /DIFFERENT ERROR DURING RETRY/
 9523  051336  047105  020124  051105
 9524  051344  047522  020122  052504
 9525  051352  044522  043516  051040
 9526  051360  052105  054522     000
 9527  051365     104  052101  020101  LIN9B:  .ASCIZ  /DATA COMPARISON ERRORS/
 9528  051372  047503  050115  051101
 9529  051400  051511  047117  042440
 9530  051406  051122  051117  000123
 9531  051414  020040  020040  020040  LIN9H:  .ASCII  /          GOOD    BAD/<CR><LF>
 9532  051422  020040  043440  047517
 9533  051430  020104  020040  041040
 9534  051436  042101  005015
 9535  051442  047514  020103  020040          .ASCIZ  /LOC       DATA    DATA/<CR><LF>
 9536  051450  020040  042040  052101
 9537  051456  020101  020040  042040
 9538  051464  052101  006501  000012
 9539  051472  047514  020103  020040  LIN9I:  .ASCIZ  /LOC       DATA/<CR><LF>
 9540  051500  020040  042040  052101
 9541  051506  006501  000012
 9542  051512  047524  040524  020114  LIN9E:  .ASCIZ  /TOTAL COMPARE ERRORS = /
 9543  051520  047503  050115  051101
 9544  051526  020105  051105  047522
 9545  051534  051522  036440  000040
 9546  051542  044124  020105  040504  LIN9G:  .ASCIZ  /THE DATA COMPARED OK/<CR><LF>
 9547  051550  040524  041440  046517
 9548  051556  040520  042522  020104
 9549  051564  045517  005015     000
 9550  051571     105  051122  051117  LIN10A: .ASCIZ  /ERROR BURST BEGINS AT WORD /
 9551  051576  041040  051125  052123
 9552  051604  041040  043505  047111
 9553  051612  020123  052101  053440
 9554  051620  051117  020104     000
```

```
 9555  051625     040  047111  042040  LIN10B: .ASCIZ  / IN DATA FIELD OF ERROR SECTOR/<CR><LF>
 9556  051632  052101  020101  044506
 9557  051640  046105  020104  043117
 9558  051646  042440  051122  051117
 9559  051654  051440  041505  047524
 9560  051662  006522  000012
 9561  051666  051105  047522  020122  LIN10C: .ASCII  /ERROR WAS NOT IN THE DATA READ - /<CR><LF>
 9562  051674  040527  020123  047516
 9563  051702  020124  047111  052040
 9564  051710  042510  042040  052101
 9565  051716  020101  042522  042101
 9566  051724  026440  006440     012
 9567  051731     105  041503  041440          .ASCIZ  /ECC CORRECTION CAN'T BE PERFORMED/
 9568  051736  051117  042522  052103
 9569  051744  047511  020116  040503
 9570  051752  023516  020124  042502
 9571  051760  050040  051105  047506
 9572  051766  046522  042105     000
 9573  051773     105  041503  041440  LIN10H: .ASCII  /ECC CORRECTION RESULTS/<CR><LF>
 9574  052000  051117  042522  052103
 9575  052006  047511  020116  042522
 9576  052014  052523  052114  006523
 9577  052022     012
 9578  052023     101  042104  020122          .ASCIZ  /ADDR    BAD     CORRECTED /<CR><LF>
 9579  052030  020040  041040  042101
 9580  052036  020040  020040  041440
 9581  052044  051117  042522  052103
 9582  052052  042105  006440  000012
 9583  052060  047503  052116  047105  LIN11H: .ASCIZ  /CONTENTS OF ERROR SECTOR (REPORTED ABOVE)/<CR><LF>
 9584  052066  051524  047440  020106
 9585  052074  051105  047522  020122
 9586  052102  042523  052103  051117
 9587  052110  024040  042522  047520
 9588  052116  052122  042105  040440
 9589  052124  047502  042526  006451
 9590  052132  000012
 9591  052134  042101  051104  020040          .ASCIZ  /ADDR    DATA/<CR><LF>
 9592  052142  020040  040504  040524
 9593  052150  005015     000
 9594  052153     040     040          LIN4SP: .ASCII  /  /
 9595  052155     040                  LINSP:  .ASCII  / /
 9596  052156  000040                  LINSP0: .ASCIZ  / /
 9597
 9598                                  .SBTTL  TELETYPE MESSAGES
 9599
 9600  052160  051104  053111  000105  UNTMSG: .ASCIZ  /DRIVE/
 9601  052166  047440  043106  044514  UNTOFF: .ASCIZ  / OFFLINE/
 9602  052174  042516     000
 9603  052177     040  047117  044514  UNTON:  .ASCIZ  / ONLINE/
 9604  052204  042516     000
 9605  052207     040  047516  020124  UNTNOT: .ASCIZ  / NOT BEING TESTED/
 9606  052214  042502  047111  020107
 9607  052222  042524  052123  042105
 9608  052230     000
 9609  052231     040  046101  042522  UNTASN: .ASCIZ  / ALREADY BEING TESTED/
 9610  052236  042101  020131  042502
```

```
9611   052244   047111   020107   042524
9612   052252   052123   042105      000
9613   052257      040   047516   020124   NOTRP:   .ASCIZ  @ NOT AN RP04/5/6@
9614   052264   047101   051040   030120
9615   052272   027464   027465   000066
9616   052300   047040   052117   050040   NOTPRS:  .ASCIZ  / NOT PRESENT/
9617   052306   042522   042523   052116
9618   052314      000
9619   052315      040   047516   020124   NOTAVL:  .ASCIZ  / NOT AVAILABLE/
9620   052322   053101   044501   040514
9621   052330   046102   000105
9622   052334   052440   051516   043101   NOTSAF:  .ASCIZ  / UNSAFE/
9623   052342   000105
9624   052344   047125   052111   051440   SYSTAT:  .ASCIZ  /UNIT STATUS:/<CR><LF><LF>
9625   052352   040524   052524   035123
9626   052360   005015   000012
9627   052364   050122   032060      000   RP04B:   .ASCIZ  /RP04/
9628   052371      122   030120   000065   RP05:    .ASCIZ  /RP05/
9629   052376   050122   033060      000   RP06:    .ASCIZ  /RP06/
9630   052403      104   044522   042526   STATHD:  .ASCII  /DRIVE PERFORMANCE SUMMARY/<CR><LF>
9631   052410   050040   051105   047506
9632   052416   046522   047101   042503
9633   052424   051440   046525   040515
9634   052432   054522   005015
9635   052436   051104   020126   040520            .ASCII  /DRV PASS ORDERS   SEEKS   WRDS XFER   WRDS READ /
9636   052444   051523   047440   042122
9637   052452   051105   020123   020040
9638   052460   042523   045505   020123
9639   052466   020040   051127   051504
9640   052474   054040   042506   020122
9641   052502   020040   051127   051504
9642   052510   051040   040505   020104
9643   052516   047523   052106   044040            .ASCIZ  /SOFT HARD  SKI MISP OTHER/<CR><LF>
9644   052524   051101   020104   051440
9645   052532   044513   046440   051511
9646   052540   020120   052117   042510
9647   052546   006522   000012
9648   052552   047504   042516   005015   PDONE:   .ASCIZ  /DONE/<CR><LF><LF>
9649   052560   000012
9650   052562   037407   040506   040524   DROPNG:  .ASCIZ  <07>/?FATAL OR EXCESSIVE ERRORS/
9651   052570   020114   051117   042440
9652   052576   041530   051505   044523
9653   052604   042526   042440   051122
9654   052612   051117   000123
9655   052616   047105   020104   043117   ENDPAS:  .ASCIZ  /END OF PASS/
9656   052624   050040   051501   000123
9657   052632   005015   047105   020104   ENDTST:  .ASCIZ  <CR><LF>/END OF TEST/
9658   052640   043117   052040   051505
9659   052646   000124
9660   052650   005015   051104   053111   DEASSG:  .ASCIZ  <CR><LF>/DRIVE DEASSIGNED/
9661   052656   020105   042504   051501
9662   052664   044523   047107   042105
9663   052672      000
9664   052673      015   025012   025052   DRNUM:   .ASCIZ  <CR><LF>/********** DRIVE #/
9665   052700   025052   025052   025052
9666   052706   020052   051104   053111
```

```
 9667  052714  020105  000043
 9668  052720  051440  040524  052122  ASGND:   .ASCIZ  / STARTED/<CR><LF>
 9669  052726  042105  005015     000
 9670  052733     015  037412  023440  NEDCLK:  .ASCIZ  <CR><LF>/? 'L' OR 'P' CLOCK REQUIRED ON SYSTEM/<CR><LF>
 9671  052740  023514  047440  020122
 9672  052746  050047  020047  046103
 9673  052754  041517  020113  042522
 9674  052762  052521  051111  042105
 9675  052770  047440  020116  054523
 9676  052776  052123  046505  005015
 9677  053004     000
 9678  053005     056     000          PERIOD:  .ASCIZ  /./
 9679  053007     077     000          QUES:    .ASCIZ  /?/
 9680  053011     111  053116  046101  INVLD:   .ASCIZ  /INVALID COMMAND/<CR><LF>
 9681  053016  042111  041440  046517
 9682  053024  040515  042116  005015
 9683  053032     000
 9684  053033     015  042412  052116  ENTCOM:  .ASCIZ  <CR><LF>/ENTER COMMANDS: /<CR><LF>
 9685  053040  051105  041440  046517
 9686  053046  040515  042116  035123
 9687  053054  006440  000012
 9688  053060  047105  042524  020122  ENTDRV:  .ASCIZ  /ENTER I.D. FOR DRV #/
 9689  053066  027111  027104  043040
 9690  053074  051117  042040  053122
 9691  053102  021440     000
 9692  053105     015  042412  052116  ENTLMT:  .ASCIZ  <CR><LF>/ENTER ADDRESS LIMITS FOR DRV #/
 9693  053112  051105  040440  042104
 9694  053120  042522  051523  046040
 9695  053126  046511  052111  020123
 9696  053134  047506  020122  051104
 9697  053142  020126  000043
 9698  053146  047105  042524  020122  ENTADR:  .ASCIZ  @ENTER BAD TRK/SEC ADRS FOR DRV #@
 9699  053154  040502  020104  051124
 9700  053162  027513  042523  020103
 9701  053170  042101  051522  043040
 9702  053176  051117  042040  053122
 9703  053204  021440     000
 9704  053207     072     000          COLON:   .ASCIZ  /:/
 9705  053211     015  042012  052101  DATEIS:  .ASCIZ  <CR><LF>/DATE: /
 9706  053216  035105  000040
 9707  053222  005015  050117  051105  IDIS:    .ASCIZ  <CR><LF>/OPERATOR I.D.: /
 9708  053230  052101  051117  044440
 9709  053236  042056  035056  000040
 9710  053244  005015  042012  053122  HEDLIN:  .ASCIZ  <CR><LF><LF>/DRV  DRV I.D./<CR><LF>
 9711  053252  020040  051104  020126
 9712  053260  027111  027104  005015
 9713  053266     000
 9714  053267     116  047117  006505  NONE:    .ASCIZ  /NONE/<CR><LF>
 9715  053274  000012
 9716  053276  020077  047111  040526  BADENT:  .ASCIZ  /? INVALID ENTRY/<CR><LF>
 9717  053304  044514  020104  047105
 9718  053312  051124  006531  000012
 9719  053320  054523  052123  046505  BUSY:    .ASCIZ  /SYSTEM BUSY.../<CR><LF>
 9720  053326  041040  051525  027131
 9721  053334  027056  005015     000
 9722  053341     015  050012  047522  INTDON:  .ASCII  <CR><LF>/PROGRAM INITIALIZATION COMPLETE/
```

```
 9723  053346  051107  046501  044440
 9724  053354  044516  044524  046101
 9725  053362  055111  052101  047511
 9726  053370  020116  047503  050115
 9727  053376  042514  042524
 9728  053402  005015  054524  042520          .ASCIZ  <CR><LF>/TYPE A 'CONTROL C' TO ENTER COMMANDS/<CR><LF><LF>
 9729  053410  040440  023440  047503
 9730  053416  052116  047522  020114
 9731  053424  023503  052040  020117
 9732  053432  047105  042524  020122
 9733  053440  047503  046515  047101
 9734  053446  051504  005015  000012
 9735
 9736                                  .EVEN
 9737
 9738                                  ;PARAMETER ENTRY TABLE
 9739
 9740  053454  053602  000000  001404  PARLST: .WORD   PAR1,0,MAXDL
 9741  053462  053610  177777  001410          .WORD   PAR2,-1,INTRVL
 9742  053470  053740  177777  001402          .WORD   PAR19,-1,PASCNT
 9743  053476  053617  177777  001414          .WORD   PAR3,-1,CMPLMT
 9744  053504  053707  000001  001420          .WORD   PAR11,1,WCSEL
 9745  053512  053715  000007  001422          .WORD   PAR14,7,RATIO
 9746  053520  053732  000001  001430          .WORD   PAR16,1,ENDET
 9747  053526  053700  000001  001416          .WORD   PAR10,1,FORMAT
 9748  053534  053723  000001  001424          .WORD   PAR15,1,AUTOCK
 9749  053542  053747  000001  001426          .WORD   PAR20,1,NOTPRT
 9750  053550  000000                          .WORD   0               ;TABLE TERMINATOR
 9751
 9752  053552  047105  042524  020122  ASKPAR: .ASCIZ  /ENTER PARAMETERS:  /
 9753  053560  040520  040522  042515
 9754  053566  042524  051522  020072
 9755  053574  000040
 9756  053576  027440  000040          SLASH:  .ASCIZ  @ / @
 9757
 9758  053602  040515  042130  000114  PAR1:   .ASCIZ  /MAXDL/
 9759  053610  047111  051124  046126  PAR2:   .ASCIZ  /INTRVL/
 9760  053616     000
 9761  053617     103  050115  046514  PAR3:   .ASCIZ  /CMPLMT/
 9762  053624  000124
 9763  053626  040515  041530  046131  PAR4:   .ASCIZ  /MAXCYL/
 9764  053634     000
 9765  053635     115  047111  054503  PAR5:   .ASCIZ  /MINCYL/
 9766  053642  000114
 9767  053644  040515  052130  045522  PAR6:   .ASCIZ  /MAXTRK/
 9768  053652     000
 9769  053653     115  047111  051124  PAR7:   .ASCIZ  /MINTRK/
 9770  053660  000113
 9771  053662  040515  051530  041505  PAR8:   .ASCIZ  /MAXSEC/
 9772  053670     000
 9773  053671     115  047111  042523  PAR9:   .ASCIZ  /MINSEC/
 9774  053676  000103
 9775  053700  047506  046522  052101  PAR10:  .ASCIZ  /FORMAT/
 9776  053706     000
 9777  053707     127  051503  046105  PAR11:  .ASCIZ  /WCSEL/
 9778  053714     000
```

```
 9779  053715      122  052101  047511  PAR14:  .ASCIZ  /RATIO/
 9780  053722      000
 9781  053723      101  052125  041517  PAR15:  .ASCIZ  /AUTOCK/
 9782  053730  000113
 9783  053732  047105  042504  000124  PAR16:  .ASCIZ  /ENDET/
 9784  053740  040520  041523  052116  PAR19:  .ASCIZ  /PASCNT/
 9785  053746      000
 9786  053747      116  052117  051120  PAR20:  .ASCIZ  /NOTPRT/
 9787  053754  000124
 9788
 9789                                          .EVEN
 9790
 9791                                  ;PARAMETER TABLE POINTERS FOR ADDRESS LIMITS
 9792
 9793  053756  053776                  TABLE:  .WORD   TABLE0           ;PARAMETER TABLE FOR DRIVE 0
 9794  053760  054044                          .WORD   TABLE1           ;PARAMETER TABLE FOR DRIVE 1
 9795  053762  054112                          .WORD   TABLE2           ;PARAMETER TABLE FOR DRIVE 2
 9796  053764  054160                          .WORD   TABLE3           ;PARAMETER TABLE FOR DRIVE 3
 9797  053766  054226                          .WORD   TABLE4           ;PARAMETER TABLE FOR DRIVE 4
 9798  053770  054274                          .WORD   TABLE5           ;PARAMETER TABLE FOR DRIVE 5
 9799  053772  054342                          .WORD   TABLE6           ;PARAMETER TABLE FOR DRIVE 6
 9800  053774  054410                          .WORD   TABLE7           ;PARAMETER TABLE FOR DRIVE 7
 9801
 9802                                  ;PARAMETER TABLE FOR ADDRESS LIMITS
 9803
 9804  053776  053635  000000  042000  TABLE0: .WORD   PAR5,0,MINCYL+DRIVE0
 9805  054004  053626  000000  041776          .WORD   PAR4,0,MAXCYL+DRIVE0
 9806  054012  053653  000022  042004          .WORD   PAR7,18.,MINTRK+DRIVE0
 9807  054020  053644  000022  042002          .WORD   PAR6,18.,MAXTRK+DRIVE0
 9808  054026  053671  000025  042010          .WORD   PAR9,21.,MINSEC+DRIVE0
 9809  054034  053662  000025  042006          .WORD   PAR8,21.,MAXSEC+DRIVE0,0
 9810  054042  000000
 9811
 9812  054044  053635  000000  042304  TABLE1: .WORD   PAR5,0,MINCYL+DRIVE1
 9813  054052  053626  000000  042302          .WORD   PAR4,0,MAXCYL+DRIVE1
 9814  054060  053653  000022  042310          .WORD   PAR7,18.,MINTRK+DRIVE1
 9815  054066  053644  000022  042306          .WORD   PAR6,18.,MAXTRK+DRIVE1
 9816  054074  053671  000025  042314          .WORD   PAR9,21.,MINSEC+DRIVE1
 9817  054102  053662  000025  042312          .WORD   PAR8,21.,MAXSEC+DRIVE1,0
 9818  054110  000000
 9819
 9820  054112  053635  000000  042610  TABLE2: .WORD   PAR5,0,MINCYL+DRIVE2
 9821  054120  053626  000000  042606          .WORD   PAR4,0,MAXCYL+DRIVE2
 9822  054126  053653  000022  042614          .WORD   PAR7,18.,MINTRK+DRIVE2
 9823  054134  053644  000022  042612          .WORD   PAR6,18.,MAXTRK+DRIVE2
 9824  054142  053671  000025  042620          .WORD   PAR9,21.,MINSEC+DRIVE2
 9825  054150  053662  000025  042616          .WORD   PAR8,21.,MAXSEC+DRIVE2,0
 9826  054156  000000
 9827
 9828  054160  053635  000000  043114  TABLE3: .WORD   PAR5,0,MINCYL+DRIVE3
 9829  054166  053626  000000  043112          .WORD   PAR4,0,MAXCYL+DRIVE3
 9830  054174  053653  000022  043120          .WORD   PAR7,18.,MINTRK+DRIVE3
 9831  054202  053644  000022  043116          .WORD   PAR6,18.,MAXTRK+DRIVE3
 9832  054210  053671  000025  043124          .WORD   PAR9,21.,MINSEC+DRIVE3
 9833  054216  053662  000025  043122          .WORD   PAR8,21.,MAXSEC+DRIVE3,0
 9834  054224  000000
```

```
054226  053635  000000  043420  TABLE4: .WORD   PAR5,0,MINCYL+DRIVE4
054234  053626  000000  043416          .WORD   PAR4,0,MAXCYL+DRIVE4
054242  053653  000022  043424          .WORD   PAR7,18.,MINTRK+DRIVE4
054250  053644  000022  043422          .WORD   PAR6,18.,MAXTRK+DRIVE4
054256  053671  000025  043430          .WORD   PAR9,21.,MINSEC+DRIVE4
054264  053662  000025  043426          .WORD   PAR8,21.,MAXSEC+DRIVE4,0
054272  000000

054274  053635  000000  043724  TABLE5: .WORD   PAR5,0,MINCYL+DRIVE5
054302  053626  000000  043722          .WORD   PAR4,0,MAXCYL+DRIVE5
054310  053653  000022  043730          .WORD   PAR7,18.,MINTRK+DRIVE5
054316  053644  000022  043726          .WORD   PAR6,18.,MAXTRK+DRIVE5
054324  053671  000025  043734          .WORD   PAR9,21.,MINSEC+DRIVE5
054332  053662  000025  043732          .WORD   PAR8,21.,MAXSEC+DRIVE5,0
054340  000000

054342  053635  000000  044230  TABLE6: .WORD   PAR5,0,MINCYL+DRIVE6
054350  053626  000000  044226          .WORD   PAR4,0,MAXCYL+DRIVE6
054356  053653  000022  044234          .WORD   PAR7,18.,MINTRK+DRIVE6
054364  053644  000022  044232          .WORD   PAR6,18.,MAXTRK+DRIVE6
054372  053671  000025  044240          .WORD   PAR9,21.,MINSEC+DRIVE6
054400  053662  000025  044236          .WORD   PAR8,21.,MAXSEC+DRIVE6,0
054406  000000

054410  053635  000000  044534  TABLE7: .WORD   PAR5,0,MINCYL+DRIVE7
054416  053626  000000  044532          .WORD   PAR4,0,MAXCYL+DRIVE7
054424  053653  000022  044540          .WORD   PAR7,18.,MINTRK+DRIVE7
054432  053644  000022  044536          .WORD   PAR6,18.,MAXTRK+DRIVE7
054440  053671  000025  044544          .WORD   PAR9,21.,MINSEC+DRIVE7
054446  053662  000025  044542          .WORD   PAR8,21.,MAXSEC+DRIVE7,0
054454  000000


054456  000000  000000  000000  CYLDER: .WORD   0,0,0,0         ;HEADER BUFFER FOR 'READHD' ROUTINE
054464  000000

        054466                  ENDPGM  =       .               ;LAST LOCATION OF PROG + 2

                                ;;*************************************************************

                                ;ROUTINE TO GET THE DATE AND THE OPERATOR FROM THE OPERATOR
                                ;CALL:
                                ;       JSR     PC,OPRDAT
                                ;       RETURN
                                ;
                                ;NOTE: THIS ROUTINE IS ENTERED ONLY AT INITIAL START

054466  104401  054600          OPRDAT: TYPE    ,ENTDAT         ;'ENTER DATE'
054472  104411                          RDLIN                   ;READ THE ENTRY
054474  012605                          MOV     (SP)+,R5        ;PUT THE ENTRY ADDRESS INTO R5
054476  112537  001220                  MOVB    (R5)+,DATE      ;STORE THE DATE
054502  112537  001221                  MOVB    (R5)+,DATE+1    ;STORE THE DATE
054506  112537  001222                  MOVB    (R5)+,DATE+2    ;STORE THE DATE
054512  112537  001223                  MOVB    (R5)+,DATE+3    ;STORE THE DATE
054516  112537  001224                  MOVB    (R5)+,DATE+4    ;STORE THE DATE
```

```
                                        MOVB    (R5)+,DATE+5    ;STORE THE DATE
                                        MOVB    (R5)+,DATE+6    ;STORE THE DATE
                                        MOVB    (R5)+,DATE+7    ;STORE THE DATE
                                        TYPE    ,ENTID          ;'ENTER OPERATOR I.D.'
                                        RDLIN                   ;READ THE ENTRY
                                        MOV     (SP)+,R5        ;ENTRY ADDRESS
                                        MOVB    (R5)+,OPERID    ;STORE THE I.D.
                                        MOVB    (R5)+,OPERID+1  ;STORE THE I.D.
                                        MOVB    (R5)+,OPERID+2  ;STORE THE I.D.
                                        MOVB    (R5)+,OPERID+3  ;STORE THE I.D.
                                        MOVB    (R5)+,OPERID+4  ;STORE THE I.D.
                                        MOVB    (R5)+,OPERID+5  ;STORE THE I.D.
                                        RTS     PC              ;RETURN

054600 005015 047105 042524     ENTDAT: .ASCIZ  <CR><LF>/ENTER DATE: /
054606 020122 040504 042524
054614 020072     000
054617    105 052116 051105     ENTID:  .ASCIZ  /ENTER OPERATOR I.D.: /
054624 047440 042520 040522
054632 047524 020122 027111
054640 027104 020072     000
054646                          .EVEN

                                .SBTTL  ROUTINE TO SIZE MEMORY

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

054646 010046                   $SIZE:  MOV     R0,-(SP)        ;;SAVE R0 ON THE STACK
054650 010146                           MOV     R1,-(SP)        ;;SAVE R1 ON THE STACK
054652 013746 000004                    MOV     @#ERRVEC,-(SP)  ;;SAVE PRESENT ERROR VECTOR PS & PC
054656 013746 000006                    MOV     @#ERRVEC+2,-(SP)
054662 010600                           MOV     SP,R0           ;;SAVE THE STACK POINTER
                                ;;SET THE ERRVEC PS TO THE PRESENT PS
054664 104400                           TRAP                    ;;PUSH OLD PSW AND PC ON STACK
054666 012637 000006                    MOV     (SP)+,@#ERRVEC+2        ;;SAVE THE PSW IN @#ERRVEC+2
054672 012737 054712 000004             MOV     #2$,@#ERRVEC    ;;SET FOR TIMEOUT
054700 012701 020000                    MOV     #20000,R1       ;;FIRST ADDRESS
054704 005711                   1$:     TST     (R1)            ;;TEST THIS ADDRESS
054706 005721                           TST     (R1)+           ;;STEP TO NEXT ADDRESS
054710 000775                           BR      1$              ;;TRY ANOTHER
054712 162701 000002            2$:     SUB     #2,R1           ;;DROP BACK
054716 010006                           MOV     R0,SP           ;;RESTORE THE STACK
054720 012637 000006                    MOV     (SP)+,@#ERRVEC+2 ;;RESTORE ERROR VECTOR
054724 012637 000004                    MOV     (SP)+,@#ERRVEC
054730 010137 054742                    MOV     R1,$LSTAD       ;;LAST ADDRESS
054734 012601                           MOV     (SP)+,R1        ;;RESTORE R1
054736 012600                           MOV     (SP)+,R0        ;;RESTORE R0
054740 000207                           RTS     PC
054742 000000                   $LSTAD: .WORD   0               ;;CONTAINS THE LAST ADDRESS

                                .SBTTL  BUSADR - GET BUS ADDRESS AND VECTOR ADDRESS FOR RH11
                                ;THIS ROUTINE IS USED TO INSURE THE BUS ADDRESS
```

```
                                ;CC THE RH11 IS SETUP FOR THE PROPER ADDRESS.
                                ;IT WILL ALSO READ THE ADDRESS FROM THE TTY IF
                                ;REQUIRED.
                                ;NOTE: THIS ROUTINE DESTROYS R0-R4
                                ;CALL
                                ;
                                ;       JSR     PC,BUSADR
                                ;       RETURN
                                ;
054744  177700          HIAD:   .WORD 177700
054746  000776          HIVEC:  .WORD 776
054750  000000          BOUND:  .WORD 0
                        ;
054752  005737  001260  BUSADR: TST     CHGADR          ;INPUT FROM TTY REQUESTED?
054756  001456                 BEQ     7$              ;NO--BRANCH
054760  005037  001260         CLR     CHGADR          ;YES--CLEAR THE REQUEST FLAG
054764  012700  001170  1$:    MOV     #$RPADR,R0      ;FIRST ADDRESS
054770  104401  055176         TYPE    ,MRPCS1         ;"RPCS1="
054774  012046                 MOV     (R0)+,-(SP)     ;PRESENT RPCS1 ADDRESS
054776  104402                 TYPOC                   ;TYPE IT
055000  104401  052155         TYPE    ,LINSP          ;2 SPACES
055004  104411                 RDLIN                   ;GET THE ENTRY
055006  012601                 MOV     (SP)+,R1        ;ADDRESS OF ASCII TEXT
055010  013737  054744  054750 MOV     HIAD,BOUND      ;SET THE ADDRESS MAX
055016  004537  055220         JSR     R5,CK.NUM       ;CHECK THE NUMBER
055022  055042                 3$                      ;CARRIAGE RETURN ONLY ENTERED
055024  055114                 7$                      ;PERIOD ONLY ENTERED
055026  054764                 1$                      ;ILLEGAL INPUT
055030  055036                 2$                      ;TERMINATED WITH A CARRIAGE RETURN
055032  054764                 1$                      ;TERMINATED WITH A ","
055034  055110                 4$                      ;TERMINATED WITH A "."
055036  010260  177776  2$:    MOV     R2,-2(R0)       ;SAVE NEW RPCS1
055042  104401  055207  3$:    TYPE    ,MRHVEC         ;"RHVEC="
055046  012046                 MOV     (R0)+,-(SP)     ;PRESENT RH11 VECTOR ADDRESS ON THE STACK
055050  104402                 TYPOC                   ;TYPE IT
055052  104401  052155         TYPE    ,LINSP          ;2 SPACES
055056  104411                 RDLIN                   ;READ THE ENTRY
055060  012601                 MOV     (SP)+,R1        ;ASCII TEXT ADDRESS
055062  013737  054746  054750 MOV     HIVEC,BOUND     ;SET THE VECTOR-MAX
055070  004537  055220         JSR     R5,CK.NUM       ;CHECK THE NUMBER
055074  055114                 7$                      ;CARRIAGE RETURN ONLY ENTERED
055076  055114                 7$                      ;PERIOD ONLY ENTERED
055100  055042                 3$                      ;ILLEGAL INPUT
055102  055110                 4$                      ;TERMINATED WITH A CARRIAGE RETURN
055104  055042                 3$                      ;TERMINATED WITH A ","
055106  055110                 4$                      ;TERMINATED WITH A "."
055110  010260  177776  4$:    MOV     R2,-2(R0)       ;SAVE INPUT
055114  013701  000004  7$:    MOV     ERRVEC,R1       ;SAVE THE ERROR VECTOR
055120  012737  055154  000004 MOV     #8$,ERRVEC      ;SETUP FOR TRAP
055126  005777  124036         TST     @$RPADR         ;CHECK FOR RH11
055132  010137  000004         MOV     R1,ERRVEC       ;RESTORE ERROR VECTOR
055136  012700  001170         MOV     #$RPADR,R0      ;FIRST ADDRESS OF NEW PARAMETERS
055142  012701  033332         MOV     #RPADR,R1       ;FIRST ADDRESS OF WHERE TO PUT THEM
055146  012021                 MOV     (R0)+,(R1)+     ;BUS ADDRESS
055150  012021                 MOV     (R0)+,(R1)+     ;VECTOR ADDRESS
055152  000207                 RTS     PC              ;RETURN
```

```
10003  055154  010137  000004          8$:     MOV     R1,ERRVEC       ;RESTORE ERROR VECTOR
10004  055160  022626                          CMP     (SP)+,(SP)+     ;CLEAN OFF THE STACK
10005  055162  104006                          ERROR   6               ;REPORT THE ERROR
10006  055164  005737  000042                  TST     @#42            ;IS THERE A MONITOR?
10007  055170  001675                          BEQ     1$              ;NO--GO ASK FOR ADDRESS
10008  055172  000137  005344                  JMP     $GET42          ;GO TO END OF PROGRAM
10009
10010  055176  050122  051503  020061  MRPCS1: .ASCIZ   @RPCS1 = @
10011  055204  020075      000
10012  055207      122  053110  041505  MRHVEC: .ASCIZ   @RHVEC = @
10013  055214  036440  000040
10014
10015                                   .SBTTL  CK.NUM - CHECK NUMBER (OCTAL)
10016                                   ;THIS ROUTINE CHECKS AN ASCIZ STRING FOR LEGAL CHARACTERS
10017                                   ;AND FORMS AN OCTAL NUMBER IN R2
10018                                   ;CALL:
10019                                   ;       MOV     #ADR,R1         ;ADDRESS OF ASCIZ STRING
10020                                   ;       MOV     #NUM,R2         ;MAX SIZE OF INPUT NUMBER
10021                                   ;       JSR     R5,CK.NUM       ;GO FORM THE NUMBER
10022                                   ;       RETURN  ADR1            ;"CR" ONLY ENTERED -- R2 = 0
10023                                   ;       RETURN  ADR2            ;"PERIOD" ONLY ENTERED -- R2 = 0
10024                                   ;       RETURN  ADR3            ;ILLEGAL CHARACTER IN THE INPUT STRING
10025                                   ;       RETURN  ADR4            ;"CR" ENTERED -- R2 = NUMBER
10026                                   ;       RETURN  ADR5            ;"COMMA" -- R2 = NUMBER
10027                                   ;       RETURN  ADR6            ;"PERIOD" -- R2 = NUMBER
10028
10029  055220  010446                  CK.NUM: MOV     R4,-(SP)        ;SAVE R4
10030  055222  010346                          MOV     R3,-(SP)        ;SAVE R3
10031  055224  010246                          MOV     R2,-(SP)        ;SAVE R2
10032  055226  005004                          CLR     R4              ;RETURN POINTER
10033  055230  005003                          CLR     R3              ;START NUMBER AT ZERO
10034  055232  005002                          CLR     R2              ;STORE RESULT
10035  055234  004537  027560                  JSR     R5,CK.CHR       ;CHECK ONE CHARACTER
10036  055240  055340                          6$                      ;ILLEGAL CHARACTER
10037  055242  055344                          8$                      ;CARRIAGE RETURN
10038  055244  055340                          6$                      ;" "
10039  055246  055342                          7$                      ;"."
10040  055250  055254                          1$                      ;DIGIT 0-7
10041  055252  055340                          6$                      ;DIGIT 8-9
10042  055254  062705  000004          1$:     ADD     #4,R5           ;INCREMENT RETURN PAST "CR" AND "PERIOD" ONLY RETURNS
10043  055260  006303                  2$:     ASL     R3              ;FOR THE OCTAL NUMBER IN R3
10044  055262  103426                          BCS     6$              ;DON'T LET IT GET TO BIG
10045  055264  006303                          ASL     R3
10046  055266  103424                          BCS     6$
10047  055270  006303                          ASL     R3
10048  055272  103422                          BCS     6$
10049  055274  060203                          ADD     R2,R3
10050  055276  004537  027560                  JSR     R5,CK.CHR       ;CHECK ONE CHARACTER
10051  055302  055344                          9$                      ;ILLEGAL CHARACTER
10052  055304  055326                          5$                      ;CARRIAGE RETURN
10053  055306  055324                          4$                      ;" "
10054  055310  055316                          3$                      ;"."
10055  055312  055260                          2$                      ;DIGIT 0-7
10056  055314  055344                          8$                      ;DIGIT 8-9
10057  055316  105711                  3$:     TSTB    (R1)            ;DOES A "CR" FOLLOW THE "PERIOD"
10058  055320  001011                          BNE     8$              ;BR IF NOT
```

```
10059  055322  005724                    TST     (R4)+           ;INCREMENT THE RETURN
10060  055324  005724            4$:     TST     (R4)+           ;INCREMENT THE RETURN INDEX
10061  055326  005724            5$:     TST     (R4)+           ;INCREMENT THE RETURN INDEX
10062  055330  023703  054750            CMP     BOUND,R3        ;INPUT VALUE TOO LARGE?
10063  055334  101003                    BHI     8$              ;BR IF IT IS
10064  055336  000401                    BR      7$              ;BR IF NOT
10065  055340  005725            6$:     TST     (R5)+           ;INCREMENT THE RETURN ADDRESS
10066  055342  005725            7$:     TST     (R5)+           ;INCREMENT THE RETURN ADDRESS
10067  055344  060405            8$:     ADD     R4,R5           ;SETUP FOR PROPER RETURN
10068  055346  010302                    MOV     R3,R2           ;LOAD ENTERED VALUE
10069  055350  005726                    TST     (SP)+           ;CLEAN OFF THE STACK
10070  055352  012603                    MOV     (SP)+,R3        ;RESTORE R3
10071  055354  012604                    MOV     (SP)+,R4        ;RESTORE R4
10072  055356  011505                    MOV     (R5),R5         ;GET RETURN ADDRESS
10073  055360  000205                    RTS     R5              ;RETURN
10074
10075
10076  055362  005015  020040  020040  TITLE:  .ASCII  <CR><LF>/          ZZ-CZRJD-C/<CR><LF>
10077  055370  020040  020040  020040
10078  055376  055132  041455  051132
10079  055404  042112  041455  005015
10080  055412  050122  032060  032457          .ASCIZ  @RP04/5/6 MULTI-DRIVE EXERCISER PROGRAM@<CR><LF><LF>
10081  055420  033057  046440  046125
10082  055426  044524  042055  044522
10083  055434  042526  042440  042530
10084  055442  041522  051511  051105
10085  055450  050040  047522  051107
10086  055456  046501  005015  000012
10087  055464  005015  047524  052040  LOADRV: .ASCII  <CR><LF>/TO TEST DRIVE 0, REPLACE THE 'XXDP' PACK ON DRIVE 0/<CR><LF>
10088  055472  051505  020124  051104
10089  055500  053111  020105  026060
10090  055506  051040  050105  040514
10091  055514  042503  052040  042510
10092  055522  023440  054130  050104
10093  055530  020047  040520  045503
10094  055536  047440  020116  051104
10095  055544  053111  020105  006460
10096  055552     012
10097  055553     127  052111  020110          .ASCII  /WITH ANOTHER PACK, CLEAR MEMORY LOCATION 40,/<CR><LF>
10098  055560  047101  052117  042510
10099  055566  020122  040520  045503
10100  055574  020054  046103  040505
10101  055602  020122  042515  047515
10102  055610  054522  046040  041517
10103  055616  052101  047511  020116
10104  055624  030064  006454     012
10105  055631     101  042116  051040          .ASCIZ  /AND RESTART THE PROGRAM/<CR><LF>
10106  055636  051505  040524  052122
10107  055644  052040  042510  050040
10108  055652  047522  051107  046501
10109  055660  005015     000
10110  055663     015  051412  051531  NOLOAD: .ASCIZ  <CR><LF>/SYSTEM HAS 16K MEMORY, 'XXDP' LOADER WILL BE OVERWRITTEN/<CR><L
10111  055670  042524  020115  040510
10112  055676  020123  033061  020113
10113  055704  042515  047516  054522
10114  055712  020054  054047  042130
```

```
10115  055720  023520  046040  040517
10116  055726  042504  020122  044527
10117  055734  046114  041040  020105
10118  055742  053117  051105  051127
10119  055750  052111  042524  006516
10120  055756  000012
10121
10122          000001                  .END
```

```
ABNRML  026524
ABS     = 000200
ACK     = 000123
ACL     = 000040
ACTDRV  033250
ACTSTR  033251
ACU     = 100000
AOE     = 001000
ASGND   052720
ASGN1   024214
ASGN2   024274
ASGN3   024346
ASGN4   024464
ASGN5   024474
ASGN6   024502
ASGN7   024512
ASKPAR  053552
ASNERR  026416
ASNLST  001462
ASNMSG  026436
ASSIGN  024206
ATA     = 100000
ATABIT  033320
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  053276
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
BOUND   054750
BPTVEC= 000014
BUFTBL  001616
BUSADR  054752
BUSY    053320
CFLAG   001262
CHGADR  001260
CI1     034672
CI3     035000
CI4     035106
CI5     035464
CI6     035506
CI7     035522
CI7B    035550
CI8     035622
CKBUS   013070
CKCLK   022412
CKCLK1  022506
CKCLK2  022554
CKCLK3  022604
CKERR   012770
CKFMT   011072
CKHCE   011266
CKSWR   = 104407
CK.CHR  027560
CK.DEC  027532
CK.DIG  027632
CK.NUM  055220
CK.OCT  027504
CLKFLG  001210
CLOCK   023546
CLR     = 000040
CLRDPB  025164
CLRQUE  041432
CMCNT   001312
CMCYL   001314
CMDAT   013436
CMHED   013342
CMPAR   013154
CMPARD  013172
CMPLMT  001414
CMPRES  017530
CMPRT   013702
CMPRX   013674
CMSEC   001316
CMSTR   013264
CMTRK   001317
COLON   053207
COMTBL  001760
CR      = 000015
CRLF    = 000200
CSF     = 000002
CSU     = 000010
CYLDER  054456
CYLIMT  001350
DATAPK  025140
DATAO   003026
DATAI   003066
DATE    001220
DATEIS  053211
DCK     = 100000
DCKER   007570
DCKER1  007722
DCL     = 000100
DCU     = 000001
DDISP   = 177570
DEASGN  024562
DEASSG  052650
DE1     = 000040
DFF20   = 000002
DH1     047460
DH14    047636
DH15    047742
DH16    050041
DH2     047465
DH3     047542
DH4     047570
DH6     047627
DIGB    = 000004
DISPLA  001142
DISPLY= 104414
DISPRE  000174
DLT     = 100000
DL64    = 000020
DMD     = 000001
DONE    007264
DPINT   033224
DPR     = 000400
DPRQS   033234
DRIVE   001242
DRIVE0  041670
DRIVE1  042174
DRIVE2  042500
DRIVE3  043004
DRIVE4  043310
DRIVE5  043614
DRIVE6  044120
DRIVE7  044424
DRNUM   052673
DROP    026446
DROPNG  052562
DRQ     = 004000
DRVACT  033174
DRVCLR= 000111
DRVER   011040
DRVINT  033562
DRVPRM  025360
DRVQUE  041530
DRVSTA  033204
DRVTYP  033214
DRY     = 000200
DSWR    = 177570
DTE     = 010000
DTEER   011676
DTSY    = 000200
DTUW    033316
DT00    = 000001
DT01    = 000002
DT02    = 000004
DT03    = 000010
DT04    = 000020
DT05    = 000040
DT06    = 000100
DT07    = 000200
DT08    = 000400
DT1     050112
DT14    050162
DT15    050204
DT16    050226
DT2     050116
DT3     050134
DT4     050144
DT6     050156
DUNIT   001464
DVA     = 004000
ECBAD0  001334
ECBAD1  001342
ECBIT   001320
ECC     014302
ECCX    015034
ECC1    014700
ECC2    015030
ECGD    001332
ECGD1   001340
ECH     = 000100
ECI     = 004000
ECMSK0  001324
ECMSK1  001326
ECSEC   001322
ECWRD   001330
ECWRD1  001336
EMPTYQ  041510
EMTVEC= 000030
EM1     045020
EM10    045312
EM11    045355
EM12    045410
EM13    045441
EM14    045513
EM15    045536
EM2     045063
EM20    045572
EM21    045613
EM22    045644
EM23    045717
EM24    045776
EM25    046044
EM26    046125
EM27    046152
EM3     045121
EM30    046207
EM31    046241
EM32    046304
EM33    046337
EM34    046414
EM35    046456
EM36    046514
EM37    046545
EM4     045157
EM40    046627
EM41    046665
EM42    046731
EM43    047012
EM44    047057
EM45    047143
EM46    047231
EM5     045214
EM50    047303
EM51    047372
EM6     045250
EM60    047435
ENDCMP  014156
ENDCN   001352
ENDCON  001372
ENDET   001430
ENDPAS  052615
```

```
ENDPGM= 054466
ENDSEK  001376
ENDSK   001356
ENDTST  052632
ENTADR  053146
ENTCOM  053033
ENTDAT  054600
ENTDRV  053060
ENTID   054617
ENTLMT  053105
ENTPR   004772
EOP     026552
EOPX    027024
EOP1    026604
EOP2    026626
ERCTR   001306
ERPRC1  006644
ERPROC  006630
ERR     = 040000
ERRVEC= 000004
EXT1    = 000001
EXT10   = 000010
EXT2    = 000002
EXT20   = 000020
EXT4    = 000004
EXT40   = 000040
FACTOR  015670
FAIRNS  001254
FALPAR  007030
FALPR1  007040
FEN     = 000200
FER     = 000020
FILBUF  016242
FMTER   012072
FMT22   = 010000
FORMAT  001416
FRSTER  001300
F1      = 000002
F2      = 000004
F3      = 000010
F4      = 000020
F5      = 000040
GENDPB  044730
GENREG  044750
GETADR  025752
GETBUF  015672
GETID   025642
GETPAR  017322
GETPAT  017274
GETREG= 000141
GETREM  027026
GETREQ  041604
GO      = 000001
GODRIV  016370
GRV     = 000010
GTSWR   = 104406
HCE     = 000200
HCEER   012150
HCI     = 002000
HCRC    = 000400
HCRCER  010710
HEDLIN  053244
HIAD    054744
HIVEC   054746
HOUR    001266
HT      = 000011
HZ      001212
IAE     = 002000
IAEER   012012
IDIS    053222
IDLE    006202
IDLE1   006300
IE      = 000100
ILF     = 000001
ILR     = 000002
INCHRD  023330
INCMIS  023400
INCSKI  023354
INCSOF  023304
INCTOT  023424
INTDON  053341
INTRVL  001410
INVLD   053011
IOTVEC= 000020
IR      = 000100
ISR     036214
IXE     = 004000
KSR     023710
KSR1    023744
LA      036056
LACNT   033262
LF      = 000012
LIMIT   001310
LINA3   050460
LINB3   050532
LINB5   050707
LINB6   051022
LINCA3  050511
LINC6   051055
LINDA3  050522
LINDEC  022274
LIND5   050672
LINEN3  050443
LINEO5  051010
LINEP5  050777
LINE1   020200
LINE2   020244
LINE2A  020414
LINE2B  020432
LINE3   020652
LINE3A  020660
LINE3B  020666
LINE3C  020700
LINE3D  020710
LINE3E  020756
LINE3F  021044
LINE4   021326
LINE5   021416
LINE5A  021516
LINE5B  021610
LINE6   021652
LINE6A  021664
LINE6B  021672
LINE6C  021700
LINE6D  021706
LINE7   021762
LINE7A  022110
LINE8   022230
LING6   051104
LINKDV  027054
LINM3   050343
LINM4   050606
LINN3   050361
LINOCT  022242
LINP3   050405
LINP5   050725
LINR6   051122
LINSP   052155
LINSPO  052156
LINSS3  050571
LINST3  050554
LINS3   050426
LINS4   050625
LINS5   050743
LINT3   050500
LINU06  051133
LINW3   050542
LINX4   050637
LIN10A  051571
LIN10B  051625
LIN10C  051666
LIN10H  051773
LIN11H  052060
LIN2C   050242
LIN2P   050263
LIN2S   050307
LIN3.1  021116
LIN3.3  021230
LIN3.4  021262
LIN4SP  052153
LIN6.1  021714
LIN6.2  021736
LIN7M   051160
LIN7O   051206
LIN7P   051216
LIN7R   051313
LIN7S   051236
LIN7T   051265
LIN7X   051277
LIN8M   051330
LIN9B   051365
LIN9E   051512
LIN9G   051542
LIN9H   051414
LIN9I   051472
LKPAR   004746
LOADRV  055464
LST     = 002000
LSTAD   001256
MAIN    005374
MAIN1   005522
MAIN2   005642
MAIN3   006076
MASK    001250
MATCH   014224
MAXCYL= 000106
MAXDL   001404
MAXER   001406
MAXSEC= 000116
MAXTRK= 000112
MCLK    = 000002
MCPE    = 020000
MCPEMX  033330
MHS     = 001000
MINCYL= 000110
MINSEC= 000120
MINTRK= 000114
MINUTE  001270
MINX    = 000004
MNDLTA  033344
MNTBL   002010
MOH     = 020000
MOL     = 010000
MONTR   005266
MPE     = 000400
MRD     = 000020
MRHVEC  055207
MRPCSI  055176
MSE     = 000020
MSTCK   = 000010
MWR     = 000040
MXDLTA  033342
MXF     = 001000
MXLACT  033340
MXWNDW  033346
M.DPID  027122
M.DP40  027160
M.DP41  027214
M.DP42  027224
M.DP44  027256
M.DP50  027270
NBA     = 100000
NED     = 010000
NEDCLK  052733
NEM     = 004000
NEWASN  024552
NEWUNT  001506
NHS     = 002000
NOLOAD  055663
NOMTCH  012546
NONE    053267
NOTAVL  052315
NOTPRS  052300
NOTPRT  001426
NOTRP   052257
NOTSAF  052334
OCYL    = 100000
OFFCOD  002220
OFFSET= 000115
OFFST   015272
OFLIN   007136
OFMSGA  002724
OFMSGO  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   011566
OPIER1  011632
```

```
OPRDAT  054466
OPT     034412
OPTBL   001766
OR      =  000200
ORDERQ  001440
PACK    001216
PAR     =  000010
PARENT  026272
PARER   011720
PARLST  053454
PARQ    001574
PAR1    053602
PAR10   053700
PAR11   053707
PAR14   053715
PAR15   053723
PAR16   053732
PAR19   053740
PAR2    053610
PAR20   053747
PAR3    053617
PAR4    053626
PAR5    053635
PAR6    053644
PAR7    053653
PAR8    053662
PAR9    053671
PASCNT  001402
PAT     =  000020
PCLOCK  001206
PDONE   052552
PERIOD  053005
PGE     =  002000
PGM     =  001000
PIP     =  020000
PIRQ    =  177772
PIRQVE=  000240
PLU     =  020000
POPQUE  041626
POSER   011512
PROCES  006540
PRTBAD  015042
PRTIM   007226
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  006726
PWRVEC=  000024
QCNT    041140
QDRV0   041232
QDRV1   041252
QDRV2   041272
QDRV3   041312
QDRV4   041332
QDRV5   041352
QDRV6   041372
QDRV7   041412
QINPT   041150
QOUTPT  041170
QSTART  041210
QSTOP   041212
QTERM =  041432
QUES    053007
QVCON   001362
QVSEK   001366
RANCYL  016702
RANPAT  017150
RANSEC  016526
RANSIZ  017014
RANTRK  016602
RANWRT  017170
RANXIT  017160
RATIO   001422
RAW     =  000020
RDCHR =  104410
RDDAT =  000171
RDHD    =  000173
RDLIN =  104411
RDY     =  000200
RD.ADR  040432
RD.RP   040406
RD.RP1  040430
RD.RP2  040550
RD.RP3  040554
RD.RP4  040560
RD.WRD  040434
READDR  022314
READHD  015316
READIN=  000121
RECAL =  000107
RECALT  015240
RECALO  015244
REDAPK  025152
REFMT   006374
REFMTX  006536
RELBUF  016026
RELSE =  000113
REPLZ   027300
RESREG=  104413
RESVEC=  000010
RETRY   001252
RMR     =  000004
RNOP    =  000101
RPADR   033332
RPAS    =  000016
RPBA    =  000004
RPCA    =  000034
RPCC    =  000036
RPCS1 =  000000
RPCS2 =  000010
RPDA    =  000006
RPDB    =  000022
RPDS1 =  000012
RPDT    =  000026
RPEC1 =  000044
RPEC2 =  000046
RPERRS  033164
RPER1 =  000014
RPER2 =  000040
RPER3 =  000042
RPINIT  033350
RPLA    =  000020
RPMR    =  000024
RPOF    =  000032
RPSN    =  000030
RPTMR   037660
RPVEC   033334
RPWC    =  000002
RP04    034120
RP04B   052364
RP05    052371
RP06    052376
RTC     =  000117
RTNCTR  015214
R6      =%000006
R7      =%000007
S       050402
SAVEFG  033272
SAVER1  001302
SAVER5  001304
SAVREG=  104412
SC      036542
SCMND   024670
SC1     =  000100
SC10    =  001000
SC11    037360
SC12    037450
SC13    037520
SC2     =  000200
SC20    =  002000
SC3     036606
SC4     036612
SC5     036624
SC6     037014
SC6A    037124
SC7     037252
SC8     037330
SDETAL  022746
SEARCH=  000131
SECLMT  001344
SECOND  001272
SEEK    =  000105
SEEKFG  033274
SELDRV=  000145
SELPAR  016446
SETFMT=  000143
SETUP   004154
SETVEC  005020
SET.IE  041066
SHDTYP  022724
SIXTEE  001274
SIZMEM  004576
SKI     =  040000
SKIER   012332
SLASH   053576
SPOTCK  020042
SRCHWT  033246
STACK   =  001100
START   004106
START1  004116
START2  004122
STATHD  052403
STATIN  001214
STATIS  015526
STATPR  022614
STKLMT=  177774
STNDAT  002762
ST0     037752
ST01    040002
ST02    040200
ST03    040250
ST05    040274
ST06    040302
ST07    040340
ST08    040370
ST09    040400
SVRH11  040750
SWR     001140
SWREG   000176
SWTIM   007100
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  052344
T       050356
TABLE   053756
TABLE0  053776
TABLE1  054044
TABLE2  054112
TABLE3  054160
TABLE4  054226
TABLE5  054274
TABLE6  054342
TABLE7  054410
TAP     =  040000
TBITVE=  000014
TD      036260
TDF     =  000040
TIMER   033276
TITLE   055362
TKVEC   =  000060
TPVEC   =  000064
TRAPVE=  000034
TRE     =  040000
TRFER   012232
TRKLMT  001346
TRK1    =  004000
TRK10   =  040000
TRK2    =  010000
TRK20   =  100000
TRK4    =  020000
TRNSWT  033244
TRTVEC=  000014
```

```
TUF    = 000100
TYPDS  = 104405
TYPE   = 104401
TYPEST   022676
TYPOC  = 104402
TYPON  = 104404
TYPOS  = 104403
TYPRI4   027430
UCPAR    007016
ULDFLG   033252
UNIT     001246
UNLOAD=  000103
UNS    = 040000
UNSAF    012466
UNTASN   052231
UNTMSG   052160
UNTNOT   052207
UNTOFF   052166
UNTON    052177
UPE    = 020000
US1    = 000001
US2    = 000002
US4    = 000004
UWR    = 000010
VUF    = 000002
VU30   = 010000
VV     = 000100
WAIT     001552
WAO    = 000002
WC       036434
WCE    = 040000
WCF    = 000040
WCFER    012370
WCKD   = 000151
WCKER    010362
WCKHD  = 000153
WCSEL    001420
WCU    = 000001
WLE    = 004000
WLEER    012044
WRL    = 004000
WRTDAT=  000161
WRTHD  = 000163
WRTPK    017546
WRTPK1   017576
WRTPK2   017756
WRTPK3   020002
WRTPK4   020034
WRT.AD   040652
WRT.RP   040562
WRT.R1   040646
WRT.R2   040732
WRT.R3   040740
WRT.R4   040744
WRT.R5   040746
WRT.WD   040650
WRU    = 000400
WSU    = 000004
ZROIND   001276
$AUTOB   001134
$BDADR   001122
$BDDAT   001126
$BDSEC=  000124
$BELL    001160
$BUF   = 000006
$CHARC   031250
$CKSWR   031726
$CMTAG   001100
$CM3   = 000000
$CNTLC   030522
$CNTLG   032335
$CNTLU   032330
$CODE  = 000024
$COMND=  000002
$CRLF    001165
$CYL   = 000012
$DBLK    031716
$DB2D    032562
$DB20    032756
$DECVL   032742
$DOAGN   005370
$DRVID=  000224
$DSPLY   027456
$DTBL    031706
$ENDAD   005360
$ERFLG   001103
$ERMAX   001115
$ERROR   030530
$ERRPC   001116
$ERRTB   004026
$ERRTY   030700
$ERTTL   001112
$FAIR  = 000072
$FILLC   001156
$FILLS   001155
$FIRST=  000122
$FMT   = 000001
$GDADR   001120
$GDDAT   001124
$GET42   005344
$GTSWR   031776
$HARD  = 000062
$HD    = 000000
$HINUM   032462
$ICNT    001104
$INTAG   001135
$ITEMB   001114
$LF      001166
$LKCSB   001176
$LKCSR   001174
$LKS     001202
$LLVEC   001204
$LONUM   032464
$LPADR   001106
$LPERR   001110
$LPVEC   001200
$LSTAD   054742
$MISPO=  000066
$MNEW    032353
$MSWR    032342
$NCODE=  000074
$NCYL  = 000100
$NEXT  = 000104
$NPATC=  000075
$NSEC  = 000076
$NTRK  = 000077
$NULL    001154
$NWRDL=  000102
$OCNT    031476
$OCTVL   033060
$OMODE   031500
$OPERC=  000036
$PACK  = 000026
$PASS    001100
$PASSC=  000070
$PATTC=  000030
$POSIT=  000042
$PREVA=  000032
$PREVO=  000027
$PSEL  = 000003
$QUES    001164
$RAND    032364
$RDCHR   032210
$RDLIN   030232
$RDSZ  = 000001
$READ  = 000052
$REG   = 000014
$RESRE   032524
$RETRY   015400
$RPADR   001170
$RPAS  = 000252
$RPBA  = 000240
$RPCA  = 000270
$RPCC  = 000272
$RPCS1=  000234
$RPCS2=  000244
$RPDA  = 000242
$RPDB  = 000256
$RPDS1=  000246
$RPDT  = 000262
$RPEC1=  000300
$RPEC2=  000302
$RPER1=  000250
$RPER2=  000274
$RPER3=  000276
$RPLA  = 000254
$RPMR  = 000260
$RPOF  = 000266
$RPSN  = 000264
$RPVEC   001172
$RPWC  = 000236
$SAVRE   032466
$SB2D    027770
$SB20    030020
$SEC   = 000010
$SETUP=  000146
$SIZE    054646
$SKI   = 000064
$SOFT  = 000060
$SSEC  = 000022
$STUP  = 177777
$SUPRS   027370
$SVPC  = 000210
$SWR   = 122000
$TATUS=  000016
$TERM  = 000032
$TIME    023450
$TKB     001146
$TKINT   030050
$TKS     001144
$TKSRV   030100
$TN    = 000001
$TNPWR   032672
$TOTAL=  000056
$TPB     001152
$TPFLG   001157
$TPS     001150
$TRANS=  000046
$TRAP    033076
$TRAP2   033120
$TRK   = 000011
$TRP   = 000015
$TRPAD   033132
$TSTNM   001102
$TTYIN   030510
$TYPDS   031502
$TYPE    031034
$TYPEC   031204
$TYPEX   031252
$TYPOC   031300
$TYPON   031314
$TYPOS   031254
$WRDL  = 000020
$WRDM  = 000004
$OFILL   031477
.      = 055760

. ABS.   055760      000

 ERRORS DETECTED:  0

 RM03:CZRJDC,CZRJDC,SEQ/SOL/NL:MC:MD:CND=RP0456.D11,RM03:CZRJDC.P11
 RUN-TIME:  30 26 1 SECONDS
 RUN-TIME RATIO: 577/57=9.9
 CORE USED:  40K  (79 PAGES)
```

J15